Intelligent Task Assignment in Industry 4.0 Production Processes Utilizing Fuzzy Sets

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*—Industry 4.0 production comprises complicated highly automated processes. However, human activities are also a crucial component of these processes, e.g., for machine maintenance. Task assignment of human resources in this domain is challenging, as many factors have to be taken into account to ensure effective and efficient activity execution and satisfy special conditions (like worker safety). To overcome the limitations of current Business Process Management (BPM) Systems regarding activity resource assignment, this contribution provides a BPM-integrated approach that applies fuzzy sets for activity assignment. Our findings suggest that this approach can be easily applied to complex production scenarios, while providing efficient performance even with a large number of concurrent activity assignment requests. Additionally, our evaluation shows its potential for improved work distribution which can lead to cost savings in Industry 4.0 production processes.**

Keywords—Business Process Management Systems; Business Process Modelling; Staff Assignment Algorithms; Assignment Automation; Fuzzy Logic.

I. INTRODUCTION

"Industry 4.0" stands for the fourth industrial revolution driven by digitalization [1]. 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 value chain by processing of real-time data from all stages involved in the chain. Cyber-Physical Systems (CPS) [2], which consist of information technology, machines, and built-in sensors, form an unit that enables comprehensive optimization of production with regard to criteria such as costs, resource consumption, quality, or availability. There is often a strong focus on autonomous systems and the highest possible degree of automation. Yet 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 business and production processes are typically governed by Business Process Management Systems (BPMS) [3] also known as Process-Aware Information Systems (PAIS). BPMS are in charge of the sequencing of the different activities belonging to a 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 [4]. 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) are utilized to achieve this. However, in Industry 4.0 production scenarios, many different factors have to be taken into account to find 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: She must have the necessary skills and abilities to correctly execute the activity, yet not be overqualified and this incur 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 [5]. Due to the high complexity of smart factories and their CPS, involving specialized external 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. that address hazards such as chemical, electrical, heat, and noise and may not be adequately tracked by automation systems.

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 able to perform an activity but cannot determine the degree of suitability. Fuzzy logic's [6] 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.

In prior work [7][8][9], we also developed an approach for contextual process management. However, this approach was tailored towards software engineering processes and did not use fuzzy sets or involve the complex specifics of Industry 4.0 nor AR processes. The main focus of this approach was extending processes with properties to enable automated software quality assurance and support collaboration of software engineers. This was realized by 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, fine grained levels of suitability are integrated to improve resource assignment results. To demonstrate its feasibility, we integrated our solution in a commons BPMS.

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 IAC. In Section V, we provide specific implementation details focusing on the Intelligent Assignment Component while addressing the overall prototype. Next, 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. The approach presented by Shahhosseini and Sebt [10] proposes a fuzzy adaptive model for competency-based employee selection. It provides a large set of competencies and a complicated fuzzy model. It is, however, created for construction companies and centers around four specific human roles. It also lacks an integration strategy with BPMS. Similarly, Kłosowski et al. [11] also discuss a fuzzy model for assigning workers to production activities. The main focus of this 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 lacks BPMS integration concepts.

Tasdemir and Toklu [12] provide an approach for fuzzy activity assignment integration with 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, some criteria for evaluating the performance of workers are not compatible with current legal regulations. Reijers et al. [13] also proposes a BPM-integrated model for activity assignment, in this case based on swarm intelligence. However, that model is targeted at emergency situations where timeliness is more important than other selection criteria and is thus also not suitable for Industry 4.0 production. In contrast to this,Antonelli and Bruno [14] deals 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. [15]. 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. Kwak et al. [16] also present a fuzzy approach for a specific domain, in this case audit staffing for accounting. As with the other approaches, these concepts cannot be easily transported to industrial production scenarios, and they do not address BPMS integration. Finally, a category of approaches similar to Simpson and Roberts [17] 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 a part of these approaches.

III. Solution Approach

While different approaches for fuzzy activity assignment exist, they are often rather generic and complicated or very specific and tailored to a certain domain. Furthermore, they are mostly not integrated 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 for this complicated domain in a practical and applicable manner, or approach has to satisfy 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 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 to the assignment process, permitting better performance optimization without the constraints imposed if one were to internally extend a specific BPMS. Furthermore, this decoupling via generic API supports a generic approach that can support integrations across a much wider range of BPMS. The conceptual architecture of the novel Assignment-Engine providing such functionality can be seen in Figure 1.

The Assignment-Engine uses a layer pattern which is further subdivided into modular components, with each of the layers contributing to the final solution. Via the modular layers, if desired, the Data and Algorithm Layer could be directly integrated into BPMS (potentially reducing performance optimization). 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 Assignment-Engine as a holistic solution to fulfill all the aforementioned requirements.

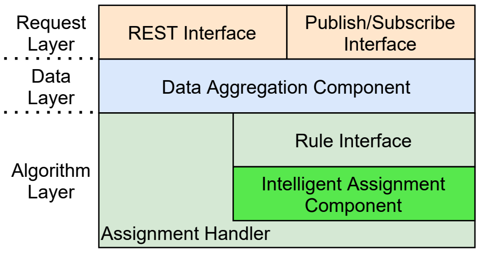
Request Layer
REST Interface
Publish/Subscribe Interface
Data Layer
Data Aggregation Component
Algorithm Layer
Rule Interface
Intelligent Assignment Component
Assignment Handler

Figure 1. Intelligent-assignment engine conceptual architecture.

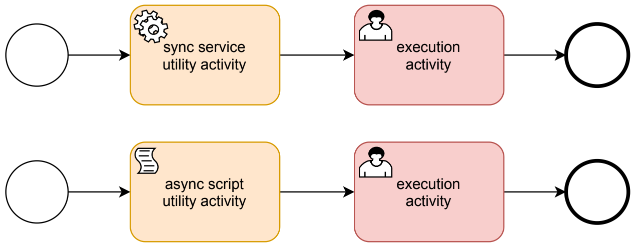

Figure 2. Activity solution variants (synchronous variant on top, asynchronous variant on bottom).

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 Assignment-Engine with any BPMS supporting BPMN 2.0 or later, as this standard offers a wide range of elements to integrate external services and functions [18] [19]. The integration takes place via a two activity concept. A utility-activity requests an assignment for the execution-activity following 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 Assignment-Engine 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 suitable level to the activity. The Async 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 Assignment-Engine 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 Assignment-Engine could even support multiple BPMS at the same time.

Once a request is received by the backend, the Data Aggregation Component (DAC) validates it. If all required information is present, the request is sent directly to the Assignment Handler. If some data is still missing the DAC can receive this by predefined external Sources, e.g., a database containing the agents or the BPMS public API.

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 therefore do not provide overall optimal assignments. While a Machine Learning (ML) approach could also be a feasible option, the fuzzy sets provide some striking advantages. For fuzzy sets, no preexisting datasets are required, and necessary weights can be configured according to experiences rather than actual data. This enables more traditional companies with weak digitalization and low to no sensor coverage an intelligent assignment capability without a costly and long running initialization phase and can transfer the intelligent assignment with adapted weights instantly to all parts of its production and workflow. Furthermore, our intelligent assignment approach without the training phase typically required by ML approaches.

In order to process the assignment score and assign the best suited worker, the Assignment-Engine either can directly assign the agent via REST-API (as 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 Assignment-Engine, the latter can be scaled independently of the scaling of the BPMS, and high workload on one of these engines does not interfere with the performance of the other engine. The separation further allows the implementation of an optimized multiprocessing 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 default 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 speed up large assignment calculations the Intelligent Assignment Component has its own scaling function introduced in Section IV.

Figure 3 shows a simplified workflow graph of a BPMN process consisting of an asynchronous script utility activity requesting an assignment from the Assignment-Engine and the execution activity being assigned after the calculation.

IV. Intelligent Assignment Component

The Intelligent Assignment Component (IAC) is a standalone component of the Assignment-Engine. 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 the internal structure.

A. Models

In order to compute meaningful assignment scores, the component requires a custom set of models. While Agents and Activity are supplied with each assignment request the

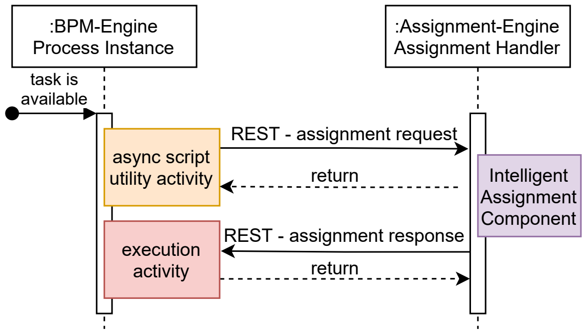
:BPM-Engine Process Instance
:Assignment-Engine Assignment Handler
task is available
REST - assignment request
async script utility activity
return
Intelligent Assignment Component
REST - assignment response
execution activity
return

Figure 3. Asynchronous integration in BPM-engine.

Assessment Criteria provides a model which the activity and agent data sets need to include. The agent and activity data can either be directly sent by the BPMS, or be actively collected by the DAC via the BPMS API and connected data sources.

1) Assessment Criteria Model: The Assessment Criteria consist of five parameters, oriented on realword examples and define the values with which the engine should conduct the assignment and which 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 overqualification 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 untypical 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 floats between 0 and 1. This provides an abstract concept, but can easily be modified for more concrete parameters as required a concrete use case.

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

3) Agent Model: The model consists of a user from the used BPMS extended with the Assignment Criteria. In a minimal engine such a user could 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. After a resting period, the agents danger levels are reset.

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 the DAC according to the introduced 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 processing time of large numbers of agents the IAC will be able to 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 small assignment calculations or non-optimally benefit large calculations.

1) Exclusion Criteria: The Algorithm further contains three exclusion criteria. If one of them is reached, the assignment score for the agent is set to 0 and further calculations are skipped, as the agent is not capable of performing the activity in a qualified or safe way. The agent's score of 0 will then be returned with the other results at the end of the calculation. The exclusion criteria are:

- Qualification below the required qualification of the activity. This prevents the assignment of unqualified agents to activities even if they are a perfect fit in all other regards.

- Workload equal to or larger than 20, as overloading with activities can lead to stressed out workers, increasing the risk of errors and accidents as well as decreasing their motivation and productivity.
- Danger thresholds lower than the required danger levels of the activity. This not only complies to work safety regulations but also prevents accidents.

2) Fuzzy Calculation: After providing the activity and agent data to the IAC, the calculation 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. The concrete calculation for each value is conducted as follows:

Distance: the distance between activity and agent is calculated from the difference between the position variables of both. These are transformed into three dimensional vectors (x, y, z) and deducted. The 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 reduce the distance agent should travel between activities at maximum. 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 they could still be assigned to the activity if they are the overall best suited agent.

Workload: the workload can take values between 0 and 20. 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 allows the assignment of new activities to workers with 10 or more activities, but prefers those with lower workloads, Further, if the workload has a value of 20 the score is set to 0 and acts as an exclusion criterion, preventing the algorithms from overburdening agents.

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. 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 prevents labor law violations and moreover increases work 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.

Qualification: the qualification is calculated in three separate fuzzy models. First the four values of qualification 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 overqualification is calculated in the 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 overqualification possible. Afterwards the overqualification 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 overqualification. 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 overqualification 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 BPMS.

After the calculation of the separate suitability levels, if no exclusion criteria is fulfilled, the assignment score is calculated for the agent according to a weight function in one final fuzzy domain. This weight function can either be configured generally and used for all calculations as a default or it can be defined on a per request basis when provided to the Intelligent Assignment Components algorithm as a parameter. All weights are defined between 0 and 1 to a total sum of 1. For the calculation, a new fuzzy set is defined and all values are summed up according to their weight. The result of the new fuzzy set is then multiplied by 100 to return a score between 0 and 100. Finally, the results are sorted in descending order and returned to the Assignment-Engine for further processing.

V. Implementation

Our prototype of the Assignment-Engine 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 multiprocessing logic. As a base image for the Assignment-Engine, 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 [20] framework from Eclipse. The DAC was added according to the architecture right after REST Layer. It is able to invoke REST requests on its own, aggregating all required data from the BPMS or configured external data sources.

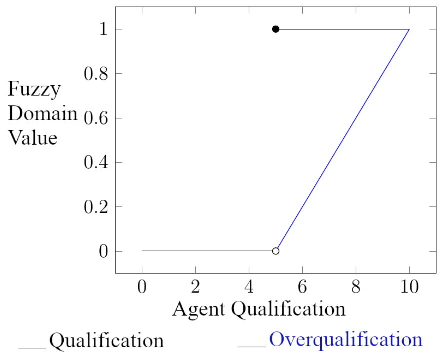

Figure 4. Fuzzy model showing fuzzy domain value vs. agent task qualification.

The fuzzy portions of the IAC were implemented using the fuzzylogic library for Python 3 [21]. As an example, Figure 4 shows the qualification and overqualification fuzzy model for the qualification values as introduced in Section IV. In this example, a qualification of 5 is required for the task. The black line represents the qualification while the blue line represents the overqualification. Below 5 the qualification value is 0, at 5 and above the qualification jumps to 1 and the overqualification starts to rise with a value of 0 by 5 and a maximum of 1 by 10. These two domains are calculated for all four qualification values and afterwards combined in a single domain. The code calculating these fuzzy domains can be seen in Figure 5.

As a BPMS for our prototype, Camunda [22] was chosen. 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.

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 following 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 Assignment-Engine. It is also possible to move this step to the DAC in the Assignment-Engine, in this case it would only be required to send the activity ID to the Assignment-Engine. While we implemented both, the sync and async variants, we focus on the asynchronous one as it provides more benefits, such as better multiprocessing support, and should be chosen if supported by the utilized BPMS. As soon as the assignment is calculated, the assignment scores are sent from the Assignment-Engine to the CamundaClient and the assignment in Camunda is handled via the client. Connected

```
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 5. Code snippet showing fuzzy implementation for qualification.

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 further can be reset to the agents' default value, e.g., on daily or weekly base as required by labor safety laws.

Alternatively, the IAC could be integrated in the Camunda-Client itself, removing the need for the additional REST-Requests between the client and the Assignment-Engine. The current approach however, allows the usage of the Assignment-Engine as a service for multiple BPMS simultaneously in a generic way.

Powerful multiprocessing 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 multiprocessing 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 Assignment-Engine itself, preventing slowdowns. If the request is too small for multiprocessing (the multiprocessing 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 multiprocessing 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 is separated in two parts. First the IAC is evaluated on their own regarding its speed, this part is further referred to as the performance evaluation. In the second part, an integration test is conducted comparing the activity assignment of a BPMS (Camunda) with the activity assignment of the Intelligent Assignment Component, referred to as the integration evaluation.

A. Performance Evaluation

The first test was conducted on a virtual server with 90GB 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, 10000 and 100000 agents getting assigned to a single activity. In the first group all agents were capable to perform 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 100000 times. The groups of 10 - 1000 agents were assigned using the IAC without multiprocessing while 10000 and 100000 agents were assigned using 17 processes ($n-1$ mode).

Figure 6 displays the assignment calculation performance if all supplied agents were capable, while Figure 7 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 multiprocessing 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 could mean that there are still some optimization problems in the elimination of incapable agents.

In general, the assignment of high volumes of agents provided no problems for the algorithms. As the IAC is meant

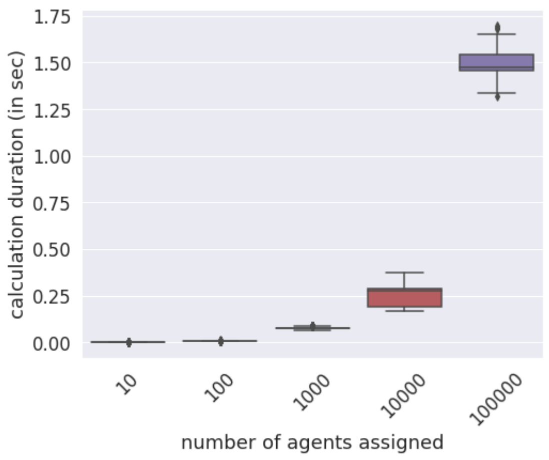

Figure 6. Calculation performance vs. number of capable agents assigned.

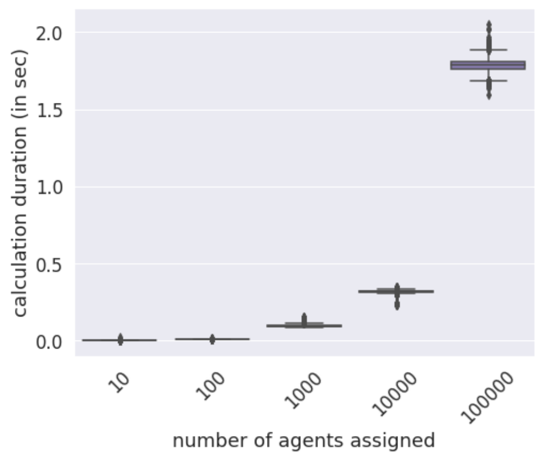

Figure 7. Calculation performance vs. number of capable and non-capable agents assigned.

to run behind SARs, rule engines or other performant basic filtering algorithms a load of 10000 possible agents for a single activity is further quite unlikely. The runtime in the sub seconds for agent values below 10000 would also allow the removal of preliminary filtering, reducing the runtime of the whole BPM process.

B. Integration Evaluation

The second test 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 Assignment-Engine 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 $21504 m^2$ and a total of 29 machines which required maintenance every 16 hours was created. The first

TABLE I. 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_overqualification (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 II. 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

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 machines 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 further 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.

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 Assignment-Engines 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 I shows a general comparison between the CMD and IA simulation, while II shows a more detailed comparison of internal and external worker stats in both simulations. In the following values from Table I will be compared with the more detailed values from Table II.

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 overqualification while using IA instead of plain Camunda can be explained with the low weighting of qualification in the algorithms as well as no presence of under-qualification in comparison to the CMD-Setup, where under-qualification was generally present. In Table II, the main source of overqualification 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 too high 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.

In summary, the IAC worked as expected with fast runtimes 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 many of them would most likely not fulfill the preconditions to be considered for the activity. However, if scaling is necessary, it can be readily achieved and works efficiently for at least 100000 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.

VII. Conclusion and Further Work

Industry 4.0 stands for highly automated production processes. However, these processes also rely on complicated tasks that can only be performed by humans manually. 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 BPM systems.

To counteract this, this paper 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. Therefore, we chose a set of important properties that incorporate aspects relevant in current Industry 4.0 production: achieve cost savings by incorporating not only underqualification but also overqualification and the separation between internal and expensive external workers; achieve a balanced workload for all workers to avoid idle times as well as overburdened workers; protect the workers from different hazards as enforced by legal 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 [23] and Camunda). The approach is built modularly and can be easily expanded. Further, the fuzzy weights used to calculate the assignment 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.

As future work, we plan to incorporate a more generic model where not only the weights are dynamic but also the criteria. Thus, the approach can be easily adapted for other domains and scenarios by extending or replacing the evaluation criteria. Other upcoming improvements could include utilizing transit path finding algorithms for the distance calculation 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 terms of number but also in terms of estimated time required to complete the activities. Finally, the slight performance decrease in the elimination of unqualified agents should be investigated.

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] 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.

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

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

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

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] H. A. Reijers, M. H. Jansen-Vullers, M. zur Muehlen, and W. Appl, "Workflow management systems + swarm intelligence = dynamic task assignment for emergency management applications," in *Business Process Management*, G. Alonso, P. Dadam, and M. Rosemann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 125–140.

[14] 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.

[15] 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.

[16] W. Kwak, Y. Shi, and K. Jung, "Human resource allocation in a cpa firm: A fuzzy set approach," *Review of Quantitative Finance and Accounting*, vol. 20, pp. 277–290, 2003.

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

[18] *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

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

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

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

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

[23] 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.