Supporting Augmented Reality Industry 4.0 Processes with Context-aware Processing and Situational Knowledge

Gregor Grambow, Daniel Hieber, Roy Oberhauser and Camil Pogolski

Dept. of Computer Science
Aalen University
Aalen, Germany
e-mail: {gregor.grambow, daniel.hieber, roy.oberhauser, camil.pogolski}@hs-aalen.de

***Abstract*—Production processes in Industry 4.0 settings are usually highly automated. However, many complicated tasks, such as machine maintenance, must be executed by human workers. In current smart factories, such tasks can be supported by Augmented Reality (AR) devices. These AR tasks rely on high numbers of contextual factors like live data from machines or work safety conditions and are mostly not well integrated into the global production process. This can lead to various problems like suboptimal task assignment, over-exposure of workers to hazards like noise or heat, or delays in the production process. Current Business Process Management (BPM) Systems (BPMS) are not capable of readily taking such factors into account. Therefore, this contribution proposes a novel approach for context-integrated modeling and execution of processes with AR tasks. Our practical evaluations show that our AR Process Framework can be easily integrated with prevalent BPMS. Furthermore, we have created a comprehensive simulation scenario and our findings suggest that the application of this system can lead to various benefits, like better quality of AR task execution and cost savings regarding the overall Industry 4.0 processes.**

***Keywords*—Business Process Management Systems; Augmented Reality; Fuzzy Logic; Business Process Modeling Notation; Resource Assignment Automation.**

I. INTRODUCTION

Today's manufacturing industry heavily relies on smart factories which enable a better customer orientation as well as more efficient and individual production. In this context, the term "Industry 4.0" is used for the fourth industrial revolution driven by digitalization. Machines with built-in sensors and information technology form Cyber-Physical Systems (CPS) [1], enabling comprehensive optimization of production with regard to criteria, such as costs, resource consumption, quality or availability. Despite the focus on a high automation level and autonomous systems, human involvement in complex processes still plays a crucial part. Human workers often have to make important decisions or perform complex tasks, like machine maintenance.

In modern smart factories, such human tasks are often supported with AR devices. With AR, the worker can receive additional information for a task, contributing to its quality, performance, and repeatability. However, the integration of such activities in the global production process remains a challenge. A primary reason for this is that human AR tasks depend on a myriad of factors that are not represented in the global process. This includes the following factors:

- The AR tasks rely on different contextual data sets, e.g., external information sources supporting task execution, such as maintenance manuals, alternative procedures, checklist variability, live data from external systems or sensors of machines, the task executor and their decisions, and context-sensitive AR data like the relative position of the worker or the machine.
- For maximal effectivity and efficiency, the task must be assigned to the best suitable worker. Simple Staff Assignment Rules (SARs) of contemporary BPMS governing the production processes are only capable of determining if an agent is able to perform a task, but not their level of suitability. For AR tasks in complex Industry 4.0 settings, however, many parameters should be taken into account, like the position of the worker and the task, the qualification of the worker, or the workload of each worker. Otherwise, task execution might be suboptimal or too expensive, e.g., because of overqualification of the worker or long distances between him and the task. Furthermore, work safety is usually enforced by legal regulations and workers' exposure to hazards like heat and noise must be strictly limited.
- Usually, workers processing AR tasks are able to communicate via the AR device. However, as the AR tasks are not integrated with the global process, decisions or information provided by the worker cannot be used in that process, leading to delays or incorrect activity choices.

Contemporary BPMS lack facilities for representing and exploiting such data sets and contextual factors. Usually, these systems utilize standard BPM languages like the Business Process Modeling Notation (BPMN) [2], which were not designed to integrate such information into the process templates. Subsequently, live data and situational knowledge cannot be utilized in the process instances based on these templates.

In prior work [3][4][5], we developed an approach for contextual process management. However, our prior approach was tailored towards software engineering processes and did not involve the complex specifics of Industry 4.0 nor AR processes. To overcome the aforementioned limitations, this contribution proposes an integrated framework extending current BPMS with the following features:

1) Facilities to model processes that incorporate contextual

 ISBN: 978-1-61208-874-7

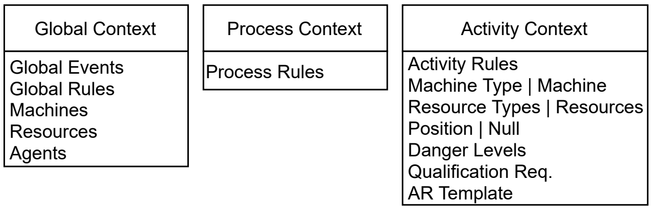

Figure 1. Context data model.

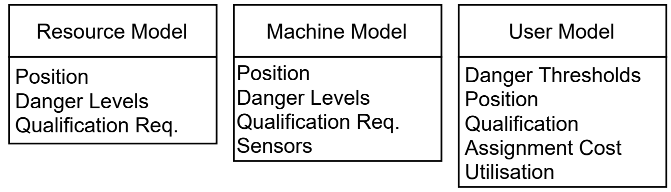

Figure 2. Actor models.

factors crucial for human AR tasks.

2) Incorporating real-time context data in BPM-Processes, enabling context dependent decision making and execution support.
3) An AR activity interface that can be used in such processes, enabling bi-directional communication between the process and the AR-supported worker.
4) An intelligent task assignment component capable of utilizing contextual data for fine-grained suitability levels to be able to match the optimal worker for any task.
5) Easy integration into existing BPMS.

The remainder of this paper is structured as follows: Section II describes the concept and solution approach, while Section III provides realization details. Thereafter, Section IV evaluates the feasibility and efficiency of the implemented system. Section V elaborates the background of the research as well as related work. Finally, Section VI provides a conclusion and outlook on future work.

II. Solution Approach

This section describes our concept for a context-aware system with AR support, called AR-Process (ARP) Framework (ARPF). It is conceived as a generic extension that any BPMS can readily integrate, providing facilities for representing contextual and AR information in executable processes in conjunction with an enactment component.

A. Contextual Processes

To enable the application of the ARPF in both new and existing processes and enable easy integration in any BPMS, the contextual information will be integrated into the processes via a generic BPMN 2.0 extension. Extending the BPMN standard not only allows an easy integration (requirement 2), but also allows the reuse of the existing BPMN service and script activities [6], heavily reducing implementation efforts. Such activities provide an intuitive interface between the BPMS and the ARPF. With this approach it is possible to decouple the ARPF from the process itself and provide it as a service to any BPMS supporting BPMN 2.0.

In the following, we will elaborate on the context data and rules or conditions crucial for contextual ARP execution. The context is separated into three major parts: global, process, and activity. A model can be seen in Figure 1.

The *global context* represents a cross-process entity containing all required global information. This includes information about different entities external to the BPMS. In particular, all machines, resources, and available agents. Further, the global context should provide facilities for defining conditions regarding the context that must be verified and fulfilled before an activity can be executed (requirement 2), e.g., check if an agent has the required danger clearance for an activity. This is realized by a *global rule set*. Another important factor is external context information that must be provided to the ARPF, e.g., priority changes for customer orders. Such data is incorporated via a global event system. in this manner, real-time context integration into BPM processes on a global level (requirement 2) can be achieved.

In addition to such global information, each process type may also have specific contextual conditions, e.g., if a specific process should only access a subset of the available machinery. To achieve this, a *process context* is employed that can overlay applicable portions of the global context if required. It further contains an additional *process rule set*. The latter is similar to the global rule set, but is limited to the processes of this type.

The third important entities requiring contextual information are concrete activities. To support these, an *activity context* is defined. It contains specific information for a single activity of a specific activity type and can be further specified during the ARP execution. As for the process and global context, an *activity rule set* is present to enable fine grained conditions on the activity level. On this level, however, a set of additional contextual information is required to enable an efficient assignment of the best suitable worker for each task. This incorporates data like the danger levels the task may involve, the qualification to successfully complete it (both defined as a dynamic set of key value pairs, containing values between 0 to 1), and the position of the activity represented by a three-dimensional vector X, Y, Z. Machine types can also be defined, as well as additional resources required for the activity, allowing the inclusion of machine context data directly into the process. Finally, the information, which AR-Component should be displayed to the worker while executing the activity (requirement 3) must be present. This is achieved by the AR Template.

B. Data Models

In addition to the contextual information added to the processes that governs how activities should be executed efficiently, the ARPF also requires information about the physical entities involved in process execution. In particular, three entities are crucial: the workers, the machines where activities (e.g., a maintenance task) are executed and their position, and resources required for such activities, like materials or tools.

 ISBN: 978-1-61208-874-7

To provide such information to the ARPF, three models are created, which can be found in Figure 2.

As simple BPM engines do not provide entries for resources and machines, new models must be created. Both contain a position, connected danger levels (e.g., noise with machines or chemical hazards with resources), and the required qualification to safely and efficiently work with the machine/resource. Machines usually also contain sensors providing real-time information about important production parameters. These are also included in the model.

Finally, the BPMS user/agent concept has to be extended, as current BPMS lack sufficient information to support AR activities intelligently. In order to assign agents to activities, the BPMS must possess compatible models. This can be achieved by extending the agent model of a BPM engine with values for position, qualification, and danger thresholds. Further, in most cases, cost-effective activity execution is also a requirement. Therefore, we incorporate information about the additional cost of an assignment of this agent (e.g., salary of an external worker, or weekend surcharge if it is not part of the contract) and the current utilization of the worker to avoid unbalanced workloads.

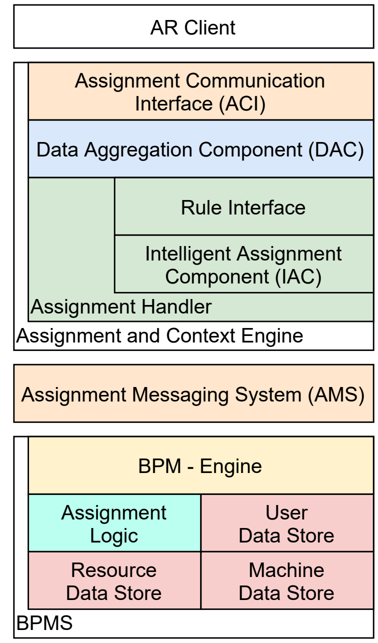

Figure 3. ARPF concept architecture.

C. Process Modeling

In current BPMS, there are rather limited and generic facilities to add context data to processes. This concerns the process modeling tools as well as the processes themselves. To overcome these limitations, our approach for adding contextual data for ARP execution is to realize such datasets as an extension for the most prevalent process language currently, BPMN 2.0. That way, the integration in a BPMS can be readily achieved, as any BPMN 2.0 compatible modeling tool can be easily extended (cf. requirement 1). To show the feasibility of this approach, we provide a prototype implementation of our extensions into a prevalent BPM modeling tool.

With an extended modeling tool at hand, a process engineer can add all contextual information and dependencies crucial for ARP execution support to new as well as existing process models without programming knowledge. Users, machines, and additional resources can be specified, including relevant parameters like their position. With appropriate data structures in place, relevant live data (e.g., from machine sensors) can be incorporated in the processes as they are executed. This data, in turn, is utilized by the components of the ARPF to provide more efficient task assignments and more effective support of the AR activities in the process.

D. AR - Process Enactment

The core architecture of the ARPF for contextual ARP enactment is depicted in Figure 3.

The core component of ARP process enactment is the Assignment and Context Engine (ACE). To provide a generalized and independent solution, this component is decoupled from the utilized BPMS. This permits a finer engineering of the ACE independent of the utilized suite. To achieve this, the ARPF incorporates two language- and platform-neutral generic communication components. The Assignment Communication Interface (ACI) enables communication between the ACE and both the BPMS, as well as the client software on AR devices, while the Assignment Messaging System (AMS) manages live data from the ARPF environment. That way, the ACE can be realized independent of any preexisting programming language or BPMS limitations. This allows the usage of the ARPF with a wide range of existing BPMS (requirement 5).

Many BPMS are provided as a standalone BPM engine (e.g., Camunda [7], jBPM) and therefore require external software to build a fully functional BPMS capable of managing all crucial data sets for contextual process enactment. To overcome these limitations and provide an easy way to extend BPM engines, we provide three generic Data Stores (DS): a User-DS, a Resource-DS, and a Machine-DS. These contain additional context information as specified by the aforementioned data models, like a more refined user model, machines used in the factory, and resources required to complete tasks. This extended context data is required in the assignment process and during the activity execution in the AR-Client.

The Assignment Logic Component (ALC) of the BPM engine is used as a bridge between engine and ACE. It aggregates all required context data for assignments; however, it is also possible to integrate the assignment request completely in the process itself via service or script tasks defined in the BPMN 2.0 standard [6] (requirement 5).

If an assignment request is sent to the ACE via the communication interface, the request is forwarded to the Data Aggregation Component (DAC) and validated for completeness. If some required context data is missing, the DAC will request it from the corresponding DS. Afterwards, the assignment request is forwarded to the Assignment Handler. The handler can then calculate a specific assignment score for the requested

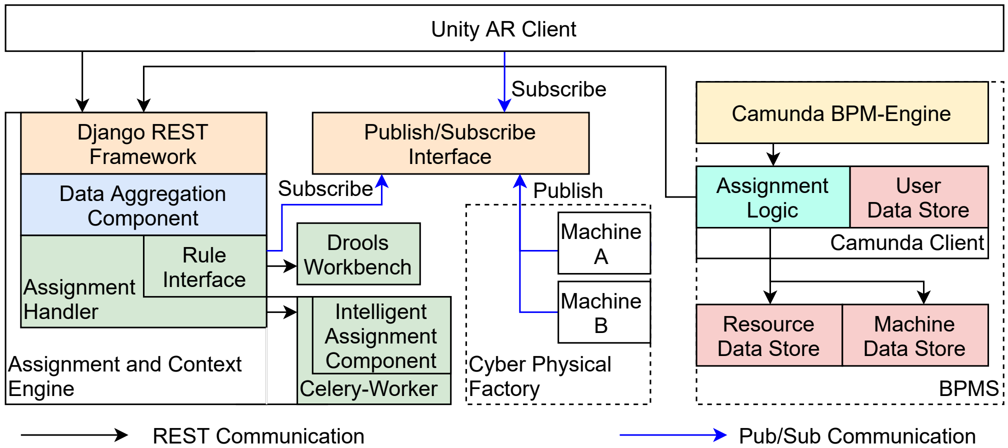

Figure 4. ARPF camunda implementation.

activity and agents in the Intelligent Assignment Component (requirement 4). If required, a presorting can be applied in a rule engine via the Rule Interface. Further preconditions of assignments (e.g., only assign the task if a sensor value is below a certain threshold) can be handled by the rule interface. To guarantee an optimal fine granular assignment score calculation fuzzy sets are utilized [8]. In this case, these are to be preferred to other solutions like Machine Learning (ML) and chaining. In contrast to ML approaches, with fuzzy sets no preexisting data sets are required nor is a training phase required, as weights can be defined directly and transparently according to the user's own knowledge and experience. Further, a fine granular calculated score between 0 and 1 is possible instead of the simple true or false of a chaining approach.

The final component of the ARPF is the AR-Client. The latter should be implemented as generic as possible to be available to a wide range of AR devices, e.g., tablets, goggles and even smartphones. The client is able to request all relevant process data via the ACI, and activities can be started, executed and completed in the AR-Client without the need to change to another software client e.g., a PC-interface, or web-client (requirement 3). Thanks to the provided AMS, it is further possible to consume real-time context changes on multiple levels (e.g., a global change of activity priorities or sensor data send from a machine connected to the activity being executed) (requirement 2).

III. Realization

This section describes the technical realization of the ARPF. It further details the communication between the components. While this section describes its integration with Camunda as a BPM engine and the AristaFlow BPM Suite [9] to demonstrate its capabilities with two mature and prevalent BPMS, the framework can be used with all BPM-Engines supporting REST-calls or external code execution.

Our prototype was implemented using Python. This approach was chosen for its rapid prototyping capabilities while still providing a large spectrum of libraries. As a base image for the ACE, a Django server was used which can be readily scaled for production deployment. To implement the ACI, the Django REST framework was integrated, providing a REST interface on top of the Django service. For the AMS, handling the real-time machine sensor communication, the Publish/Subscribe (Pub/Sub) system MQTT [10] was chosen, utilizing the Eclipse Mosquitto broker as the main component. As both technologies use well-defined industrial standards, an easy integration in BPMS is supported.

Figure 4 shows the architecture for the implementation of the ARPF with Camunda. Compared to the concept from Figure 3, some minor changes were made and the communication specified. The implementation of the AristaFlow BPM-Suite follows the same base architecture, however the full suite is provided by AristaFlow, removing the need for our own Data-Stores or an ALC. The communication is symbolized by colored arrows in Figure 4.

While the AristaFlow Suite does not require extensions, the Camunda solution requires implementation of a minimal BPM-Suite around the engine itself. This could either be realized as a single Java application relying on the Camunda Java API or using REST. In order to stay consistent with the general architecture, we use REST for our minimal BPM-Suite and split it into three sections. The Camunda BPM engine in its base version, a Camunda Client Django server containing the Assignment Logic, and the User-DS as well as a final Django Server containing the Resource and Machine Data Store. In order to connect the ARPF to a BPMN process template, it is required to create a service or script task sending a REST call to the Camunda Client. This call must contain the process instance id that can be acquired during process runtime in the same activity. During the process execution, the Camunda engine then calls the Assignment Logic via the created activity and triggers the assignment process. The Assignment Logic confirms the request to the engine and then spawns a new process handling the request. It then aggregates all data required for this assignment and sends an assignment request to ACE.

The Django REST Framework based ACI receives the assignment request and then executes the data aggregation component, validating that all required data for an assignment is available. In the Camunda implementation all required data is already present, in the AristaFlow implementation, the required data can be received from predefined endpoints. Afterwards the Assignment Handler is called. If preconditions are implemented (e.g., confirming the temperature of a machine sensor), the Rule Interface takes action. It first subscribes to all required machine sensor data endpoints via the Mosquitto Broker and then calls the connected rule engine via REST. In the implementation Drools is used for the Camunda Implementation while AristaFlow provides its own XPath-based solution. The preconditions can either be run in a loop (e.g., waiting for a sensor to cool down) until the condition is fulfilled, or in single-shot mode, aborting the assignment if the check is negative.

If the assignment is aborted, a response is sent to the Camunda Client/AristaFlow suite which are then required to provide a fallback plan, e.g., a retry after some time, a fallback process, or human intervention.

If the preconditions have been fulfilled, the assignment request is forwarded to the Intelligent Assignment Component (IAC), which itself is detached from the ACE to a celery worker. Utilizing the Celery Python framework, all assignment calculations are outsourced from the ACE and do not bind resources, therefore the I/O operations of the backend are not affected, even if many concurrent assignments are calculated. Each fuzzy assignment calculation is assigned its own processor for optimal execution speed. After the calculation is finished the IAC sends the assignment to the Celery Client/AristaFlow Suite handling the assignment update in the BPM engine.

The AR-Client is implemented using the Unity AR Foundation framework, this allows the creation of a generic AR frontend usable with a majority of present AR devices like AR goggles, tables, or phones. Instead of communication with the BPM-Suite itself, the AR-Client communicates via REST with the ACE and all requests to other sources are handled by the ACE. This enables the creation of a truly generic frontend independent of the BPMS, as all requests are parsed to the required model in the ACE. With this approach combined with a powerful AR interface, the user is able to complete and perform all activities in the AR-Client without the need to utilize another software solution or device. As the BPM workflow is still handled solely by the BPMS, it is however possible to switch at any moment to another solution (e.g., the Camunda Tasklist or the AristaFlow Client) if the worker deems it more beneficial, e.g., filling a long form.

While all process management communication is handled via REST between the AR-Client and ACE, the client can also access the Pub/Sub data via the Pub/Sub interface. It is therefore possible to see all relevant sensor values of a machine while working on it, or receive global updates (e.g., a change of priority or information a new assignment).

The final component of the ARPF is the Pub/Sub Interface, handling all MQTT messages. This contains all machines sensor data for the Rule Interface or the AR-Client, as well as global worker specific updates like a new assignment or priority updates. While the Camunda Client makes no use of global events via MQTT, the event feature is implemented in the AristaFlow suite.

In our prototype a Cyber Physical Factory is simulated using the OPC-UA protocol to connect machines' sensor data to the ARPF. As OPC-UA supports MQTT, this can be achieved in an easy and generic way, further easing the implementation into existing production environments.

To enable the creation of context-aware processes, a new BPMN modeler is created as an extension of the open source Camunda Modeler. While it is possible to create processes using the ARPF with any BPMN 2.0 modeler, a specific implementation comes with certain advantages. The modeler is linked to the different data stores and can therefore display all available machines, resources, and workers as specific entities or groups (e.g., CNC mill, maintenance workers, etc.) during the modeling of processes. This allows the process engineer to easily include the context during process creation. Further. it is possible to see available rules of the connected rule engine, enabling their integration as preconditions to activities. Moreover, the ARPF specific assignment request is moved to the background of activities, removing them from the eye of the user reducing the potential for user interface overload.

IV. Evaluation

The framework was evaluated using AnyLogic simulation software instead of a physical factory. This allowed for a more controlled and reproducible evaluation in a safe environment. Camunda was integrated as a BPM engine and Drools used as a rule engine.

The complete framework was deployed on a virtual server with 90GB main memory. However, the memory consumption never exceeded 24GB during our evaluation and can easily be halved by removing the Drools rule engine. The AnyLogic simulation was run on a Lenovo T495 with 14GB main memory utilizing Arch Linux as an operating system. To simulate values for the machine sensor, an OPC-UA server was hosted, utilizing a common industrial standard for this use case.

The evaluation was used to compare a BPMS using the ARPF against a plain BPM engine. To simulate workers and a realistic workflow, an AnyLogic simulation model was created and two simulation setups were configured.

As an environment, a factory with $21504m^2$ and a total of 29 machines requiring maintenance every 16 hours were created. The first maintenance was scheduled between 0 to 16 hours after start of the simulation. Further, the machines had an average breakdown interval of 36 hours. If a machine required maintenance or repair, a new Camunda process instance with the required worker qualification and the machine's position was started. The activity takes between 1 to 3 hours and requires an engineering qualification of 4 for maintenance and 6 for repairs. Other qualifications (electric, computer, bio_chemical) were not required and set to 0. As most modern manufacturing environments contain hazards requiring special training and regulations dangers were implemented in the simulation represented by values for noise: 0.01, heat: 0.03, electricity: 0.05, and chemicals: 0.02. While these values are quite abstract, they can easily be further refined and specified. A total of 5 workers (the agents in this use case) were available to complete these activities. Four internal workers, waiting in a maintenance building in the factory hall and one external worker, waiting 165 meters away. The external worker is used to display the need for highly trained personal which often has to be contracted by external service providers. The internal workers had engineering qualifications of 4, 5, 6 and 7 while the external worker had an engineering qualification of 8. The other qualification values were set to 0 to avoid bias. Their danger thresholds were set to 0.7 for all values. The usage of the external worker further was connected to an additional cost of 25000 (250€/activity), while the usage of internal workers incurred no additional costs. In their idle state, an worker checked every 5 minutes if a new activity was available. If they were working, they immediately checked after completion of their current activity for another enqueued activity. If no activity was enqueued, they switched back to the idle state and moved to their starting position. The simulation was separated into 5 work-shifts (each 8 hours long) with a break of 4 hours between shifts. During this break, workers were allowed to complete their current activity, but could not start new ones nor was it possible for machines to create a new task during the break. At the beginning of each work-shift, all tasks are reassigned and the danger thresholds of workers are reset to their default.

In the Camunda Setup (called CMD-Setup in the following), the workers fetched their activities directly from Camunda. All activities of the simulation were available to all of the workers and no further verification performed. If an activity is available to the group, the workers try to claim it and, if successful, work on it. In the ARPF Setup (further called ARP-Setup), the workers checked their personal worklist at the Assignment Engine REST API. If their personal worklist contains an activity, they start to work on it, otherwise they remained idle.

The five workdays were simulated for both configurations, using the same seed for the simulations random number generator. This process was repeated 10 times with different seeds to get statistically relevant test data. For the ARPF the model introduced in Section III was used. The qualification value was weighted half, to increase utilization of the more qualified workers and reduce the downtime of the machines. Further adjustment of the weighting could lead to heavily deviating results. An optimal weighting has to be configured according to the needs of the activities.

Table I shows a general comparison between the CMD and ARP simulation, while Table II shows a detailed comparison of internal and external worker stats in both simulations. In the following, values from Table I will be discussed and argued with the values from Table II.

TABLE I. ANYLOGIC ARP EVALUATION.

	ARP	Camunda
work_time	2103.31	2310.60
idle_time	524.49	396.38
avg_overqual	0.12	0.08
avg_tasks_day	3.52	3.62
violations	0.00	5.12
traveled_distance	9304.40	9502.27
cost	2000.00€	4600.00€
max_avg_underqual	0.00	-0.02
downtime_maintain	439.83	293.14
downtime_repair	218.90	249.32

The average work time and total activities per worker are lower in the ARP run, while the utilization of the internal workers (ARP-int) is slightly increased and the external utilization (ARP-ext) is heavily reduced. The average idle time is increased, which results from the low external utilization. The heavily reduced average cost of a simulation run, if using the ARPF instead of a plain BPM engine is due to the preferred use of internal workers. The increase in overqualification while using ARP instead of plain Camunda can be explained with the low weighting of qualification in the algorithms and no under-qualification, in opposition to the CMD-Setup, where under-qualification was generally present (to make it more realistic, under-qualified workers required 60 minutes longer than qualified workers). Taking a look at Table II, the main source of overqualification in the ARP simulation comes from the usage of the external worker, who was mainly used for activities below their qualification. This happened because the workload was too high and could be resolved by employing another internal worker with lower qualification to help out with these activities. This would lead to reduced costs and downtime. Optimization in the simulation or company values is needed rather than an adaptation of the algorithm.

The traveled distance for the internal workers is slightly increased in the ARP simulation compared to the CMD run. This correlates with the increased workload, and a stronger weight regarding the distance could reduce this effect. While the time for maintenance in the ARP run is around 40% higher than in the CMD-Setup, the actual down time for repairs could be reduced. This would increase the overall efficiency, as machines scheduled for maintenance still function properly while fast intervention is required on broken down machines. Further, the cost could be reduced to 43% of the CMD-Setup.

While the ARPF also utilized rules via Drools to validate if the work on the machine was safe by checking the values of the machine's temperature sensor against a max threshold, the base BPM engine did not provide such features. Violations against this precondition can be found under violations in Table II. In a real environment this would either lead to a safety regulation violation or would require a change of tasks for the worker, leading to even lower performance.

Finally, the ARPF could support workers more efficiently with their tasks, as it displays AR-instructions according to the qualification of the user. This could lead to a further speedup which has to be evaluated in a real-world setup.

Concluding, the ARPF worked as expected and the IAC

TABLE II. ANYLOGIC WORKER EVALUATION.

	ARP-int	ARP-ext	CMD-int	CMD-ext
work_time	2358.93	1080.83	2336.76	2205.95
idle_time	324.34	1325.07	381.82	454.65
avg_overqual	0.04	0.42	0.05	0.19
cost	0.00	200.00	0.00	460.00
avg_tasks_day	4.00	1.60	3.60	3.68
violations	0.00	0.00	4.88	6.10
traveled_distance	9952.32	6712.70	9386.80	9964.14
max_avg_ underqual	0.00	0.00	-0.02	0.00

produced comprehensible results. The utilization of ARPF in the simulations reduced the downtime of machines through failure and prevented any safety regulation violations.

V. RELATED WORK

Generalized context models are difficult to achieve and are not prevalent, as a survey on context models conclude [11]. An example is presented in [12]. The model is heavily tailored towards general pervasive computing scenarios and lacks several components crucial for industry 4.0 AR processes. In contrast, the ARPF context model presented is rather specific and yet readily extensible, due to its three layer context based on global, process, and activity context. Furthermore, the integration of context into process languages is challenging because they are not flexible enough, as stated in [13]. The contribution also proposes a BPMN extension for context integration, which is, in turn, tailored heavily towards mobile processes and not suitable for industry 4.0 production.

Focused on context processing the Java Context Aware Framework [14] is a technical object- and service-oriented framework targeting modeling context changes via rules. However, the processing of such rules is forwarded to the application layer. In JCOOLS [15], this limitation is overcome by by integrating JCAF with the Drools rule engine. The approach taken is rather complicated and generic, lacking support for both programmers and end users.

Examples of context modeling approaches include Coutaz and Crowley [16] and Ghiani, Manca, and Paternò [17]. However, these approaches primarily target the creation of context rules by the application developer that can later be completed with concrete values by end users, without providing the execution infrastructure.

There are also contextual approaches for Industry 4.0 production. Giustozzi et al. [18] provides a context model for industry 4.0 processes. Some of the mentioned entities are similar to the ones in the ARPF. However, the model is ontology-based and the paper primarily deals with logical relations of the concepts, which makes concrete implementation in an industry-ready system problematic. Furthermore, only a model is presented, lacking other components for integration process enactment. Another model for industry 4.0 production based on ML is presented in [19]. This model, however, is also not applicable for enactment of AR processes, as it primarily deals with predicting the degradation of the state of machines.

Another approach is taken by Tasdemir and Toklu [20]: it focusses on fuzzy task assignment and integrates BPM concepts. The described system is not suitable for the Industry 4.0 scenario, as it focuses on teams and the social relationships of the worker in the team. In addition, it lacks other components like a real-time data context model.

In summary, ARPF provides a unique approach for contextual processing for Industry 4.0 processes with human AR tasks, supporting integration with existing BPMS and utilizing a BPMN extension to include AR and context in new and existing process models. Other approaches lack the inclusion of information needed for representing processes and their connection to AR devices and workers, machines, and resources with their specific contextual properties and rules. In addition, most of these approaches do not present an integrated framework for comprehensively supporting process enactment in such complicated domains utilizing real-time data.

VI. CONCLUSION

This contribution described our ARPF approach for incorporating contextual factors crucial for AR tasks into industry 4.0 production processes. The presented framework incorporates components for integrating such factors when modeling the processes and utilizes live data from different sources while executing them. That way, contextually supported process enactment becomes possible, providing improved task assignment capabilities and better support for the AR activities. Furthermore, by providing bi-directional communication interfaces between the process and the AR task, the latter can be seamlessly integrated into the process.

We further implemented a prototype integrating our approach with two prevalent BPMS. The prototype shows that the integration with real BPMS is feasible and achievable with little effort. Further, we conducted an evaluation executing a comprehensive simulation scenario with our prototype. Our findings suggest that our approach can lead to various improvements for Industry 4.0 processes with AR tasks. Task assignments can be improved by incorporating contextual factors. Further, AR task execution can be better supported with matching contextual information. Thus, the overall process execution can be improved, resulting in better resource usage and cost savings. Moreover, other factors, like worker safety can also be taken into account and be seamlessly integrated into the processes.

Future work includes: the optimization of our context-integrated process editor to improve its appearance and usability; integration of ARPF with further BPMS; application of ARPF to other domains; further improvements to the BPMN 2.0 extension; and a comprehensive empirical evaluation in a real production environment.

ACKNOWLEDGMENTS

The authors wish to acknowledge Felix Gräber's contribution regarding the technical foundation of the IAC. This work was partially funded via the PARADIGMA project by ”Zentrales Innovationsprogramm Mittelstand” (i.e., the ”Central Innovation Programme for small and Medium-Sized

Enterprises (SMEs)"), of the Federal Ministry for Economic Affairs and Energy of the Federal Republic of Germany.

REFERENCES

[1] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.

[2] T. Allweyer, *BPMN 2.0: introduction to the standard for business process modeling*. BoD–Books on Demand, 2016.

[3] G. Grambow, R. Oberhauser, and M. Reichert, "Contextual injection of quality measures into software engineering processes," *Int'l Journal on Advances in Software*, vol. 4, no. 1&2, pp. 76–99, 2011.

[4] G. Grambow and R. Oberhauser, "Towards automated context-aware software quality management," in *2010 Fifth International Conference on Software Engineering Advances*. IEEE, 2010, pp. 347–352.

[5] G. Grambow, R. Oberhauser, and M. Reichert, "Towards automatic process-aware coordination in collaborative software engineering," in *6th Int'l Conference on Software and Data Technologies*. ScitePress, 2011, pp. 5–14.

[6] *Business Process Model and Notation(BPMN) - Version 2.0.2*. Object Management Group, 12 2013, retrieved: 2021.06.10. [Online]. Available: https://www.omg.org/spec/BPMN/2.0.2/PDF

[7] Camunda. Last Visited: 2021.06.10. [Online]. Available: https://camunda.com

[8] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, 1988.

[9] M. Reichert, "Enabling flexible and robust business process automation for the agile enterprise," in *The Essence of Software Engineering*. Springer, Cham, 2018, pp. 203–220.

[10] G. C. Hillar, *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017.

[11] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca, "A data-oriented survey of context models," *SIGMOD Rec.*, vol. 36, no. 4, p. 19–26, Dec. 2007.

[12] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Modeling context information in pervasive computing systems," in *Pervasive Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 167–180.

[13] J. Dörndorfer and C. Seel, "A meta model based extension of bpmn 2.0 for mobile context sensitive business processes and applications," *Proc WI 2017*, pp. 301–315, 2017.

[14] J. Bardram, "The java context awareness framework (jcaf) – a service infrastructure and programming framework for context-aware applications," vol. 3468, 05 2005, pp. 98–115.

[15] J. Park, H.-C. Lee, and M.-J. Lee, "Jcools: A toolkit for generating context-aware applications with jcaf and drools," *Journal of Systems Architecture*, vol. 59, no. 9, pp. 759–766, 2013.

[16] J. Coutaz and J. L. Crowley, "A first-person experience with end-user development for smart homes," *IEEE Pervasive Computing*, vol. 15, no. 2, pp. 26–39, 2016.

[17] G. Ghiani, M. Manca, and F. Paternò, "Authoring context-dependent cross-device user interfaces based on trigger/action rules," *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, pp. 313–322, 2015.

[18] F. Giustozzi, J. Saunier, and C. Zanni-Merk, "Context modeling for industry 4.0: an ontology-based proposal," *Procedia Computer Science*, vol. 126, pp. 675–684, 2018.

[19] J.-R. Ruiz-Sarmiento *et al.*, "A predictive model for the maintenance of industrial machinery in the context of industry 4.0," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103289, 2020.

[20] C. Tasdemir and C. Toklu, "Qos driven dynamic task assignment for bpm systems using fuzzy logic," in *Emerging trends and challenges in information technology management*. Hershey, Pennsylvania (701 E. Chocolate Avenue, Hershey, Pa., 17033, USA: IGI Global, 2006.