Utilizing Fuzzy Sets and Rule Engines for Intelligent Task Assignment in Industry 4.0 Production Processes

Gregor Grambow, Daniel Hieber and Roy Oberhauser

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

***Abstract*—Today's Industry 4.0 Smart Factories involve complicated and highly automated processes. Nevertheless, certain crucial activities such as machine maintenance remain that require human involvement. For such activities, many factors have to be taken into account, like worker safety or worker qualification. This adds to the complexity of selection and assignment of optimal human resources to the processes and overall coordination. Contemporary Business Process Management (BPM) Systems only provide limited facilities regarding activity resource assignment. To overcome these, this contribution proposes a BPM-integrated approach that applies fuzzy sets and rule processing for activity assignment. Our findings suggest that our approach has the potential for improved work distribution and cost savings for Industry 4.0 production processes. Furthermore, the scalability of the approach provides efficient performance even with a large number of concurrent activity assignment requests and can be applied to complex production scenarios with minimal effort.**

***Keywords*—Business Process Management Systems; Fuzzy Logic; Rule Engines; Resource Allocation Algorithms; Assignment Automation.**

I. Introduction

With this paper we extend our previous work [1], where we first introduced our intelligent assignment concept. We expound on the concrete workflow and implementation of the utilized fuzzy sets for task assignment in Industry 4.0 processes. Furthermore, we present a functional implementation of the Rule Interface, introduced in our previous work as a theoretical concept. Finally, we tie together our other work, successfully implementing the intelligent task assignment.

"Industry 4.0" stands for the fourth industrial revolution driven by digitalization [2]. Highly automated Smart Factories enable more efficient and individual production methods as well as greater customer focus. This includes the comprehensive control and organization of the entire production value chain by utilizing real-time data processing across all production stages. Cyber-Physical Systems (CPS) [3], which consist of information technology (IT), machines, and built-in sensors, form a unit that enables comprehensive optimization of production with regard to criteria such as costs, resource consumption, quality, or availability. While a strong focus on autonomous systems and the highest possible degree of automation exists, in highly complex processes human involvement remains indispensable. Often the production process depends on activities, in which people intervene, perform complex activities and make important decisions.

Such higher-level business and production processes are typically governed by Business Process Management Systems (BPMS) [4], also known as Process-Aware Information Systems (PAIS). BPMS are in charge of the sequencing of the different activities belonging to a business process including automated activities and those processed by human agents. The success of any BPM process realization can be endangered by excessive activity automation and poor design of work assignment strategies [5]. Therefore, assigning the optimal agent to an activity and vice-versa is a time-consuming but necessary task with every BPMS. In most BPMS, so-called Staff Assignment Rules (SARs) (or resource allocation) are utilized to achieve this. Yet this area has not received sufficient research attention, as indicated by the survey by Arias et al. [6] .

Moreover, in Industry 4.0 production scenarios, many different factors have to be taken into account to select an agent that can process an activity in an efficient and effective manner. An obvious example for such factors is the qualification of the agent, who must have the necessary skills and abilities to correctly execute the activity without being overqualified (and thus incurring unnecessary cost overhead). Usually agents with a much higher qualification level should not be assigned to a particular activity. Such optimizations should also consider balancing the agent workload to not overburden an agent while others are idle.

In large production facilities, the physical location of the agents and where the activities are to be performed also play an important role. An example are maintenance activities that have to be executed from time to time across a large number of production machines at a large facility. If not optimized properly, agents may waste a substantial amount of time in transit to activities, analogous to the well-known Traveling Salesman Problem [7]. Due to the high complexity of Smart Factories and their CPS, involving specialized external (maintenance) workers with specific knowledge to maintain a system can incur additional costs. To contain these costs, utilization of internal employees should be preferred if possible, depending on the urgency, availability, and qualification levels. In modern production, worker safety is also an important factor that is usually regulated by respective laws, which address hazards

such as chemical, electrical, heat, and noise and may not be adequately tracked by automation systems.

When taking such factors into account, it becomes evident that standard BPMS SARs are insufficient because they are only capable of determining if an agent is available to perform an activity, but cannot readily determine the degree of suitability. Fuzzy logic's [8] fine granular classification between 0 and 1 provides a way to overcome the limitations of simple Boolean logic and determine a specific assignment score for each agent for each possible assignment. Automating such a generic and recurring activity can optimize work efficiency and manpower cost, while reducing employee frustration when automated systems seem inflexible or make unsuitable assignments. By combining fuzzy logic with rule processing, prefiltering allows the fuzzy-specific areas to be factored from the rest of the more obvious rule-based resource allocation problem, utilizing the best of both fuzzy logic and rule engines.

In prior work [9][10][11], we also developed a different approach for contextual process management that did not rely on Fuzzy Sets but rather utilized an adaptive process management engine. software engineering processes and did not use fuzzy sets or involve the complex specifics of Industry 4.0 nor processes with integrated AR support. The main focus was extending processes with properties to enable automated software quality assurance and support collaboration of software engineers. This was realized via automatic process adaptations.

In this paper, we contribute an approach for activity assignment in Industry 4.0 projects that takes the aforementioned factors into account. By applying fuzzy sets and a rule engine, fine-grained levels of suitability are integrated to improve resource assignment results. To demonstrate its feasibility, we integrated our solution with a common BPMS. We further extended our previous version of the Intelligent Assignment Component with the integration of a rule engine, which was only theoretically discussed in our earlier work.

The remainder of this paper is structured as follows: Section II highlights related research and background information. Section III then describes the general concept and an initial solution approach, while Section IV details the concept for our Intelligent Assignment Component (IAC). In Section V, we provide specific implementation details focusing on the IAC while addressing the overall prototype. Then in Section VI we evaluate our solution. Finally, Section VII provides a conclusion and outlook on future upcoming work.

II. Related Work

In literature there are numerous approaches for activity assignment optimization utilizing different algorithms like fuzzy sets. Kubler et al. [12] provide a survey of the application of Fuzzy Logic in combination with Multiple Criteria Decision-Making, and within the category resource allocation, Shahhosseini and Sebt [13] is the only example of its application to human resource allocation. However, it is specific to construction companies, centers around four specific human roles and lacks an integration strategy with BPMS. Similarly, Kłosowski et al. [14] also discuss a fuzzy model for assigning workers to production activities. The main focus of their approach is employee assessment and a rich set of properties. However, for our use case, the model is too generic and contains unnecessary properties, while at the same time neglecting other important factors like worker safety or location. Furthermore, it also lacks BPMS integration concepts. Seifi et al. [15] apply fuzzy logic to optimize human resource allocation for project planning in small-to-medium sized organizations and does not consider live processes with a BPMS. In contrast, our work focuses on the Industry 4.0 production and is integrated with a BPMS.

Kluza and Nalepa [16] provide a formalized model combining a procedural business process model with Attribute Relation Diagrams for rules, and describe an algorithm that can generate an executable BPMN model with decision table schemas for rules in XTT2 representation. In contrast, our approach does not address automatic generation of models, and while supporting the simplicity of rules, our approach can address more complex problems with resource allocation by using fuzzy logic.

Antonelli and Bruno [17] deal with an Industry 4.0 topic: activity assignment in human robot collaboration. This approach splits the activity assignment problem into activity classification with a decision tree classifier and activity assignment with a decision-making algorithm. However, the approach does not address BPMS integration and relies on Boolean rather than fuzzy values, which makes it somewhat synthetic. In addition, worker safety is not taken into account.

Another approach for activity-resource assignment that applies fuzzy logic is presented by Xu et al. [18]. It contains a comprehensive but complicated fuzzy model targeted at collaborative logistics networks comprising logistics service integrators, activity contractors, and resource providers. Thus, the model cannot be used for the assignment of single workers in Industry 4.0 production. Finally, a category of approaches similar to Simpson and Roberts [19] utilize various algorithms like Bayesian methods, heuristic algorithms or game theoretic approaches for activity assignment in spatial crowdsourcing. As this domain has rather specific properties on which the algorithms rely, they also cannot readily be applied to Industry 4.0 production and for similar reasons, BPMS integration is not included in these approaches.

Approaches using rule-based resource allocation include iDispatcher [20], which focuses on the insurance domain using ILOG JRules. It does not address non-human allocations, context-awareness, or fuzzy problems. Li et al. [21] use the Drools rule engine and focus on human resource allocation for the IT service order rather than the Industrial Internet of Things (IIOT) domain. While mentioning the term workflow engine, it does not state which one nor explicitly address how it is practically integrated with a BPMS to do the assignment for an activity. Havur et al. [22] use Answer Set Programming with the clasp solver on timed Petri Nets using RDF Schema as a resource ontology for the railway engineering domain. In contrast, our work shows integration with commercial BPMS

and utilize the combination of a rule engine and fuzzy logic. Sikal et al. [23] utilizes process mining for resource variability discovery, but does not explicitly address rule modeling, fuzzy problems, nor the integration in a specific BPMS, and presumes this data is available a priori for analysis. Erasmus et al. [24] apply the Fleishman taxonomy for specifying activities, human resources, and their ability-based allocation during runtime for a manufacturing case study. The information from their method is stored in an SQL database and Java methods are used to make the assignments with the Camunda BPM. It does not address non-human allocations nor fuzzy problems. Vasilecas et al. [25] describe a rule- and context-based approach of dynamic business process modeling and simulation. It consists of a custom .NET-based implementation without BPMN compliance, is focused on simulation, and considers an ordering process.

Further non-fuzzy resource-allocation related work includes Ihde et al. [26] who describe a resource-aware extension framework to a traditional BPMS for process designers to specify resource allocation constraints, enabling external allocation services with different algorithms to process activities. They did not apply fuzzy logic nor did it involve a Industry 4.0 or production situation. Tan et al. [27] propose an optimal resource allocation strategy for cooperative task scheduling in cross-organizational business processes. It focuses on team formation, considering professional and cooperative ability, analyzing process event logs for an insurance claim business process. In contrast, our work extends an actual BPMS, uses fuzzy logic, and is applied in the Industry 4.0 domain.

III. Solution Approach

While different fuzzy-based approaches for activity assignment exist, they are often rather generic and complicated, or too specific and tailored to a certain domain. Moreover, they typically do not address integration with contemporary BPMS. To overcome these limitations and be able to create a usable system for Industry 4.0 scenarios, we focus on a more concrete model and a specific component executing the activity assignments while addressing integration with current BPMS.

To achieve suitable assignments in a practical and applicable manner, our approach addresses these requirements:

1) The system shall calculate an assignment score that reflects the suitability level of agents for handling a specific activity.
2) The runtime shall be capable of handling a large number of concurrent assignment scoring requests efficiently.
3) Integration into BPMS shall be readily feasible.

To maximize the efficiency of optimization options and support easy integration into various BPMS, a new system for handling assignments is created. By decoupling the assignment process from the BPMS, a separate component can be implemented solely for the assignment process, permitting better performance optimization without the constraints imposed if one were to internally extend a specific BPMS. Furthermore, this decoupling via a generic API supports a generic approach

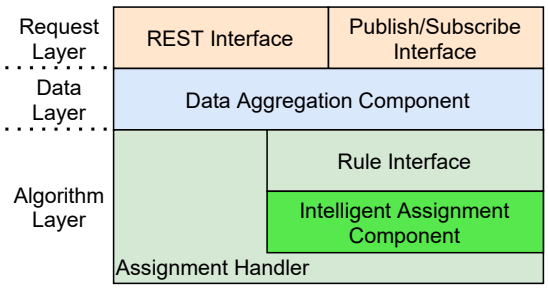

Figure 1. Assignment and context engine conceptual architecture.

that can support integration across a much wider range of BPMS. The conceptual architecture of the novel Assignment and Context Engine (ACE) providing such functionality can be seen in Figure 1.

The ACE uses a layer pattern, which is further subdivided into components, with each layer contributing to the final solution. Via the modular layers, if desired, the Data and Algorithm Layer could be directly integrated into BPMS (potentially enhancing performance). Alternatively, only the Assignment Handler or its individual components could be directly integrated in a BPMS (with a reduced set of features). Thus, we hitherto focus on the ACE as a holistic solution to fulfill the aforementioned requirements.

The public REST and Publish/Subscribe (Pub/Sub) Interfaces in the Request Layer are used as a BPMS- and programming-language-independent interface, allowing the usage of the ACE with any BPMS supporting BPMN 2.0 or later. The REST and Pub/Sub interface can be used used interchangeably as required by the concrete use-case, depending on the message volume and other factors. This standard offers a wide range of elements to integrate external services and functions [28] [29]. The integration is based on a dual activity concept. A utility-activity requests an assignment for the execution-activity succeeding it in the process workflow. For the utility-activity, two approaches in BPMN 2.0 are possible and should be chosen according to the capabilities of the applicable BPMS (Figure 2).

The synchronous variant utilizes a Service Task to request the assignment from the ACE synchronously. The service activity receives the required data from the process and then awaits the calculated assignment. Finally, it assigns the agent with the highest suitability level to the activity. The asynchronous variant utilizes a Script Task that obtains all the required values itself, accessing the BPMS and then requesting an assignment asynchronously. The BPMS can then ignore the process until the assignment is calculated and no resources have to await a response. As soon as the ACE finishes the calculation, it calls the BPMS API and assigns the best fitting agent to the activity itself. With this approach every state-of-the-art BPMS can easily be integrated (requirement 3) and one ACE could even support multiple BPMS at the same time.

Once a request is received by the backend, it is validated by the Data Aggregation Component (DAC). If all required information is present, the request is passed on to the Assignment

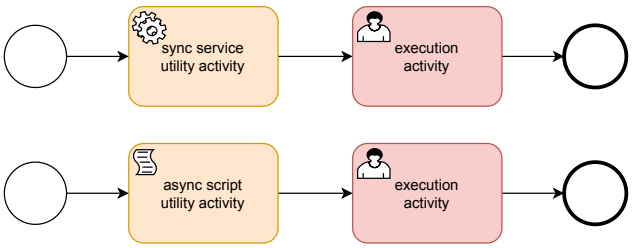

Figure 2. Activity solution variants (synchronous variant on top, asynchronous variant on bottom).

Handler. If some data is still missing the DAC can receive this via predefined external sources, e.g., a database containing the agents attributes (like position, or qualification) or directly via the BPMS API if endpoints to request additional information are provided.

The Assignment Handler is responsible for the overall assignment process, this is split into three parts:

- Checking preconditions
- For the requesting activity, determining the assignment score for each agent; and
- Triggering the assignment process in the BPMS

To reduce the overall load on the IAC, and to enable task filtering on external values (like sensor measurements or available inventory), rule-based methods are employed. By creating a Rule Interface as a facade, external rule engines can be readily connected to the ACE. These can utilize efficient algorithms to validate if preconditions are fulfilled or if something prevents the successful assignment and execution of the task. This rule engine integration can also be used to pre-filter agents, removing incapable agents from the actual assignment calculation. This can be advantageous when involving a high number of agents per task, as chaining algorithms are generally more efficient than fuzzy sets (requirement 2).

In order to provide the desired assignment score with a fine granular suitability level, fuzzy sets are chosen. As seen in Section II fuzzy approaches are able to generate very precise assignment scores (requirement 1) in an efficient way (requirement 2). This is an improvement over currently employed chaining-based SARs, which are capable of calculating accurate assignments, but lack the capability to differentiate between suitable agents, and thus do not provide overall optimal assignments. While a Machine Learning (ML) approach would also be feasible, the fuzzy sets provide some striking advantages. For fuzzy sets, no preexisting datasets are required, and necessary weights can be (re-)configured according to empirical manual feedback or settings rather than requiring actual digitalized data for analysis and training. This enables more traditional companies with weak digitalization and low to no sensor coverage an intelligent assignment capability without a costly and long running preparation phase. Also, this capability can transfer the intelligent assignment with adapted weights instantly to all parts of its production and workflow. Moreover, our intelligent assignment approach avoids the (costly) training phase typically required by ML approaches.

While it may seem possible to achieve similar functionality with an alternative non-fuzzy approach, due to the benefits and arguments noted above, we believe this would require far more work to achieve the equivalent out-of-the-box functionality, flexibility, and maintainability that a fuzzy set approach provides (without necessitating preexisting data or training).

For determining the assignment score and assigning the most suitable worker, the ACE can either directly assign the agent via a REST-API (present in many of the most popular BPMS), or the assignment could be conducted in the sync service utility activity via script access to the BPMS from within the process itself.

Due to the complete decoupling of BPM and ACE, the latter can be scaled independently of the scaling of the BPMS, such that a high workload on one of these engines does not affect the performance of the other. The separation further allows the implementation of an optimized multi-processing and scaling functionality, guaranteeing optimal efficiency even at high load (requirement 2). The performance optimization takes place at different levels. First, a multi-threading approach is utilized in the Request Layer following reference architectures for REST and Pub/Sub APIs. The subsequent handling of the request in the Data and Algorithm Layer is handled in a separate process decoupled from the Request Layer. To further accelerate large assignment calculations, the IAC has its own scaling function introduced in Section IV.

Figure 3 shows simplified workflow graphs for two BPMN processes using the IAC to assign an agent to its execution activity. In Figure 3A) this is done via the concept described above, utilizing an asynchronous script utility activity. This activity requests an assignment from the ACE via REST and instantly receives an empty response to complete the utility activity. Afterwards, an external rule engine is called via REST using the Rule Interface. Receiving the response, the Assignment Handler than triggers the IAC to calculate the assignment scores via fuzzy sets. Afterwards, the Assignment Handler assigns the execution activity via the BPMS REST-API to the most suitable agent. The execution activity is available all this time (right after async script utility activity is completed by the empty REST response from the Assignment Handler), however, it is not assigned to any agent until the Assignment Handler does so via the REST API.

In Figure 3B. the assignment is handled via a synchronous service utility activity, calling the rule engine directly via a REST call and incorporating the IAC directly in the script activity itself. This approach supports the execution activity during the assignment process and does not allow refined scaling options outside of the BPMS itself.

As previously mentioned, it is also possible to call the Assignment Handler via a synchronous service utility task, merging the two approaches displayed in Figure 3. This works similarly to asynchronous script task approach, with the difference being that the assignment is handled by the service task rather than via an API call from the Assignment Handler.

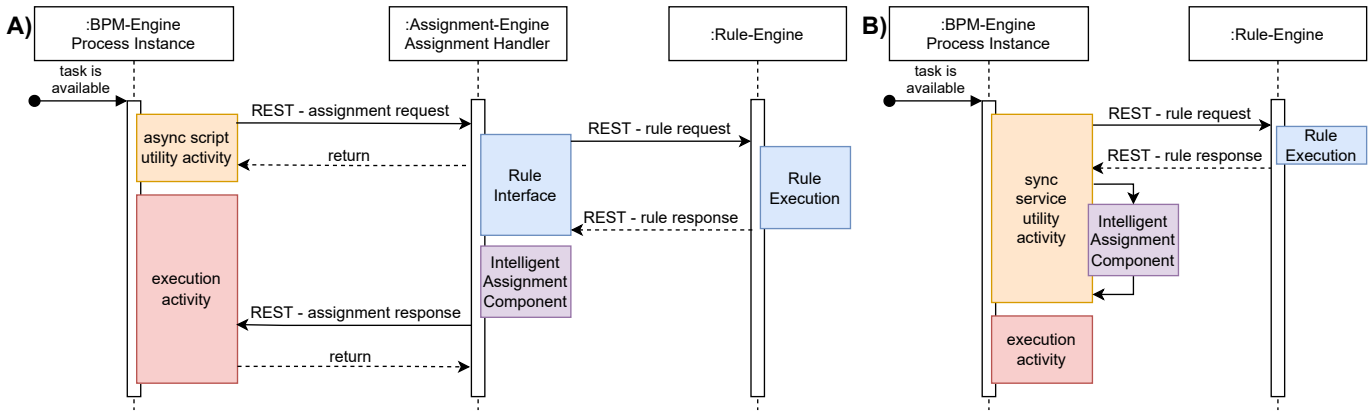

Figure 3. Different integration options in a BPM-engine with the utilization of a rule engine.

IV. Intelligent Assignment Component

The IAC is a standalone component of the ACE Assignment Handler. Containing the fuzzy logic for the assignment calculation, it is the functional core of the engine. This section highlights the conceptual decisions behind the component and details its internal structure.

A. Models

In order to compute meaningful assignment scores, the IAC requires a custom set of models. This is provided by the Assessment Criteria which are supplied as part of each activity and agent data set. These data sets can either be directly sent by the BPMS, when a new assignment request is sent, or be actively collected by the DAC via the BPMS API and connected data sources. For the second version the ID of all available agents and the activity must still be provided as part of the assignment request.

These models were also used as a foundation to create the Context and Augmented Reality eXtension (CARX) for BPMS [30]. This extension includes a modeler to create new BPM process templates, which contain all information required to enable an assignment via the IAC, while being fully BPMN 2.0 compliant and supporting the easy addition of information to new processes or the upgrade of existing processes.

1) Assessment Criteria Model: The Assessment Criteria consist of five parameters (oriented on real-world examples) that define the values for determining the assignment. These are required in activities and agents used with the IAC. It can be viewed as an interface required by all data and components connected to the assignment.

Distance: calculates the distance between agent and activity position optimizing assignments regarding the travel distance. Position objects contain 3D coordinates with numeric values for X, Y and Z.

Qualification: calculates the difference between the required qualification for an activity and the existing qualification of an agent. It answers the Boolean question if the agent is capable of performing the activity, and permits the determination of a possible over-qualification to prevent utilizing expensive agents on trivial activities. Qualification objects consist of the four parameters: ”electrical”, ”computer”, ”engineering” and ”bio_chemical”, which represent the different skills of agents or activity requirements in this area. As this skill cannot be calculated automatically and must be defined by humans, each parameter will be represented by a number between 0-10. This provides an accustomed scale to rank skill and requirements instead of a default fuzzy scale from 0-1, which is more abstract and an unusual scale for people.

Hourly Rate: calculates the extra cost of using a given agent for an activity per hour in Cents. This prevents the usage of external/temporary workers that incur extra costs if a similar qualified employee is available. This should not include the salary of permanent staff, as their salary is independent of their utilization rate. The cost is represented by an integer to prevent floating errors.

Workload: calculates the capacity utilization of agents, preferring agents with few enqueued activities and preventing overloaded agents form enqueueing additional activities. Thus, load balancing between resources and compliance with labor protection regulations can be supported within the algorithm. This could either be added to the agent itself by the BPMS, or can be calculated using the agents work list, which is present in all BPMS. The parameter is represented by an integer value.

Danger Level: calculates if an agent can safely perform an activity. As some activities have special hazards and given safety regulations, only agents with an appropriate safety clearance can be assigned to these. The Danger is thereby defined by an object consisting of the four parameters: ”noise”, ”heat”, ”electrical”, and ”chemical”. The separate values are represented by float values between 0 and 1. This provides an abstract concept, but can easily be modified for more concrete parameters as required for a given concrete use case.

2) Activity Model: This model can consist of any BPM activity extended with the Assessment Criteria except Hourly Rate and Workload, since the cost of an activity is irrelevant for its optimal assignment, and an activity itself has no workload in the context used by the IAC.

As the model extends BPM activities the activities priority remains intact and is respected by the IAC during activity assignment.

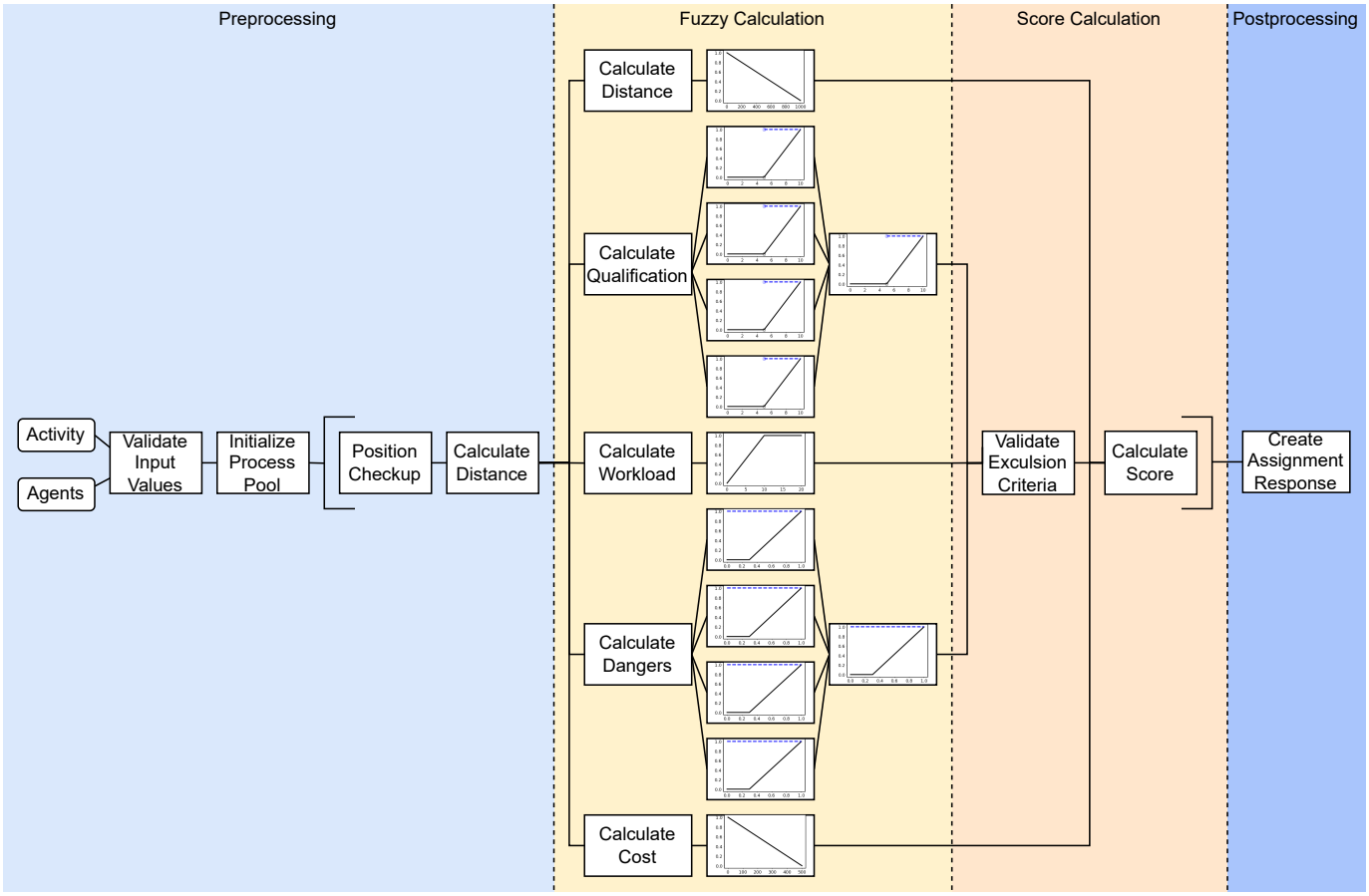

Figure 4. IAC - workflow tree.

3) Agent Model: The model consists of a BPMS human resource (user) extended with the Assignment Criteria as attributes. In a minimal engine configuration, such a resource might only contain an ID. In contrast to the activity, all criteria are mandatory, as they all provide valuable data for calculating optimal suitability levels. The Danger Level object is renamed to Danger Threshold on the agent level for a more descriptive and easier-to-understand naming. If a task is assigned to an agent the danger level of the activity is subtracted from the agent's threshold, preventing an overload with too many dangerous activities, which could lead to potential labor law violations and increased incident risk. After a resting period, the agent's danger levels are reset (e.g., at the start of the agent's next shift or after a longer break).

B. Overall Assignment Algorithm

The internal algorithms in the IAC are based on a Fuzzy Logic approach. In contrast to ML, no existing datasets are required, only a scheme of the data is mandatory to configure the fuzzy sets. However, the same level of fine calculation of the suitability score is possible, as opposed to the simple calculation of suitability based on chaining. As described earlier, the activity and agent list are provided to the component either directly by the BPMS or via the DAC according to the aforementioned models, and can therefore be directly supplied to the algorithms as parameters without further aggregations or parsing. After executing the algorithms, the IAC will return the suitability level for all provided agents to the Assignment Handler.

To speed up the processing time for large numbers of agents, the IAC can run calculations in a multi-processing configuration with multiple available modes. This allows an optimal resource allocation concerning the concrete assignments, rather than a general solution that could slow simple assignment calculations or non-optimally benefit more complex calculations.

An approximate workflow of the IAC can be seen in Figure 4. However, while parts of the fuzzy calculation are shown in parallel, it is actually executed sequentially. This format was chosen for its space-efficient layout. The order of the different calculations does not matter for the final result. The block displayed in brackets can either be executed in parallel via multi-processing or sequentially as a loop. The figure also contains images of the used fuzzy models to provide a general idea how they look. Figure 5 displays the over-qualification model in detail.

1) Preprocessing: After providing the activity and agent data to the IAC, some preprocessing steps take place (c.f.

Figure 4 Preprocessing). While the input values are already validated and per definition complete, the position entry for tasks is not necessarily set, as other defaults can be used. Namely, the position of a connected machine - if no positional constraints are enforced, and the position of the agent. Therefore, if no position is set, "Validate Input Values" attempts to determine the position of the machine connected to the task. If no machine is set, the value is left empty.

For the second step "Initialize Process Pool", the configuration is checked for the multi-processing flag. If this flag is set, it is checked if a pool amount is set. If no pool amount is set, the number of processes in the pool is set to $n - 2$ where n is the number of cores of the host system. A process pool is created and the following steps shown in brackets are executed in parallel by the different processes for each agent supplied to the IAC. Else, if no multi-processing flag is set, they are executed in a loop, once for each agent, in a sequential manner.

"Position Checkup" is the final agent-specific validation of the position. If, in earlier steps, no position was assigned to the activity, it is set to the agent's current location. This is required, as the fuzzy set for distance requires a position value of the activity, even if the activity itself is not location-dependent. If the activity position is already initialized (e.g., via the activity context or a connected machine) this step is skipped.

The final preprocessing step "Calculate Distance", calculates the Euclidean distance between the agent and activity. This is done by transforming the position values of agent and activity into three dimensional vectors (x, y, z) and deducting them. This can be used as a simple orientation. In a finalized productive system, however, this should be replaced by a more refined path calculation algorithm, providing a more resilient calculation for the following fuzzy set.

2) Fuzzy Calculation: After the preprocessing steps, the calculation of the actual fuzzy models can begin. Overall, the agent with the highest score is preferred. Therefore, all Assignment Criteria for each requested agent are calculated distinctly, with 1 being the best score possible and 0 the worst. Besides the score calculation, the fuzzy sets also contain certain exclusion criteria. In this section we only mention when they are reached, with a brief description why they are chosen. In the next section, we further elaborate these exclusion criteria in detail. The concrete calculation for each value is conducted as follows:

Distance: this algorithm uses the distance calculated in the corresponding preprocessing step as a basis. Distances between 0 and 1000 are mapped to a fuzzy value from 1 to 0, where all distances above 1000 are also mapped to 0, in order to simplify the handling of large distance values. Therefore, agents in close proximity to the activity are preferred in the assignment, and all agents further than 1000 units away are heavily discriminated by the fuzzy set. However, as this is no exclusion criteria, even agents further than 1000 units away could still be assigned to the activity if they are the overall most suitable agent.

Workload: the workload can take values between 0 and 20 at maximum. Values between 0 and 20 are mapped to the fuzzy value between 1 and 0, while values equal to or greater than 10 are mapped to 0. This still allows the assignment of new activities to workers that already have 10 or more activities assigned to them, but prefers those with smaller workloads. Further, if the workload has a value of 20, the score is set to 0 and triggers the overloaded exclusion criterion, preventing the algorithms from assigning any more activities to this specific worker, thus preempting the overburdening of agents.

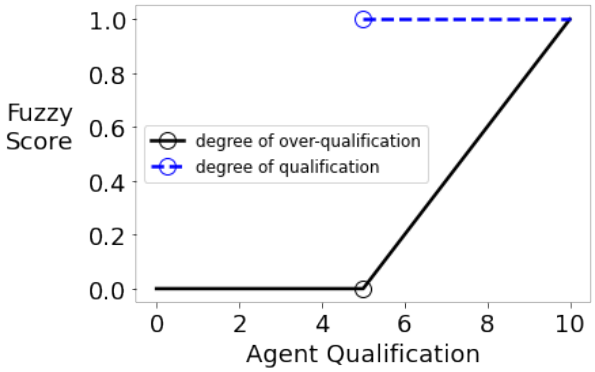

Figure 5. Qualification fuzzy model showing fuzzy score of agent over-/qualification for a task with qualification requirement 5.

Danger Thresholds: for each danger value, a separate fuzzy set is calculated. After the disjunct calculations, all values are added to a common fuzzy domain and weighted according to a configuration (cf. Figure 4 "Calculate Dangers", first each domain is processed isolated, then all values are combined in a single domain). In the default case, all values are weighted the same, leading to a 25% weight per value. All danger values between 0 and 1 are mapped to fuzzy values from 0 to 1, where all values below the activity's danger level are 0 and trigger an exclusion criterion. This addresses labor law regulations while increasing worker safety. All values above the requirement through the maximum danger threshold of 1 are mapped between 0 and 1. An agent who approximately meets the requirements can therefore work on the activity but gets a score of 0. This prefers agents with higher danger thresholds, as they are most likely more experienced and more rested than agents with lower danger thresholds. While a valid agent who barely fulfills the requirements and an agent who fail the requirements both get a score of 0, a final filter in the "Validate Exclusion Criteria" Algorithm of the Score Calculation block removes all invalid agents from the assignment.

Qualification: the qualification is calculated in three separate fuzzy models. First, the four values of qualification (electrical, engineering, computer, bio_chemical) between 0 and 10 are compared to the values of required qualification of the activity between 0 and 10 via separate fuzzy sets similar to the danger levels. All values below the activity's requirements are assigned to 0 and trigger an exclusion criterion, as the agent is technically not capable of performing the requested activity. All values above the requirement are assigned to 1. Subsequently, the degree of over-qualification is calculated in the

TABLE I. IAC EXCLUSION CRITERIA.

Criteria	Condition	Explanatory note
Qualification	Agent qualification below activity requirement	This prevents agents from being assigned to activities they are formally not capable of, potentially reducing rate of errors, incident risk and execution time.
Workload	Workload of agent greater or equal to 20	This prevents the overburdening of agents with too many activities. Also, the general time before an activity is executed is reduced, as they are split more equally between the available workers.
Danger level	Agent's danger threshold below activity's danger level	This prevents potential labor law violations and increases the safety of activity executions.

second fuzzy model. Starting from the required qualification up to the max qualification of 10, each qualification value is assigned to a fuzzy value between 0 to 1, where 0 perfectly fits the required qualification value and 1 is the maximum amount of over-qualification possible. Afterwards the over-qualification is subtracted from the qualification value resulting in a value between 0 and 1 called degree of qualification, where 1 is a perfect fit without over qualification and 0 is a maximum over-qualification.

These values were all empirically selected and provided good results during our early tests. In a productive environment they could be applied for a base configuration. However, we recommend the selection of own values tailored to the concrete environment.

Figure 5 displays the two fuzzy models described above for an activity's qualification requirement of 5. The agent's score value for the (under-)qualification is displayed as the blue dashed line, where the fuzzy set for all possible agent qualifications below 5 returns 0, and jumps to 1 for all values fulfilling the requirements. The over-qualification is highlighted as a solid black line, the transition points between the qualification score of 0 to 1 are marked by circles. All agent qualifications below and at the required activity qualification of 5 return 0, representing no over-qualification. Above 5 the agent's qualifications are slowly assigned to a fuzzy value between 0 and 1, representing an increasing degree of over-qualification.

After this, two steps are conducted for all four properties of an agent's qualification. The four separate degrees of qualification are added to the final fuzzy domain and weighted according to the configuration. The process is identical to the Danger Threshold calculation and also uses a 25% weight distribution per value as a default. The resulting value is used as the qualification in the final calculation of the score, preferring qualified agents with as low an over-qualification as possible.

Hourly Rate: the hourly rate is mapped to the fuzzy value from 1 to 0 for values from 0 to 50000 (being equal to 500€ following the integer data format). All values over 50000 are set to 0. This prefers agents with low additional cost (like employees) over external workers, costing extra money and therefore improves the economic efficiency of the process.

3) Score Calculation: In the score calculation step, two algorithms are employed. First, the tree aforementioned exclusion criteria are involved. These could be triggered by the calculation of the fuzzy sets as described above. Table I lists them in detail.

To check if an exclusion criterion was triggered, the fuzzy value for each of the three domains is checked. If one of these values is 0, it is further checked if the agent's value is below the task's value. If this is also the case, the exclusion is triggered. Following this, the assignment score is set to 0 and the Score Calculation step is completed. As a fuzzy value of 0 is per definition valid (e.g., just fulfilling the qualification requirement also receives a fuzzy value of 0), comparing the agents value against the tasks value is required. Setting the fuzzy value to another value, e.g., -1, is not possible, as this would interfere with the calculation of the combinatorial fuzzy sets for qualification and danger.

Another approach could be throwing the exclusion criteria as an exception right after encountering it. This could potentially reduce the computing time of the IAC, as no further calculations would be required after encountering the first exclusion criteria. While this coding-by-exception approach is generally considered bad practice, it would be possible to move the "Validate Exclusion Criteria"-block right after each fuzzy model containing exclusion criteria, rather than having one block after all calculations.

After the exclusion criteria are checked and none was thrown, the agent's assignment score for the activity is calculated. This is once again achieved by a fuzzy set, combining all scores from the five previous fuzzy sets and once again applying a weight to them, similar to the calculation of the danger and qualification values. The weights for this final score calculation can either be supplied on a per assignment request basis directly by the BPM process, or via a default value in a configuration file. The first approach is quite beneficial if an activity is very one-sided and strongly focuses on one assignment criteria, e.g., a focus on distance alone if a fire has to be extinguished.

When the final fuzzy value has been calculated, it is multiplied by 100, giving the agent a final assignment score between 0 and 100 for the activity, while 0 means the agent is completely unsuited and 100 is a perfect fit.

4) Postprocessing: As the last component of the IAC, the "Create Assignment Response" algorithm is triggered. This takes all calculated assignment scores together with the agent's id as an identifier and orders them by score, putting the highest score on top. Afterwards the final assignment score structure is returned to the ACE for further processing.

V. IMPLEMENTATION

The ACE has been implemented as part of the Augmented Reality Process Framework (ARPF) [31], additionally incor-

porating AR and context support for workers during the execution of tasks. However, following the modular solution concept, it is still possible to use the IAC as a standalone module (e.g., directly integrated in a BPMS) or to use the ACE as a standalone product without the ARPF. CARX also utilizes ARPF (and therefore the IAC) as the CARX BPMS IIoT Extension Framework.

Like ARPF, our prototype of the ACE with focus on the IAC is implemented using Python. This approach was chosen for its fast prototyping capabilities while still providing performant libraries and refined multi-processing logic. As a base image for the ACE, a Django server was created, providing the most powerful REST-Server available for Python. In contrast to other Python server-frameworks, Django offers not only fast and simple prototyping capabilities, but can also be scaled up to a performant production deployment. To provide the required REST interface, the Django REST framework was integrated. A Pub/Sub interface was implemented using Python paho, the Python MQTT [32] framework from Eclipse. Architecturally the DAC is adjacent to the REST Layer. It can invoke REST requests on its own, aggregating all required data from the BPMS or configured external data sources.

The fuzzy portions of the IAC were implemented using the fuzzylogic library for Python 3 [33]. As an example, the complex fuzzy set for the qualification is displayed in Figure 6. It consists of the calculation of the under- and over-qualification, as well as the final score with its defined weights.

As BPMSs for our prototype, we integrated AristaFlow [34] (using a synchronous service activity approach) and Camunda [35] (using an asynchronous script activity approach). In the following, we focus on the Camunda implementation. It is a well-known application in the BPM context and further provides all required functionality as well as a BPMN Modeler as an open-source solution. In addition to a full implementation of the BPMN 2.0 standard, Camunda also provides a Connector element, allowing easy REST requests from within process instances via script and service activities. The well documented REST-API [36] supports a quick and easy integration of the communication interface.

As the free version of Camunda only provides a BPMS with minimal user management, an extension in form of a minimal REST-Backend (further called CamundaClient) handling users and assignments was required. Users are added via a new backend and saved according to our agent model. The process templates were extended as planned in the solution approach. The utility activity requests a score calculation from the CamundaClient for the subsequent execution activity. This execution activity must contain the Assignment Criteria as described in the activity model. The CamundaClient then loads the required user data from the database and sends a request to the ACE. It is also possible to move this step to the DAC in the ACE; in this case, it would only be required to send the activity ID to the ACE. While we implemented both the synchronous and asynchronous variants, we focus here on the asynchronous one, as it provides additional benefits such as better multi-processing support and should be chosen if

```
def eval_qualification(required: dict, values: dict):
    """eval qualification and over qualification for tasks
    required value and agents value"""
    qualifications = {"types": {}}

    for qualification_type, required_qualification in required.items():
        Qualification = Domain("qualification", 0, 10, res=1)
        qualified = (
            rectangular(required_qualification, 10)
            if required_qualification < 10
            else singleton(required_qualification)
        )
        over_qualified = (
            Set(R(required_qualification, 10))
            if required_qualification < 10
            else Set(constant(0))
        )
        begin_flat = Set(singleton(required))

        if required_qualification != 0:
            begin_flat = MAX(
                ~Set(rectangular(0, required_qualification)),
                Set(singleton(required_qualification)),
            )

        Qualification.qualified = qualified
        Qualification.over_qualified = over_qualified
        Qualification.not_over_qualified = (
            Set(S(required_qualification, 10))
            if required_qualification < 10
            else Set(begin_flat)
        )

        if required_qualification != 0 and required_qualification != 10:
            Qualification.not_over_qualified = product(
                begin_flat, Set(S(required_qualification, 10))
            )
        if required_qualification == 10:
            Qualification.not_over_qualified = begin_flat

        qualifications["types"][qualification_type] = float(
            Qualification.min(values[qualification_type])
        )

    Qualification = Domain("qualification", 0, 1, res=0.001)
    weight = {"electric": 0.25, "computer": 0.25, "social": 0.25, "bio_chemical": 0.25}

    w_func = weighted_sum(weights=weight, target_d=Qualification)
    qualifications["weighted"] = w_func(qualifications["types"])

    return qualifications
```

Figure 6. Code snippet showing fuzzy implementation for qualification.

supported by the utilized BPMS. As soon as the assignment is calculated, the assignment scores are sent from the ACE to the CamundaClient and the assignment in Camunda is handled via the client. Connected to this, the workload of the assigned agent(s) is increased and their Danger Threshold is decreased by the Danger Level of the activity. The Danger Levels can further be reset to the agents' default value, e.g., on a daily or weekly basis as required by labor safety laws.

Alternatively, the IAC could be integrated in the CamundaClient itself, removing the need for the additional REST-Requests between the client and the ACE. The current approach, however, supports the generic usage of the ACE as

```
package com.paradigma.temperature_rules;

Rule "TemperatureLowerThan"
when
    $sensorData: SensorData()
then
    $sensorData.setValid(
        $sensorData.getMaxValue()
        > $sensorData.getCurrentValue()
    );
end
```

Figure 7. Example precondition rule validated by Drools.

a service for multiple BPMS simultaneously and entails less restrictions.

The Rule Interface was implemented for the Drools rule engine [37]. The engine was chosen for its widespread use in the industry as well as its REST interface. While the implementation of the Rule Interface itself has to be customized via an adapter to the utilized rule engine itself, the internal API called by the Assignment Handler was created in a generic way, allowing an easy exchange of rule engines.

Currently the Rule Interface can only be used to filter activities itself. For this, preconditions have to be created in the language of the rule engine and have to be added to the process template. The precondition is then sent to the rule engine when the assignment is requested for the connected activity and it is evaluated. This is available in two modes; loop and single. In loop, the rule is executed in a loop with a small delay between executions until it is fulfilled. In single mode, the condition is evaluated once, if this fails an error is returned to the BPMS. While currently no pre-filtering of agents is supported, this could be added to the Rule Interface in coming updates. Figure 7 shows a simple example for such a precondition rule, validating that the current temperature of a sensor $\$sensorData.getCurrentValue$ is below the maximal allowed threshold $\$sensorData.getMaxValue$.

Powerful multi-processing capabilities were implemented in the Intelligent Assignment Component and managed by an intelligent orchestrator. While the Assignment Component is already realized with a runtime of $O(n)$, its performance can be further increased with our multi-processing approach. Assignments with large numbers of agents can therefore be run in a multi-processing configuration with multiple modes. The default for large requests is $n-1$ processes, where n is the maximal number of cores available on the machine. This provides maximum calculation speed while still preserving one process for the ACE itself, preventing slowdowns. If the request is too small for multi-processing (the multi-processing overhead would slow down the computation speed), the orchestrator runs the calculation in a single process. Finally, it is possible to run the calculation in $n-m$ processes, where $m, m < n$ is calculated according to the server's performance in multi-processing mode. We implemented a semi-automatic test setup, calculating the optimal m for a server for 10, 100. 1000, ... 1000000 agents in a single assignment request. The calculated m can then be used in the server configuration to allow maximum performance according to the utilized hardware.

VI. Evaluation

The evaluation focuses on two aspects of the solution: the first considers performance and scalability implications of our IAC agent assignment algorithm utilizing our prototype within a realistic software and hardware environment, in order to determine if there are unforeseen practical limitations or bottlenecks that would hinder its usage. For the second part, we evaluate assignment optimizations achieved when a BPMS utilizes the IAC versus a BPMS only (Camunda without the IAC) in order to determine if a significant benefit can be shown.

MatPlotlib [38] was used to briefly analyze the data as well as to graphically process the results. The evaluation itself was conducted utilizing the Python libraries pandas [39], SciPy [40] and NumPy [41].

A. Performance Evaluation

The performance evaluation was conducted to analyse the performance and scalability implications of our IAC on a virtual server with 90GB of main memory. As an operating system, Debian 10 was chosen utilizing Python 3.7.2 for the algorithm execution. The test was separated in two groups of 10, 100, 1000, 10,000 and 100,000 agents being assigned to a single activity. In the first group, all agents were capable of performing the activity according to the assignment criteria. In the second group, only certain agents were capable of performing the activity. The assignment of each group of agents to their activity was conducted 100,000 times. The groups from 10 to 1000 agents were assigned using the IAC without multi-processing while 10,000 and 100,000 agents were assigned using 17 processes ($n-1$ mode).

Figure 8 displays the assignment calculation performance if all supplied agents were capable, while Figure 9 displays the calculation performance if only some agents fulfilled the requirements. The calculation duration results show an approximately linear scaling in the single processing mode (10-1000 agents), while multi-processing decreases with larger numbers of agents (10000-100000 agents). Unexpectedly, calculation duration for assignments with only capable agents is lower than that of agents with mixed requirements. This originates from some optimization problems in the elimination of incapable agents. A possible solution to this can be found in the evaluation summary below.

In general, the assignment of high volumes of agents caused no issues for the algorithms. As the IAC is meant to run behind SARs, rule engines, or other performant basic filtering algorithms, a load of 10,000 possible agents for a single activity is further quite unlikely. The runtime in the sub seconds for agent values below 100,000 would also allow the

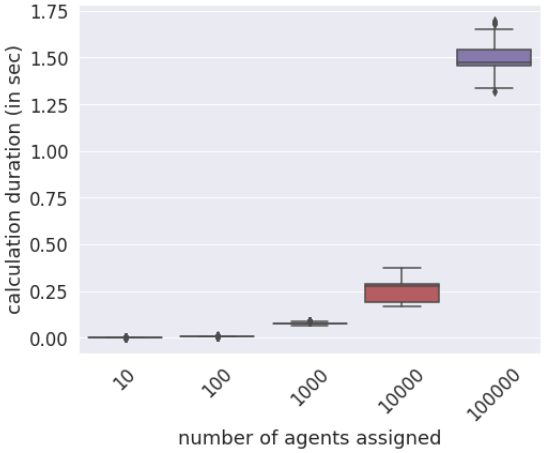

Figure 8. Calculation performance vs. number of capable agents assigned.

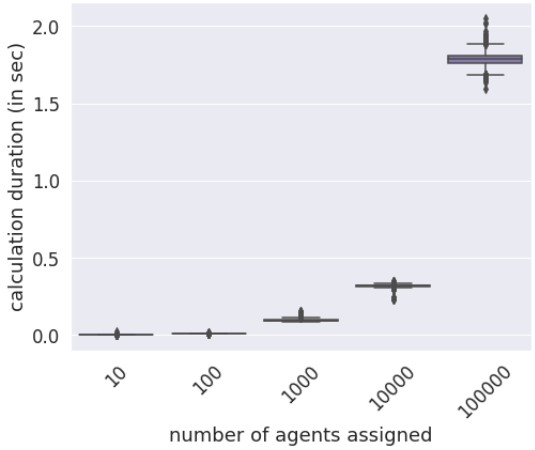

Figure 9. Calculation performance vs. number of capable and non-capable agents assigned.

removal of preliminary filtering, reducing the runtime of the whole BPM process.

B. Integration Evaluation

The integration evaluation was conducted using the AnyLogic simulation software. The AnyLogic simulation was run on a Lenovo T495 with 14GB main memory utilizing Arch Linux as an operating system. No changes were made between the performance evaluation and this one besides the setup of this edition. The laptop and server containing the BPM and ACE were on the same network.

The evaluation was used to compare a BPMS using the IAC against a plain BPMS. To simulate workers and a realistic workflow, an AnyLogic simulation was built and two simulation setups were configured.

A factory with 21,504m^2 and a total of 29 machines which required maintenance every 16 hours was 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, it started a new Camunda process instance with the required qualification and the machine's position. 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. A total of 5 agents were available to complete this activity; four internal workers, waiting in a maintenance building in the factory hall and one external agent, waiting 165 meters away. The internal agents had engineering qualifications of 4, 5, 6 and 7 while the external agent had an engineering qualification of 8. The other qualification values were set to 0 to avoid bias. The usage of the external agent was connected to an additional cost of 2500 (25€/activity), while the usage of internal workers incurred no additional costs. In their idle state, an agent checked every 5 minutes if a new activity is available. If they were working, after completion of their current activity they checked if another activity was enqueued. If no activity was enqueued, they switched back to the idle state and moved to their starting position. This part of the setup was identical in both simulation setups.

The factory size and machine breakdown/maintenance intervals were empirically selected to achieve a high utilization of the available agents without overstraining their capacity, allowing for a realistic environment. The qualification of the internal and external assignments were selected in a way, which allows the internal agents to do most of the work on their own, but requiring the external agents support with high workloads. We have refrained from mixing more different tasks/more complex tasks with many different requirements, to provide an easy to understand and analyzable evaluation. Mixing more different task types/requirement constrains would not affect the technical evaluation in a meaningful way, as all fuzzy models are run even with requirements of 0, however, it would become harder to understand and analyse the simulations findings.

In the Camunda Setup (called CMD-Setup), the agents fetched their activities directly from Camunda. All activities of the simulation were available to all of the workers with no further verification. If an activity is available to the group, the agents try to claim it and, if successful, work on it. In the IAC Setup (called IA-Setup) the agents checked their personal worklist at the ACEs REST API. If their personal worklist contains an activity, they start to work on it, otherwise the stayed idle.

A timespan of 36 working hours 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 the statistical relevant test data. For the IAC, the model introduced in Section IV was used. The qualification value was weighted half to increase utilization of the more qualified agents 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 II shows a general comparison between the CMD (only Camunda, no IAC) and IA simulation (Camunda with the IAC), while III shows a more detailed comparison of internal

TABLE II. IA/CMD-SETUP SIMULATION MEASUREMENTS.

	IA	Camunda
total_activities (amount)	13.98	16.84
work_time (in minutes)	1636.38	1955.20
idle_time (in minutes)	523.62	204.80
cost (in €)	10.00	420.00
avg_over-qualification (value)	0.34	0.09
max_avg_under-qualification (value)	0.00	-0.02
traveled_dist (in meters)	7346.92	8911.79
downtime_maintain (in minutes)	484.18	303.30
downtime_repair (in minutes)	204.00	138.23

TABLE III. INTERNAL/EXTERNAL WORKER SIMULATION MEASUREMENTS.

	IA-int	IA-ext	CMD-int	CMD-ext
total_activities	17.38	0.40	16.85	16.80
work_time	2037.25	32.88	1946.62	1989.50
idle_time	122.75	2127.12	213.38	170.50
cost	0.00	10.00	0.00	420.00
avg_overqual	0.05	1.50	0.06	0.20
max_avg_uqual	0.00	0.00	-0.02	0.00
traveled_dist	9082.78	403.45	8750.17	9558.25

(-int) and external (-ext) worker stats in both simulations. In the following values from Table II will be compared with the more detailed values from Table III.

The average work time and total activities per worker are lower in the IA run, while the utilization of the internal workers (IA-int) is slightly increased and the external utilization (IA-ext) is heavily reduced. The average idle time is increased which can be deducted from the low external utilization. The heavily reduced average cost of a simulation run, if using the IAC instead of a plain BPMS, can be attributed to the preferred use of internal workers.

The increase in over-qualification while using IA instead of plain Camunda can be explained with the low weighting of qualification in the algorithms, as well as the lack of under-qualification in comparison to the CMD-Setup, where under-qualification was generally present. In Table III, the main source of over-qualification in the IA simulation comes from the usage of the external worker, who was mainly used for activities below his qualification. This happened because of a extreme workload and could be solved by employing another internal worker with lower qualification to help out with this activity. This would lead to reduced cost and downtime. Optimization in the simulated company is needed, rather than an adaptation of the algorithm.

The traveled distance for the internal workers is slightly increased in the IA simulation compared to the CMD run. This, however, stands in linear dependency with the increased workload. A stronger weight regarding the distance could reduce this effect.

The downtime in the IA run is around 50% higher than in the CMD-Setup, while the cost was reduced to 4.2% of the CMD-Setup. This was expected behavior, as the algorithms by default try to save money and therefore did not employ the external worker as much as the CMD-setup.

C. Summary

The IAC performed as expected with fast execution times on mid-to-low budget hardware. Scaling was only required for the case when more than 1000 agents could be assigned to the same activity. This is highly unlikely even in companies with more than 1000 employees, as the BPMS most of the time already pre-filters valid agent groups. Further, the number of agents available for assignment in such large corporations could further be reduced by extending the precondition filtering as mentioned in Section V. If further scaling is necessary, it can be readily achieved and works efficiently for at least 100,000 agents per activity. The algorithms further produce comprehensible results for analysis by non-experts, which can be adjusted as required through dynamic weighting of the different variables in the algorithm.

Due to the non-blocking REST-API design and decoupled async assignment process, we do not foresee any multi-tenant performance issues.

Taking the current runtime in the sub seconds for less than 100,000 agents into account, the implementation of agent pre-filtering via rule engines should not be implemented for low- to mid-size systems, as the additional REST calls would ultimately mean a slowdown and increase the overall runtime.

However, the evaluation also shows an unexpected finding, whereby the calculation duration for assignments with only capable agents is lower than that of agents with mixed requirements. As in the second case, the execution criteria are triggered and the final score calculation can be skipped for some agents. This leaves room for optimization regarding the handling of exclusion criteria for future work.

VII. CONCLUSION AND FUTURE WORK

Industry 4.0 stands for highly automated production processes. However, these processes also rely on complicated tasks that can only be performed manually by humans. The integration of such activities into the processes is still problematic. One important issue is efficient task assignment, which is not solved well in contemporary BPMS.

To counteract this, the current contribution described an approach for more effective and efficient activity assignment for Industry 4.0 production processes. The focus of this approach was to build a compact model of fuzzy sets that can be easily applied to real projects, while also combining rules for pre-filtering for more obvious logical determinations and combinatorial constraints that do not require fuzziness. These rules can be easily adjusted and adapted by users and efficiently executed on a rules engine, thus focusing the fuzzy sets on those areas for which it is specialized. For our realization and evaluations we chose a set of important properties that incorporate aspects relevant in current Industry 4.0 production: achieve cost savings by incorporating not only under-qualification but also over-qualification, and the separation between internal and expensive external workers; achieve a balanced workload for all workers to avoid both idle time and overburdened workers; protect the workers from different hazards as enforced by government regulations; and finally,

optimize assignments with knowledge about the locations of workers and their potential activities by minimizing transit overhead.

Besides providing a practical model, our approach also features concepts for the direct integration with BPMS. To demonstrate its feasibility, we have currently implemented, integrated and tested our prototype approach with two concrete BPMS; AristaFlow and Camunda. The approach is built modularly and can be easily expanded. Furthermore, the fuzzy sets used to calculate an assignment utilize weights that can be changed dynamically according to the users' specific needs. It is also possible to use this prototype with any other BPMS supporting BPMN 2.0 with minimal effort.

The evaluation showed that our approach is an efficient way to automatically compute assignments. We evaluated the algorithms regarding performance and built a comprehensive simulation scenario to show its effectiveness and efficiency in providing optimal assignment recommendations. However, the Rule Engine interface, as well as the exclusion criteria filtering of incapable agents, still leaves room for optimization.

For future work, we plan to incorporate a more generic model where not only the weights are dynamic, but also the criterion. Instead of hardcoded values, it should be possible to define them via configuration files, or dynamically as part of the REST call to the ACE. Thus, the approach can be easily adapted for further domains and scenarios by extending or replacing the evaluation criteria. Additionally, we plan to rework the exclusion criteria in order to speed up the removal of unsuited agents from the assignment process. Other upcoming improvements could include utilizing transit path finding algorithms for the distance calculation in order to provide a more realistic and resilient calculation. The duration of activities could also be considered in order to measure the workload not only in the amount, but also in terms of estimated time required to complete the activities.

ACKNOWLEDGMENTS

We thank Felix Gräber for his contribution regarding the fuzzy algorithms and implementation, and Camil Pogolski for his input on the models. 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] G. Grambow, D. Hieber, and R. Oberhauser, "Intelligent task assignment in industry 4.0 production processes utilizing fuzzy sets," in *INTELLI 2021, The Tenth International Conference on Intelligent Systems and Applications*, 07 2021, pp. 1–9.

[2] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, 08 2014.

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

[4] D. Karagiannis, "Bpms: business process management systems," *ACM SIGOIS Bulletin*, vol. 16, no. 1, pp. 10–13, 1995.

[5] C. Moore, "Common mistakes in workflow implementations," *Giga Information Group, Cambridge, MA*, vol. 2, 2002.

[6] M. Arias, R. Saavedra, M. R. Marques, J. Munoz-Gama, and M. Sepúlveda, "Human resource allocation in business process management and process mining," *Management Decision*, vol. 56, no. 2, pp. 376–405, Jan. 2018.

[7] M. M. Flood, "The traveling-salesman problem," *Operations Research*, vol. 4, no. 1, pp. 61–75, Feb. 1956.

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

[9] G. Grambow, R. Oberhauser, and M. Reichert, "Semantic workflow adaption in support of workflow diversity," in *4th Int'l Conf. on Advances in Semantic Processing*. Xpert Publishing Services, 2010, pp. 158–165.

[10] G. Grambow, R. Oberhauser, and M. Reichert, "Employing semantically driven adaptation for amalgamatingsoftware quality assurance with process management," in *Second Int'l Conf. on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE'10)*. Xpert Publishing Services, 2010, pp. 58–67.

[11] G. Grambow, R. Oberhauser, and M. Reichert, "Enabling automatic process-aware collaboration support in software engineering projects," in *Communications in Computer and Information Science (CCIS) 303*. Springer, 2012, pp. 73–89.

[12] S. Kubler, J. Robert, W. Derigent, A. Voisin, and Y. Le Traon, "A state-of the-art survey & testbed of fuzzy ahp (fahp) applications," *Expert Systems with Applications*, vol. 65, pp. 398–422, 2016.

[13] V. Shahhosseini and M. Sebt, "Competency-based selection and assignment of human resources to construction projects," *Scientia Iranica*, vol. 18, no. 2, pp. 163 – 180, 2011.

[14] G. Kłosowski, A. Gola, and A. Świć, "Application of fuzzy logic in assigning workers to production tasks," *Distributed Computing and Artificial Intelligence, 13th International Conference*, pp. 505–513, 01 2016.

[15] H. Seifi, N. Shams, and K. M. Cyrus, "A novel self-regulating and intelligence meta-heuristic-fuzzy approach for integrated and optimal human resource allocation in normal and critical conditions," *International Journal of Fuzzy Systems*, vol. 24, no. 1, pp. 121–134, Jun. 2021.

[16] K. Kluza and G. J. Nalepa, "A method for generation and design of business processes with business rules," *Information and Software Technology*, vol. 91, pp. 123–141, Nov. 2017.

[17] D. Antonelli and G. Bruno, "Dynamic distribution of assembly tasks in a collaborative workcell of humans and robots," *FME Transactions*, vol. 47, pp. 723–730, 01 2019.

[18] X. Xu, J. Hao, L. Yu, and Y. Deng, "Fuzzy optimal allocation model for task–resource assignment problem in a collaborative logistics network," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 5, pp. 1112–1125, 2019.

[19] E. Simpson and S. Roberts, *Bayesian Methods for Intelligent Task Assignment in Crowdsourcing Systems*. Cham: Springer International Publishing, 2015, pp. 1–32.

[20] J. Quinzaños, A. Cartas, A. P. Vidales, and A. Maldonado, "iDispatcher: Using business rules to allocate and balance workloads," *Frontiers in Artificial Intelligence and Applications*, vol. 261, pp. 110–119, 2014.

[21] D. Li, G. Wang, C. Huang, S. Liu, and W. Zhang, "The research on auto-assignment method of service orders based on rule reasoning," in *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 1, 2019, pp. 2694–2698.

[22] G. Havur, C. Cabanillas, J. Mendling, and A. Polleres, "Resource allocation with dependencies in business process management systems," in *Lecture Notes in Business Information Processing*. Springer International Publishing, 2016, pp. 3–19.

[23] R. Sikal, H. Sbai, and L. Kjiri, "Promoting resource discovery in business process variability," in *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, ser. NISS19. New York, NY, USA: Association for Computing Machinery, 2019.

[24] J. Erasmus, I. Vanderfeesten, K. Traganos, X. Jie-A-Looi, A. Kleingeld, and P. Grefen, "A method to enable ability-based human resource allocation in business process management systems," in *The Practice of Enterprise Modeling*, R. A. Buchmann, D. Karagiannis, and M. Kirikova, Eds. Cham: Springer International Publishing, 2018, pp. 37–52.

[25] O. Vasilecas, D. Kalibatiene, and D. Lavbič, "Rule- and context-based dynamic business process modelling and simulation," *Journal of Systems and Software*, vol. 122, pp. 1–15, Dec. 2016.

[26] S. Ihde, L. Pufahl, M.-B. Lin, A. Goel, and M. Weske, "Optimized resource allocations in business process models," in *Lecture Notes in*

Business Information Processing. Springer International Publishing, 2019, pp. 55–71.

[27] W. Tan, L. Zhao, N. Xie, A. Tang, X. Hu, and S. Tang, "Methods for optimal resource allocation on cooperative task scheduling in cross-organizational business process," in *Computational Data and Social Networks*. Springer International Publishing, 2018, pp. 126–138.

[28] *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

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

[30] G. Grambow, D. Hieber, R. Oberhauser, and C. Pogolski, "A context and augmented reality bpmn and bpms extension for industrial internet of things processes," in *9th International Conference on Business Process Management (BPM 2021) Workshop Proceedings*. Springer, 2021.

[31] G. Grambow, D. Hieber, R. Oberhauser, and C. Pogolski, "Supporting augmented reality industry 4.0 processes with context-aware processing and situational knowledge," in *eKNOW 2021*, 07 2021, pp. 29–36.

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

[33] A. Kiefner. Python3 fuzzylogic. Last visited: 2021.06.14. [Online]. Available: https://github.com/amogorkon/fuzzylogic

[34] 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.

[35] Camunda. Last visited: 2021.06.10. [Online]. Available: https://camunda.com

[36] Camunda docs - rest api reference. Last visited: 2021.12.15. [Online]. Available: https://docs.camunda.org/manual/7.5/reference/rest/

[37] M. Proctor, "Drools: A rule engine for complex event processing," in *Applications of Graph Transformations with Industrial Relevance*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 2–2.

[38] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[39] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[40] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[41] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.