Optimizing resource management during business process execution: A case study

Johannes Pflug
Research Group Workflow Systems and Technology
University of Vienna
Vienna, Austria
Email: johannes.pflug@univie.ac.at

Stefanie Rinderle-Ma
Research Group Workflow Systems and Technology
University of Vienna
Vienna, Austria
Email: stefanie.rinderle-ma@univie.ac.at

***Abstract*—Dynamic Instance Queuing (DIQ) is a promising technique to optimize resource management during business process execution. It is domain-independent, does not require any predefined knowledge, and can be applied complementary to other optimization strategies. It represents a compromise between a non-optimized process execution and comprehensive business process redesign approaches which require a lot of time, money and knowledge. This paper reports on practical experiences and lessons learned in applying, implementing, and deploying a runtime-based resource optimization approach such as DIQ. In addition to new concepts such as handling multiple resources and several optimizations, an extensive application of the DIQ approach in a real-world scenario from the service domain is presented.**

I. Introduction

"Organizations are undergoing major transformations – to shift to digital business, become more customer-centric, and keep pace with regulatory changes. Any transformation impacts business processes, often requiring dramatic changes to how people work. Yet over 70% of transformation initiatives fail." [1]. This forces companies to think about optimizing their business process models beyond classical redesign conducted typically at design time and requiring additional knowledge. Instead shifting optimization to runtime is a promising approach, as it will not be always possible to provide additional knowledge such as rules for the optimization. Hence, flexible, domain and knowledge independent approaches are needed. One such approach is Dynamic Instance Queuing (DIQ) [2], [3], [4] that enables the optimization of instance processing at critical resources at runtime without any further knowledge. The basic idea is depicted in Fig. 1: Incoming instances at a critical activity are first collected (step 1). When the first resource shifts to idle state due to a lack of instances, all waiting instances are classified into groups (step 2). The classification is based on instance attributes. The instance clusters are then transferred to the buffers of the queuing system where each buffer represents one instance cluster (step 3). The determination of a suitable resource requires a permanent assessment of the effectiveness of the current setting (step 4). The instances from one buffer are finally processed in a row (step 5). The core idea is that certain instances can be processed more efficiently if they are handled in a certain order (the application of a *processing rule*) by making use of optimizations such as reduced changeover times, caching or gaining of routine by human resources than randomly distributed instances.

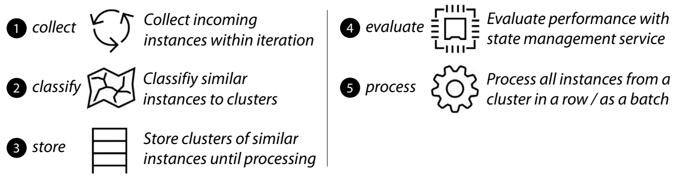
1 collect
Collect incoming instances within iteration
2 classify
Classifiy similar instances to clusters
3 store
Store clusters of similar instances until processing
4 evaluate
Evaluate performance with state management service
5 process
Process all instances from a cluster in a row / as a batch

Figure 1. Dynamic Instance Queuing (schematic description)

Performance gains achieved by DIQ in terms of time have been analyzed for different application scenarios, i.e., the medical domain (14% decrease in changeover time) [2] and the manufacturing domain (13% in throughput time) [3], both when compared to FIFO. This work investigates in how far the DIQ approach can be pushed to its limits in terms of further performance gain. For this, several optimizations are provided that result in the *extended DIQ* approach, i.e., by a) improving the logic of DIQ based on the so called *Triple Classification Approach* (TCA), b) specifying attributes that shall be adopted for classification, c) incorporating multiple resources to determine the most efficient one, and d) considering costs and quality of service in addition to time. Optimizations a) - d) are evaluated based on a real-world process from a Tier 1 technical support in a medium size organization.

Sect. II presents DIQ optimizations which are applied in a real-world scenario in Sect. III. Results and the evaluation are presented in Sect. IV. Sect. V and Sect. VI discuss future potentials and related work resp. Sect. VII closes with a summary.

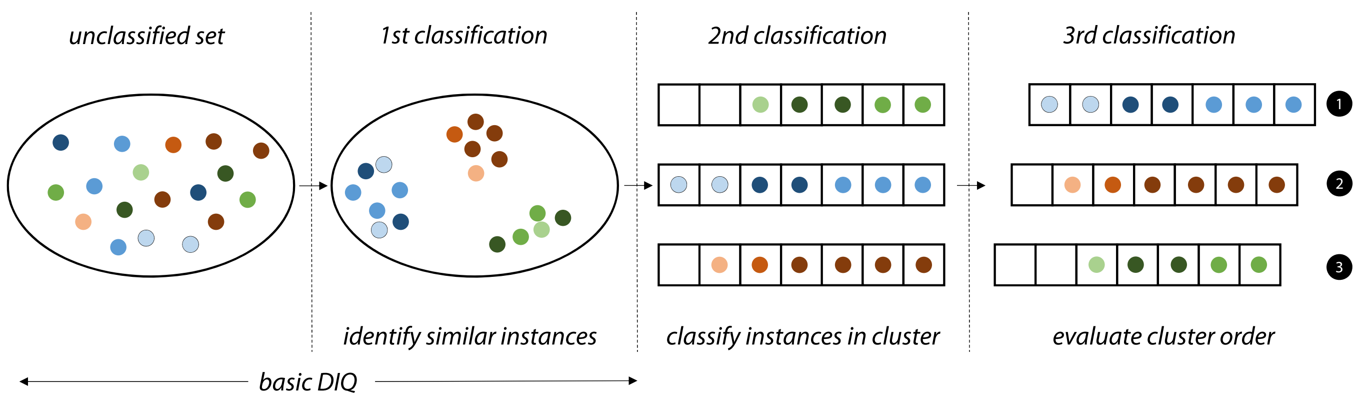

Figure 2. Triple Classification Approach (TCA)

II. DIQ Extensions and Optimizations

A. Triple Classification Approach (TCA)

While basic DIQ classifies instances once per activity based on the instances' attributes, TCA exploits further potentials by considering three different levels for determining an optimized execution order of instances, i.e., (1) the identification of similar instances, (2) their order inside the clusters and (3) the processing order of the clusters themselves (cf. Fig. 2).

1. Identification of similar instances. In the first step, *similar* instances are identified based on the instance attributes and grouped in the same cluster [2], [3]. Let I be the set of all instances and C be the set of all instance clusters. Then cl denotes the surjective function that maps each $i \in I$ onto a cluster $c \in C$, i.e., $cl : I \mapsto C$. There are two different classification techniques: The identification of similar instances can either be based on all attributes the instances possess (*generic approach*) or the attributes embraced for classification are specified for each task in advance (*specific approach*). The generic approach allows to apply DIQ without any a priori configuration, while the specific approach enables a processing optimized on the specific characteristics of each resource, which allows to set the classification attributes similar to the ones that trigger a processing rule and thus achieving a better performance. For any application scenario, the classification algorithm that fits the requirements best can be implemented, e.g. decision tree approaches, the k-Means clustering method, fuzzy means or Diana classification algorithm. Based on the performance evaluation from the state management service (cf. Fig. 1), the target cluster size and number can be adapted to the specifics of the available resources (if supported by the classification algorithm).

2. Determination of the optimum instance order inside the clusters. A second classification is executed to identify the instances with the highest similarity score for the same activity [5]. These instances are placed at the start of a cluster's queue, as similar instances offer the potential to reduce the processing time [2], [3]. This technique is called *shortest job first* (SJN) scheduling strategy. Schrage [6] proved that the SJN strategy minimizes the expected waiting time. It results in an optimum waiting time for all jobs [7] since it prevents the so called convoy effect [8]. The convoy effect is a phenomenon occurring in different disciplines meaning that some slow processes can slow down a whole system, leading to longer average waiting times and lower resource utilization [9]. This is particularly important for the application at hand, in which processing times have quadratic impact on the waiting times of the upcoming instances [10].

3. Optimum processing order of clusters. The instances of one cluster are processed in a batch, however, if more clusters than available resources exist, a processing order for the clusters should be defined. At this level, it is ensured that the mean average waiting times do not explode by considering the arrival times of the instances within the clusters: The processing order of clusters is evaluated based on the arithmetic mean cumulative difference to the respective instance's arrival times at the resource. By doing so, a certain sequential processing of instances at a high level is ensured, i.e., instances with a higher waiting time at the respective node are preferred compared to instances that arrived lately. If due times are defined, the arithmetic mean cumulative difference to the respective instance's due times is used instead.

B. Multiple Resource Support

Basic DIQ only supports scenarios in which exactly one resource is associated to one activity [2], [3]. In this paper, DIQ is extended towards the support of multiple resources, i.e., assigning several resources to one activity and vice versa. An evaluation by Russel et al. [11] shows that the support of resource patterns that offer work items to multiple resources is still limited in contemporary workflow engines. In our approach, offering work items to multiple resources is especially challenging, as clusters of instances are transferred to the resource's work list as a batch.

The mapping is applied by an extended resource broker which has access to all relevant performance data during runtime. It implements the decision function to determine the most suitable resource out of a list of available resources for an activity. We offer a reference function that incorporates both, the expected waiting time and the expected costs for processing in Formula 1.

$$r_{best} := min_{\forall r \in R_a} \{ \sum_{i \in I_{r,a}} (\frac{1}{\alpha} \cdot T(p_{i,r}) + \frac{1}{\beta} \cdot C(p_{i,r})) \} \quad (1)$$

In Formula 1, the most suitable resource r_{best} out of the set of resources R_a that are able to process activity a is evaluated. This means for any resource $r \in R_a$, the instances that are located in its waiting queue $I_{r,a}$ for activity a are considered. For any instance $i \in I_{r,a}$, both the expected processing time $T(p_{i,r})$ and the costs $C(p_{i,r})$ are considered: $T(p_{i,r})$ is evaluated based on the processing durations of previous instances processed by resource r for the same activity. Typically, information about the expected costs $C(p_{i,r})$ for processing instance i is known (or assumed) a priori, but a dynamic evaluation based on the costs for processing previous instances is an option as well.

The relative importance of the factor *time* and *costs* is individual for any application scenario. In Formula 1, α reflects an individual factor for the impact of the parameter *time*, while β represents the impact factor for the *costs*. That way, the relation between time and costs can be set; e.g. $\alpha = 10$ and $\beta = 1$ represents a setting in which $time$ is ten times more important than $costs$. Remember that $\alpha, \beta \geq 1$ must be fulfilled. In this decision function, lower values represent a better performance. Different decision functions can be implemented that fit the application scenario best.

III. Case Study and Simulation Setup

We applied DIQ including TCA and multiple resource support in a human process scenario based on a real-world dataset[1]. The process describes the operation of the Tier 1 technical support in a medium size organization. The employees of the IT service department provide basic support for users in terms of software administration and hardware replacement. The scope of the scenario is one day.

Fig. 3 provides the process model for the application scenario: Users or their responsible local IT personnel register certain requests in a ticket system which are then handled by the tech support. At first, new incidents are checked by ticket administrators for completeness of information. If important data is missing, certain skilled employees will contact the inquirer and complete the missing information. This potentially iterative process is represented by the loop pattern in the process model. Once the information is complete, tickets are being classified into administrative tickets and hardware incidents. Administrative issues are processed by software staff, while hardware incidents involve the work of both storage keepers to fetch the necessary items and hardware deployers to introduce them. When the work is finished, the corresponding ticket is being closed by a ticket administrator.

[1]Available at cs.univie.ac.at/project/wst-dc

Table I
Overview of roles, employees and tasks

employee	role	salary	task
Ryan Crowington	ticket admin	0.7	check ticket, close ticket
Melinda Jones	ticket admin	0.7	check ticket, close ticket
Jennifer Winston	sen. service admin	0.7	ask for more information
Rilla Sanders	team Leader	1.2	classify ticket
Faron Warrick	software admin	0.9	do administration work
Mark James	software admin	0.9	do administration work
Jack Osborne	storage keeper	0.6	get hardware from stock, deliver hardware
Jeff Bray	hardware deployer	0.7	set up hardware
Christian Horner	hardware deployer	0.6	set up hardware

In this scenario, support incidents represent process instances. Instances possess certain attributes that represent the characteristics of the associated support request: The name of the editor, a subject, an individual problem description provided by the editor, the id of the workstation (computer, mobile device) as well as the location of the workstation are available right at the start of the process. Further instance attributes, such as the ticket category, administration category, the affected hardware pieces and the location of the affected hardware in the storage are supplemented by the employees from the technical support within the execution of the process depending on the individual scenario.

Employees of the Tier 1 support represent *resources* that process the relevant tasks. A group of tasks is associated to a certain *role*. Most tasks can be processed by different employees. On the opposite, three members of the staff are able to perform different activities. A list of all existing roles, associated employees and relevant tasks is shown in Table I.

Preliminarily, the company has executed a process analysis to gain a better understanding of the occurring tasks, their dependencies and performance parameters. This process analysis also included the evaluation of estimated processing times for the tasks within the process. The analysis showed that the processing times depend on the order in which the work (resp. process instances) is done. For example, the task *ask for more information*, which is normally estimated to last 8 minutes, can be reduced to 2.5 minutes if the editor was the same as for the previous ticket. The reason is that information for both support incidents can be gained within one phone call, which prevents the redundant execution of subtasks such as searching the phone number of the editor, dialing or contacting an agent if the editor is not available. Two or more consecutive tickets from the same editor typically occur if local IT personnel reports incidents for the employees who then initialize several incidents as a batch, i.e. one ticket for each affected employee. Equivalent potentials were identified for other tasks: The software administration performs faster if tickets concerning the same software are processed in a

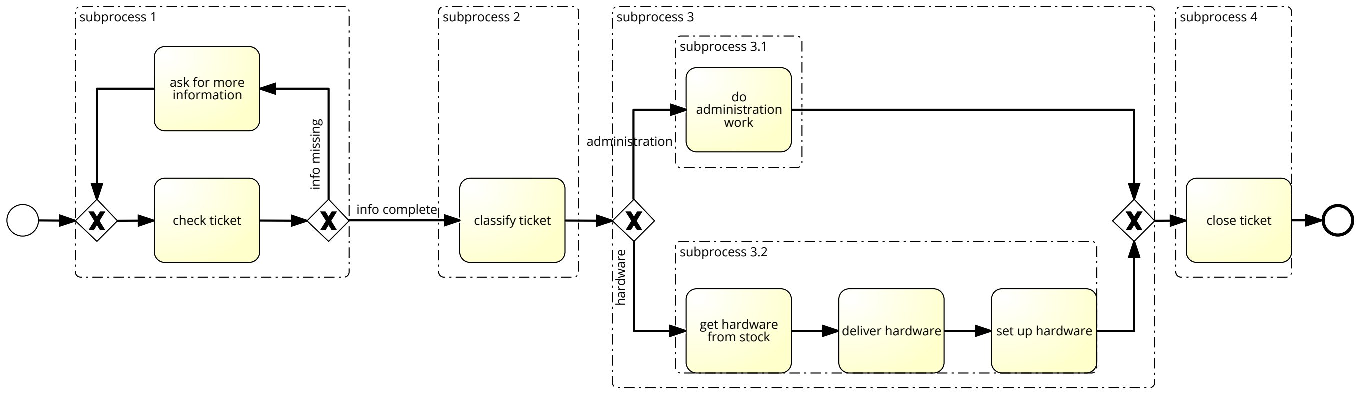

Figure 3. Tier 1 technical support process (produced using Signavio®)

Table II
OPTIMIZATION POTENTIALS FOR CERTAIN TASKS

task	pt	pt_{opt}	relevant attributes
ask for more information	8	2.5	editor, ticket id
do administration work	15	12	administration category
get hardware from stock	7	6	hardware location, hardware
deliver hardware	16	9	location of workstation

batch as the administration tool for the software application needs to be initialized only once and the delivery time of new hardware can be reduced if the location of the editor is the same (loss of redundant distance). All these potentials share the fact that they occur if a resource processes instances with certain attributes in a row (the triggering of a *processing rule*). *Reordering* instances means reordering tickets in the work lists from the employees. The applicable activities as well as the relevant attributes are marked in the process model (Fig. 3) and illustrated in Table II, where pt represents the expected processing time for regular processing and pt_{opt} the time if a *similar* instance according to the described attributes has been processed before. The processing time of the remaining tasks *check ticket*, *classify ticket*, *set up hardware* and *close ticket* does not depend on the processing order.

By introducing DIQ including the optimizations presented in Sect. II as well as by optimizing organizational processes, the following targets shall be achieved:

- *Time:* Reduction of the overall throughput time for the processing of support incidents in the Tier 1 tech Support. Time savings can be achieved primarily through computing an ideal order of work items in terms of reducing the processing time at critical activities.
- *Costs:* Reduction of the resource costs. Tasks shall be assigned to the tech support employees with the best relation of salary and estimated processing time (tech support is provided by external service providers).
- *Quality of service:* Avoidance of redundant calls as part of the *ask for more information* activity; reduction of multiple disturbance for users during hardware replacements.

The application of extended DIQ to the process scenario set out in this section was implemented and simulated based on lightweight application logic that strongly orients on the generic architecture of workflow engines described by [12], extended for the logic that is required for extended DIQ. This DIQ prototype consists of several components [3] including *workflow engine*, *rule provider*, *resource broker*, *state management service* and a *classifier*. At first, the engine processes an inbound queue containing completed work items. A work item is completed if the processing of the corresponding instance in its previous node of the process model has been finished. The rule provider evaluates the next activity of the instance based on the underlying process model. The resource broker then offers processing capacity for the evaluated activity. Based on information from the state management service, the resource broker ensures that the most efficient resource for the respective task is returned. In the last step, the instance is transferred to the work list of the resource, where the triple classification by the classifier takes place to determine an ideal work item order. Ultimately the work item will be processed by the resource associated to the corresponding work list. This architecture shows that DIQ is an extension (basically consisting of the classifier and the state management service) that can be easily added to existing workflow engines.

Three parameters are needed to execute the simulation of a process in a workflow engine, i.e., a) an *instance log* containing a list of all instances including trigger times and initial instance attributes; b) the *process model* holding the definition of all activities and their dependencies. The workflow engine used for this scenario expects a process model in BPMN-2 notation (BPMN XML 2.0); and c) the *resource definition* including costs and expected processing times. In the DIQ context, the resource definition contains information about instance attributes relevant for classification (*specific approach, cf. Sect. II-A*) as well.

For the scenario depicted in Fig. 3, the processing times have been derived from an existing analysis the company has executed before. We applied the simulation twice, once based on the extended DIQ approach (sim_{eDIQ}) as described

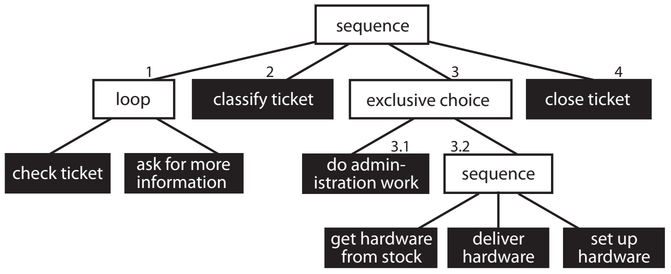

Figure 4. Process structure tree

in Sect. II and once based on the FIFO processing logic (sim_{FIFO}), which represents the *gold standard* (cf. Sect. IV). For sim_{FIFO}, instances are not reordered within the work lists of the resources. This means the *first in first out* (FIFO) processing strategy is applied, based on which process instances are processed in the same order as they arrived at the node of the process model. For sim_{eDIQ} the triple classification (TCA) was applied employing the k-Means algorithm for classification (setting: expected instance-cluster ratio: 3, minimum/maximum number: 2/5 clusters per classifications with a number of 50 iterations). A reduced processing time occurs if a resource processes two instances in a row that trigger a processing rule as described in Table II. This can happen in both cases, i.e., if similar instances are placed in a row by chance or if the classification results in an alignment (DIQ). Hence, sim_{FIFO} and sim_{eDIQ} are fully comparable.

IV. Results and Evaluation

To understand the simulation results in detail, in the following, the process model provided in Fig. 3 will be discussed along its subprocesses indicated by dotted lines. The subprocesses can be detected based on the corresponding process structure tree [13], represented by Fig. 4. The analysis focuses on Subprocesses 3.1, 3.2 and 1 which contain DIQ-relevant activities (cf. Table II).

In detail, Subprocess 1 represents the initial loop, including the activities *check ticket* and *ask for more information*. Subprocess 3 represents the exclusive choice, which splits in the first branch (to which we refer as Subprocess 3.1) containing the single-task *do administration work* and Subprocess 3.2 including the sequence of *get hardware*, *deliver hardware* and *set up hardware*.

The analysis starts with Subprocess 3.1 representing a single-task. Activity *do administration work* is executed by 74 workflow instances, i.e., 62% of all support tickets are of category *administration*. The task is processed by two employees. The first one handled 42 and the second one 32 support tickets. In Sect. III, we described that the estimated processing time for this activity is 12 instead of 15 minutes, if the previous ticket for the same resource had the same administration category. For sim_{eDIQ}, the reduced processing time, i.e., 12 minutes occurred 24 times. For sim_{FIFO}, a reduction to 12 occurred for 28 cases. This does not constitute a significant difference. The reason is that *do administration work* is not a critical activity – the two resources are not utilized for the whole time. In fact, a classification was enrolled only for 5 cases where more than one instance for the activity was in the resources' work lists. A low number of items to classify is a drawback as potentials from an optimized work order cannot substantially take effect [3].

Subprocess 3.2 represents a sequence of activities *get hardware from stock*, *deliver hardware*, and *set up hardware*, of which the first and second offer the potential for reduced processing times when certain instances with similar attributes are processed in a row. These attributes differ between the two activities. This circumstance is beneficial, as DIQ evaluates a separate optimized processing order for any of the two activities to adapt on the resources' specifications best: The average processing time for one instance for *get hardware from stock* is 17 seconds lower when making use of the DIQ optimization compared to the simulation run with DIQ disabled, for *deliver hardware*, the difference is even 112 seconds. For Subprocess 3.2, the total processing time over all instances is 34 hours (h) 19 minutes (m) for sim_{eDIQ} compared to 35h 28m for sim_{FIFO}, i.e., a reduction of 3.4%. The reduction of the processing time has quadratic effect on the waiting times of the upcoming instances [10]. This is reflected in the results of the simulation, where DIQ reduces the overall throughput time by 19.7%. The reason for the reduction in sim_{eDIQ} is the identification rate of *similar* instances: For *get hardware from stock* and *deliver hardware*, sim_{eDIQ} identified 36 similar instances, while sim_{FIFO} aligned 11 similar instances in a row. The theoretic probability for a processing rule to apply (compared to the number of work items processed) based on a random distribution of work items is 16.4%. The actual percentage of processing rules that applied for sim_{FIFO} was 12.2% and for sim_{eDIQ} 40.0%. sim_{FIFO} processed instances based on an initial order that we consider similar to a random distribution. In fact, the computed percentage (12.2%) differs only 4.2 points from the one theoretically evaluated (16.4%), i.e. the assumption that the processing logic without reordering of work items is random in terms of the instance order is valid for this scenario.

For Subprocess 1, *ask for more information* holds a processing rule: If tickets with the same editor are processed in a row, the processing of the second and sequential tickets takes less time. *Ask for more information* is executed 128 times for 77 different instances. The characteristic of loop patterns is the fact that positive (or negative) implications multiply with the number of executions [4]. Another effect occurs: As instances pass activities multiple times, not only the set of work items for the classification increases, but also the number of work items that are associated to the same instance attribute that triggers the processing rule (*editor*). For this reason, the classification is expected to work more efficient. The results of the simulation acknowledge this fact. For sim_{eDIQ}, the classification algorithm identifies 39 similar instances, while for sim_{FIFO}, only 5 instances are processed in a row. sim_{eDIQ} reduces the average throughput time by over 57 minutes resp. 35%.

Table III
RESULTS OF THE SIMULATION

Category	target	sim_{FIFO}	sim_{eDIQ}
Time	Avg. processing time / inst.	41m 31s	39m 28s
Time	Avg. throughput time / inst.	3h 59m 32s	3h 03m 34s
Costs	Utilization of cheaper resource	29.7%	56.2%
Costs	Due to busy time	4942.1 units	4698.1 units
QoS	Disturbance of users	reduced 5 times	reduced 39 times

Cost reduction potentials arise from both, the allocation of work to resources with a good performance-cost relation and from reduced busy times. For the deployment of hardware, two employees are available; Jeff Bray has a higher salary, but a lower expected processing time, while Christian Horner has the opposite characteristics. This makes *set up hardware* a typical *operations research* problem. For this scenario, one of the targets was to explicitly allocate work to resources with low costs. In fact, for sim_{eDIQ}, Christian Horner was busy 56.2% of his working time (sim_{FIFO}: 29.7%), while Jeff Bray was occupied 66.0% (sim_{FIFO}: 70.1%). Moreover, cost reductions arise from the coherence of processing times and expense. If one evaluates costs on a *costs per busy time unit* method (which is appropriate for external service providers), savings arise equivalent to the reduced processing times offered by the DIQ approach. This means for Subprocess 3.1, sim_{eDIQ} increased costs by 1.2%, while costs for Subprocess 3.2 decreased by 4.5% and costs for subprocess 1 decreased by 3.5%.

Quality of service is a highly individual target that differs between application scenarios. In this scenario, quality of service is considered a smooth running process of processing support tickets in a way that the users are only little affected by its execution. The tasks *ask for more information* is of special interest, as users often feel disturbed by phone calls from the service department. This disturbance can be reduced if questions concerning tickets from the same editor are combined. All in all, the 119 support tickets were created by 22 different editors. In 39 cases, sim_{eDIQ} identified that tickets are written by the same person and the task should be executed as a batch, i.e. user disturbance was minimized 39 times. For sim_{FIFO}, this happened only 5 times.

Evaluation of eDIQ over FIFO processing strategy: We consider sim_{FIFO} the reference processing strategy for processing workflow instances, as it computes items in the same order as they arrived at the resource (FIFO). This technique is very common, thus representing a *gold standard* to evaluate sim_{eDIQ} against. All simulation results are summarized in Tab. III: The expected probability for a processing rule for sim_{FIFO} (1.66%) was clearly increased by sim_{eDIQ} (12.0%), resulting in the application of 99 processing rules. For sim_{FIFO}, only 44 processing rules applied (5.35%). The temporal impact of sim_{eDIQ} on the average processing time per instance was a reduction by 2m 3s (5.2%). For the average throughout times, sim_{eDIQ} took 3h 3m 34s, while for sim_{FIFO}, the average throughput time per instance was 3h 59m 32s which represents a reduction of 23.4%. For 39 instances, the processing at the last activity of the process model was finished earlier for sim_{FIFO}, while 5 instances were finished at the same time for both settings. In contrast, 79 instances were finished faster when applying sim_{eDIQ}. This result is robust: Introducing a confidence level of one minute, i.e. counting only those instances that were at minimum one minute faster applying sim_{eDIQ}, this value drops only by one to 78 instances. Even with a confidence level of three minutes, 70 out of the 119 process instances are finished earlier for sim_{eDIQ}.

These time reductions also affect the costs, if they are evaluated based on busy times on employees. Further on, sim_{eDIQ} also realized the allocation of work to resources with lower costs. By doing so, the utilization rate of the cheaper resource increased by 26.5 points. Finally, the quality of service could at least be improved concerning the disturbance of users by phone calls asking for additional ticket information. sim_{eDIQ} decreased the number of duplicate calls 39 times. Overall, for the scenario at hand, sim_{eDIQ} offers a better performance concerning time, costs and quality of service compared to a non-optimized approach.

Evaluation of TCA: In order to assess the effects of the optimizations and extensions offered in Sect. II, i.e., the triple classification and the handling of multiple resources, a third simulation was conducted that "strips off" extension and optimization from sim_{eDIQ} resulting in the application of basic DIQ (sim_{DIQ}) as initially proposed and applied in [2], [3]. We analyzed the concrete effect by executing the simulation in a third setting with only *basic* DIQ applied (cf. Sect. II). Basic DIQ classifies instances only once (based on the instance attributes), while TCA optimizes the instance order within the cluster and the cluster order themselves.

Figure 5, left side shows the average throughput times for sim_{FIFO} (3h 59m 32s) and sim_{eDIQ} (3h 03m 24s). The application of basic DIQ sim_{DIQ} results in an average throughput time of 3h 17m 22s which is better than sim_{FIFO} but worse than the extended DIQ sim_{eDIQ}. Furthermore, sim_{eDIQ} decreases the global throughput time from 16h 56m 06s for sim_{DIQ} to 15h 32m 30s, i.e., TCA causes an optimization by 9.0% for this scenario. The global throughput time represents the duration from the processing start of the first instance until the processing end of the last instance throughout the whole process execution. Other time parameters demonstrate the positive impact of TCA as well: The average processing time for sim_{eDIQ} is 2.1% faster than for sim_{DIQ}; the average cumulated throughput time is optimized by even 7.5% (cf. Fig. 5). For sim_{eDIQ}, the overall costs are decreased by 8.9%. For *ask for more information*, sim_{DIQ} groups incidents from the same editor 23 times (out of 128 in total) which reduces the disturbance of users (Fig. 5). The advanced classification for sim_{eDIQ}, in contrast, identified 39 similar instances, which enables a higher degree of QoS.

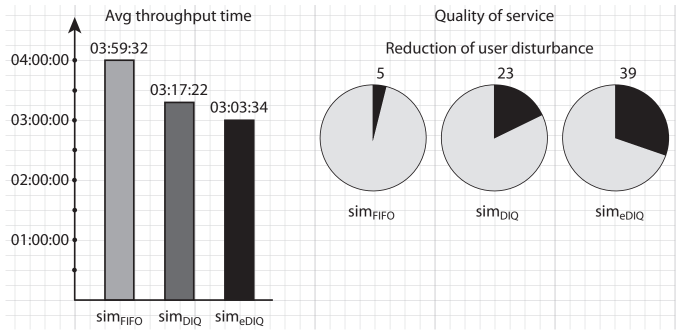

Figure 5. Temporal and QoS performance for sim_{FIFO}, sim_{DIQ}, and sim_{eDIQ}

The results show that TCA is a vital extension to basic DIQ which enables optimizations concerning times and costs. Although basic DIQ is a reasonable way to increase the workflow execution and was successfully implemented in certain scenarios ([2], [3]), it is strongly recommended to apply the new TCA extension for future applications.

Evaluation of Multiple Resource Support: There are four activities that can be processed by different employees: *Check ticket*, *close ticket*, *do administration work* and *set up hardware*. Some resources can perform different activities as well (cf. Table I). This requires an extension of DIQ to deal with multiple resources as presented in Sect. II-B

In the application scenario at hand, we applied the decision function as described in Sect. II-B to determine the most suitable resource. For the simulation, *time* and *costs* are equally weighted by $\alpha = \beta = 1$. Especially for the activity *set up hardware*, this means that a compromise between the expected processing time and the costs needs to be evaluated: *Jeff Bray* (24 min / 0.7 costs per time unit) has a better expected average processing time, while the working costs of *Christian Horner* (28 min / 0.6 costs per time unit) are lower. As costs and time are weighted equally (α=β), *Jeff Bray* offers a better overall performance. The decision function therefore assigns 73% of all tasks to *Jeff Bray*.

Contrary, for the two *ticket admins* both assigned to *check ticket* and *close ticket*, the expected processing times and costs are equal. The decision function evaluates a balanced distribution, where 56% of the instances are assigned to *Ryan Crowington* and the remaining 44% of instances to *Melinda Jones*. DIQ's resource broker obviously recognizes the different resources' characteristics and offers a mapping based on a function that balances the performance parameters.

Evaluation of DIQ in other scenarios: Previous versions of the DIQ technique have been applied in various application scenarios. A case study from the medical domain covered the printing management of specialized medical images [2]. In this one-activity scenario, the image processing printers represent a critical resource, which results in the arising of waiting queues. By reordering the instances (i.e. print jobs), changeover times declined by 14%, and costs could be reduced by 7%.

In a further scenario from the industrial domain, sequences of activities were optimized [3]. This case study covered the process of a retailer with a production line for construction materials. The process included six activities, which is especially beneficial to DIQ (cf. Sect. V): The throughput time was reduced by 13%; considering the processing end at the last activity of the process model, the instances from the DIQ run were processed in a total 9h 38m 38s faster than using the FIFO processing logic. In this scenario, DIQ was also evaluated against the theoretical, near-optimum scheduled process execution (which requires a posteriori knowledge, i.e. it can only computed after the process execution). DIQ turned out to be only 7% worse concerning throughput time than this optimum schedule. Although the early versions of DIQ implemented in these two scenarios were not that advanced as the eDIQ algorithm described in this work, these scenarios show that the principle of reordering process instances is a valid mean for optimizing the execution of business processes.

V. Discussion

DIQ is an approach to optimize the processing strategy of workflows during runtime. It can be applied either in addition to conventional optimization strategies or instead of them in order to achieve a certain degree of improvement with little effort. In the following, we will analyze the potentials of the DIQ approach: We will discuss (1) the context in which the application of DIQ is particularly beneficiary, (2) the requirements the applications scenarios must fulfill, (3) beneficial parameters, and (4) the expected performance of DIQ.

Context: A business process implemented within a PAIS might include automated activities, human interaction, or a combination of both. One of the following situations might exist: (a) There is a lack of time or money to conduct a comprehensive process analysis and redesign as described by state-of-the-art optimization approaches (cf. Sect. VI), (b) the process is too complex to evaluate a proper processing logic or there is a lack of knowledge to do so, (c) the process execution is too dynamic in terms of quantity of instances or resource behavior to foresee an ideal processing strategy or (d) optimizations shall be achieved on very short notice. These factors often occur conjointly in complex service or manufacturing scenarios with a high degree of collaborative human involvement.

Requirements: The process instances are distinguishable based on their attributes, e.g. instance id, and strict sequential processing in terms of a first-in-first-out processing order is not a requirement. Further on, at least for one resource of the workflow, the processing performance depends on the instance order, i.e., at least one processing rule exists. This is the case for a lot of scenarios, e.g. through gaining routine by humans or reduced changeover times at machines.

Parameters: Certain parameters improve the result of DIQ: The occurrence of critical activities is an asset. We refer to an activity as being critical if due to restricted resources assigned to the activity, a significant number of process instances cannot be processed momentarily after arrival at the resource associated to the activity continuously throughout the workflow execution [2]. The latter constraint acknowledges the fact that waiting queues might occur as a result of temporary inhomogeneous instance trigger times. Such sporadic events do not constitute a sufficient requirement for a critical activity on their own. A critical activity is characterized by a continuous imbalanced ratio of processing ability and instances to be processed. Concerning the process model, a large number of sequences with several orchestrated activities are a plus, while the occurrence of parallelizations is negative to the performance of DIQ. Finally, a large number of resources with associated processing rules are a plus. Concerning the process instances, the characteristics of the instance attributes are important to the quality of the classification. Numeric attributes are an asset, as classification techniques are mature for numbers. Non-numeric attributes are harder to handle, as semantic clustering techniques are difficult to apply and quality of outcome is ambivalent.

Potentials: DIQ will most probably offer a better performance than a non-optimized processing approach - in any case, the expected performance is not worse than a typical first-in-first-out processing strategy [2]. However, DIQ will not accomplish performance gains in an extent as provided by optimization techniques that are tailored on the specific application scenario. This includes business process redesign techniques and operations research approaches (cf. Sect. VI), which, however, both require expert knowledge and/or comprehensive input data. In fact, this is not the context DIQ is made for: DIQ represents a heuristic approach that achieves performance gains in a certain extent without or with little a priori work necessary. DIQ might also be considered as a first step towards a comprehensive process analysis. Flexibility is an asset of DIQ, as the approach is independent from the concrete application scenario and therefore covers changes in the process or resource model.

VI. Related Work

Until today, a structured and repeatable methodology that could be generally applied to business process improvement was not established [14]. Differences among the approaches concern the required parameters, the goals pursued and the scenarios they can be applied in. The optimization approach described in this work, e.g., aims at improving the processing strategy in a simple and lightweight way, which is ideal for scenarios with the need for an improved processing strategy on short notice and/or a lack of time, money or knowledge to apply a comprehensive process redesign. Alternative optimization approaches are described in the following.

Business process optimization approaches including business process redesign and/or reengineering efforts [15], [16], [17] promise exceptional results [14]. However, significant performance improvements will require fundamental changes in the process model [18]. Moreover, existing approaches remain at a rather abstract level in describing redesign phases [14], leaving the *core* reengineering work to the process designer's intuition [19]. DIQ operates automatically at runtime and hence does not require any redesign efforts.

Expert systems offer decision-making ability based on a reasoning and knowledge base mechanism [20]. The quality of the defined rule system hence determines the efficiency of the process. Expert systems are suitable especially for scenarios in which the stakeholders incorporate a comprehensive knowledge about the process and are willing to invest resources to develop such a system. DIQ, in contrast, is designed for process optimizations with little investment.

Scheduling and operations research approaches refer to allocating instances on resources in an ideal temporal manner (overview provided in e.g., [21]). Especially mixed integer linear programming (MILP) has become one of the most widely explored methods to solve scheduling problems [14]. Formal mathematical [22] and fuzzy approaches [23] are common as well. Generally, there is a great diversity of application scenarios in which scheduling approaches can be applied. Especially in the domain of logistics, scheduling is applied wide spread. Optimization capabilities are targeted at a specific application area and cannot be easily transferred toward another discipline [24]. Moreover, many scheduling techniques can only be applied for constraints and objectives defined in a formal manner [24]. DIQ deals with the allocation of instances on resources as well. Contrary to scheduling approaches in which the keeping of due times is the primary target, DIQ aims to achieve a compromise between kept due times and global temporal parameters. Furthermore, operations research and algorithmic approaches require an extensive amount of data, e.g. information about due dates, resource capacities or the instance distribution [19], while DIQ requires only litte a priori definition (in terms of relevant instance attributes for optimization) and no input data.

Evolutionary computing offers the potential to adapt to the specific circumstances of the application scenario constantly. While evolutionary computing has been successfully implemented in the context of scheduling [25], application for process optimization in general is rather limited so far [14]. Evolutionary approaches and DIQ share the characteristic that they are applied during runtime. In our future work, we will introduce elements of evolutionary computing in DIQ as well by implementing supervised learning techniques for computing appropriate clusters of instances.

Queuing and batching approaches have been proposed for process optimization [26] and as a strategy to prevent delays and deadline violations [27]. For these approaches, even if queuing and batching become effective at runtime, the underlying rules and strategies are still static, i.e., fixed at design time. Hence, the specifics of the current situation in the system known by the Process-Aware Information System are not taken into consideration. In contrast, DIQ operates at runtime with no a priori definition needed.

VII. Summary

Business process optimization provides a major competitive edge. However, financial and temporal restrictions often prevent a comprehensive process analysis and redesign that is required for most of the common optimization approaches. DIQ is a technique that allows ad-hoc improvements concerning time, costs, quality of service and flexibility. It fills the gap between a non-optimized process execution and processes that are optimized by common techniques. We implemented an extended version of the DIQ approach including a multi-level classification (TCA) with multiple resource support in a real-world scenario to meet short-term management targets. We learned that the total throughput time of the process could be reduced by 19.7% and implemented a higher standard quality of service. Furthermore, the service costs for the relevant task decreased by 4.5%. This could be achieved with only little a priori investment, which makes the extended DIQ approach an easy way to optimize the resource management during process executions in various application domains.

Acknowledgment

This work has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-072.

References

[1] Gartner, "Transforming business through strategic process management," Gartner Press Release, Egham, UK, June 2016.

[2] J. Pflug and S. Rinderle-Ma, "Dynamic instance queuing in process-aware information systems," in *Proc. 28th Annual ACM Symposium on Applied Computing (SAC '13)*, 2013, pp. 1426–1433.

[3] J. Pflug and S. Rinderle-Ma, "Application of dynamic instance queuing to activity sequences incooperative business process scenarios," *International Journal of Cooperative Information Systems*, vol. 25, no. 1, p. 1650002, 2016.

[4] J. Pflug and S. Rinderle-Ma, "Analyzing the effects of reordering work list items for selected control flow patterns," in *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, 2015, pp. 14–23.

[5] J. Pflug and S. Rinderle-Ma, "Process instance similarity: Potentials, metrics, applications," in *International Conference on Cooperative Information Systems (CoopIS) 2016*, September 2016.

[6] L. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, pp. 687–690, 1986.

[7] M. Sindhu, R. Rajkamal, and P. Vigneshwaran, "An optimum multilevel cpu scheduling algorithm," in *International Conference On Advances in Computer Engineering*, 2010, pp. 90–94.

[8] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts, 7th edition*. John Wiley & Sons, 2005.

[9] R.-S. Chang, J.-S. Chang, and P.-S. Lin, "An ant algorithm for balanced job scheduling in grids," *Future Generation Computer Systems*, vol. 25, no. 1, pp. 20 – 27, 2009.

[10] N. Slack, *Operations and process management: principles and practice for strategic impact*. Harlow, England: Prentice Hall/Financial Times, 2009.

[11] N. Russell, W. van der Aalst, A. H. ter Hofstede, and D. Edmond, "Workflow resource patterns: Identification, representation and tool support," in *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[12] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*. Berlin: Springer, 2004.

[13] J. Vanhatalo, H. Völzer, and J. Koehler, "The refined process structure tree," in *Int'l Conf. on Business Process Management*, 2008, pp. 100–115.

[14] K. Vergidis, A. Tiwari, and B. Majeed, "Business process analysis and optimization: Beyond reengineering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 1, pp. 69–82, Jan. 2008.

[15] J. A. Palma-Mendoza, K. Neailey, and R. Roy, "Business process redesign methodology to support supply chain integration," *Journal of IM*, vol. 34, no. 2, pp. 167 – 176, 2014.

[16] T. Jaeger, A. Prakash, and M. Ishikawa, "A framework for automatic improvement of workflows," in *Sixth International Conference Tools with Artificial Intelligence*, 1994, pp. 640–646.

[17] T. A. Aldowaisan and L. K. Gaafar, "Business process reengineering: an approach for process mapping," *Omega*, vol. 27, no. 5, pp. 515 – 524, 1999.

[18] A. Gunasekaran and B. Kobu, "Modelling and analysis of business process reengineering," *International Journal of Production Research*, vol. 40, no. 11, pp. 2521–2546, 2002.

[19] I. Hofacker and R. Vetschera, "Algorithmical approaches to business process design," *Computers & OR*, vol. 28, no. 13, pp. 1253–1275, 2001.

[20] C. Tasso and G. Guida, *Topics in expert system design: methodologies and tools*. Elsevier, 2014, vol. 1.

[21] C. A. Méndez, J. Cerdá, I. E. Grossmann, I. Harjunkoski, and M. Fahl, "State-of-the-art review of optimization methods for short-term scheduling of batch processes," *Computers & Chemical Engineering*, vol. 30, no. 6–7, pp. 913 – 946, 2006.

[22] C. A. Floudas and X. Lin, "Mixed integer linear programming in process scheduling," *Annals of Operations Research*, vol. 139, no. 1, pp. 131–162, 2005.

[23] H. Rommelfanger, "The advantages of fuzzy optimization models in practical use," *Fuzzy Optimization and Decision Making*, vol. 3, no. 4, pp. 295–309, 2004.

[24] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering," *European Journal of Operational Research*, vol. 153, no. 1, pp. 3 – 27, 2004.

[25] E. Hart, P. Ross, and D. Corne, "Evolutionary scheduling: A review," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 191–220, 2005.

[26] J. Liu and J. Hu, "Dynamic batch processing in workflows: Model and implementation," *Future Generation Computer Systems*, vol. 23, no. 3, pp. 338 – 347, 2007.

[27] W. van der Aalst, M. Rosemann, and M. Dumas, "Deadline-based escalation in process-aware information systems," *Decision Support Syst.*, vol. 43, no. 2, pp. 492–511, 2007.