Adaptive web-based tools for capability-driven workforce management and task sequencing optimization

G. Albanis*, K. C. Apostolakis*, C. Öztürk‡, D. Zarpalas* and P. Daras*

* Information Technologies Institute, Centre for Research and Technology Hellas (CERTH), Thermi-Thessaloniki, Greece

‡ United Technologies Research Center Ireland, Cork City, Ireland

galbanis@iti.gr, kapostol@iti.gr, OzturkC@utrc.utc.com ,zarpalas@iti.gr, daras@iti.gr

May 24, 2018

ABSTRACT

Increasing demand for customized products due to customers' changing requirements has led to a non-standardized assembly operation even for the same product. In addition, assembly operations usually require more than one worker, each characterized by diverse capabilities and constraints. Industrial manufacturers need to ensure that both productivity, as well as workers' satisfaction remains in high levels. Toward this end, optimal sequencing of assembly operations can help minimize manufacturing lead-time and prevent excess fatigue for manual labor workers. Shop floor supervisors and managers should also be supported in efficiently allocating the available workforce to carry out these tasks in the most efficient manner, matching each individual's capacity to best fit the task requirements. In this paper we introduce a web based Decision Support System (DSS) for assigning workers to tasks and producing the optimal sequence for an air handling unit assembly line, leveraging values from real-life use cases most commonly associated with the Heating, Ventilation and Air Conditioning industry. The proposed solution showed in simulation to effectuate significant improvement in reducing unproductive time and workers' fatigue and due to its flexibility, task allocation plans can easily be adapted to fit changing requirements of both production parameters, as well as worker capabilities.

Keywords: resource assignment, Industry 4.0, smart manufacturing, optimization

BACKGROUND & INTRODUCTION

In practical assembly systems, in addition to the hard (i.e., technological) precedence relations, which enforce completion of certain set of predecessor assembly tasks before starting the successor ones, there are most likely soft precedence constraints and sequence dependent setup times in between. For example in car manufacturing, assembling a seat before the seat belt may extend the assembly time of the seat belt, because the seat is an obstacle which requires additional movements and/or prevents from using the most efficient installation procedure. Similarly, certain assembly tasks might require certain set of tools or re-positioning the assembly unit, setting up several parameters in the work center before starting the consequent task, which eventually extend the overall assembly time with these sequence dependent setup times. Hence, optimal sequencing of assembly operations is crucial in several ways: (1) minimizing overall assembly time and reducing manufacturing lead-time; (2) minimizing assembly worker fatigue due to many tool changes and setup times; and (3) eventually maximizing worker physical wellbeing [1]. Optimal sequencing of assembly operations is becoming especially crucial in manufacturing of big size units with mostly manual workers like in the Heating, Ventilation and Air Conditioning (HVAC) industry, in which various size of Air Handling Units (AHU) are being produced. Although they share some basic features, each AHU has unique properties regarding to the customer requirements. Therefore, technological precedence between assembly operations and required tools to perform them, mostly changes from one AHU to other. In addition, due to changing customer specs and assembly instructions, workers continuously need to check the Computer-Aided Design (CAD) documents. Hence, optimal sequencing of assembly operations to minimize the total assembly time and reduce the time wasted for tool changing between operations is important. In an ideal assembly plan, all operations which require the same tool, should be ordered consecutively so that workers do not need to waste time to walk between the AHU and the tool magazine to change tools. Meanwhile, fatigue of workers due to walking between AHU and tool magazine is reduced.

In this paper, a decision-support system (DSS) is presented in support of foremen and supervisors toward most efficiently managing the available workforce in an AHU assembly station. The station consists of a CAD drawing table, tool magazine and the AHU itself. Regarding to the complexity and the size of the AHU being produced, one or two workers can be assigned to perform the assembly operations. Experience of the workers in the assembly can vary between novice, intermediate or expert. While expert workers have full autonomy and can work all type of AHUs, other workers can work only certain type of AHUs and need to work with an expert worker for complex AHUs. Henceforth, in complex AHUs, matching certain assembly operations to specific workers is an important operational decision to take into account. Hence, in this study, our DSS is comprised of corresponding modules to: (1) optimally sequence

the assembly operations so that non-productive time and worker fatigue due to setup/tool-change time is minimized; and (2) optimally match assembly operations with the best workers and worker groups, taking into account the operations requirements and capabilities of the workers. A multi-agent adaptation approach is followed to ultimately propose the most efficient task allocation plan.

The remainder of this paper is organized as follows: In the *Related Work* Section we will present an overview of related work in dealing with multi-agent adaptation architectures in manufacturing. In the *Integrated DSS* Section, we will present the developed DSS and provide an overview of the modules and algorithms incorporated to address the AHU assembly station use case. In the *Application* Section, the DSS will be demonstrated in an industrial instance, which heavily borrows from a real-life HVAC industry case, to validate its efficiency, before finally drawing conclusions and discussing further work in the *Conclusion & Further Work* Section of this paper.

RELATED WORK

The State of the Art in manufacturing systems typically depicts the latter as a composition of multiple manufacturing machines connected by a common transfer system, partially processing raw material in multiple stages before the final output of the finished product. A clear distinction between Dedicated Manufacturing Lines (DML) and Flexible Manufacturing Systems (FMS) is evident. DMLs are characterized by high throughput, and the ability to maintain high production capacities (mass production), while FMSs are more tailored to the production of customizable product at significantly slower production rates. A hybrid approach, which borrows from both DML (throughput) and FMS (flexibility) systems, was proposed by Koren and Shpitalni [2] as Reconfigurable Assembly/Manufacturing Systems (RAS/RMS). Within RMS, configuration of the system impacts productivity, responsiveness, convertibility and scalability [2]. Recent research on technologies in the fields of Information and Communication Technologies (ICT), digitalization and virtualization have vastly contributed to the design of RMS [3] as well as the assembly of products, e.g. the Reconfigurable/Flexible Assembly Systems (RAS/FAS) “plug-and-produce” paradigm [4], and the development of various enablers currently on the market [5, 6].

Resource management and allocation is an important issue for industries that want to manage their time correctly and allocate their resources effectively and efficiently, without lowering productivity [7]. The process is extremely important in Dynamic Assembly Systems (DAS), i.e. modular factory platforms for light assembly, inspection, test, repairing and packing applications [8]. DAS combine flow-oriented dynamic production control and modular automation for increased production efficiency with ergonomic solutions for manual assembly. They usually include a range of standardized modules, such as

workstations, robot cells, conveyors, and flexible buffers. A relevantly new trend towards implementing DAS was first proposed by [9], in the scope of multi-agent architectures in which manufacturing resources are represented as agents. Each agent has a set of capabilities, and the combination of different agents' capabilities represent the full spectrum of capabilities offered by the system. In this context, "plug-and-produce", a unified term to describe interoperability within the scope of reconfigurable assembly systems carried out under the Industry 4.0 scheme in Germany as well as the Internet of Things (IoT)/Industrial Internet Consortium (IIC), is implemented by allowing easy resource integration and/or exclusion from the overall system without affecting its core functionality. In [10], this multi-agent architecture was extended to adapt to changes, such as product specifications and consequently dynamically synthesize a new configuration of low-level manufacturing operations. In [11] an approach was first described (Cyber-Physical Production Systems – CPPS) that transforms typical hierarchical infrastructure into hierarchical architecture of inter-communicating resources. By acclamation, this work is considered to usher in the 4th Industrial Revolution (Industry 4.0) [12]. Approaches have seen been described in [12] [13] that match assigned capabilities of manufacturing resources against product requirements via sets of matching rules. Fully automatic adaptation of manufacturing environments and production environments in which human operators validate the proposed configurations are addressed. However, both require a human expert to check the feasibility of the proposed solution as well as insert all process combinations needed to manufacture the product. Our approach combines sequencing optimization and capability-based matching to automatically derive feasibility and support shop floor supervisors. In [14] a skill-based approach is followed, using the same model for describing the product and the resources. A similar approach is described in [15], where product information is provided from an external process-planning tool. Both works focus on automation, while our approach is defined in a human assembly environment, and allows for smooth adaptation of properties and constraints in devices as well as workers' skills and preferences in the manufacturing plan.

INTEGRATED DSS

Considering Industry 4.0 demands in accordance to the requirements for adaptive user-machine interfaces and a human-centered factory, a web based DSS has been developed. Throughout the following Sections, a detailed overview is provided describing the subsystems comprising the Integrated DSS.

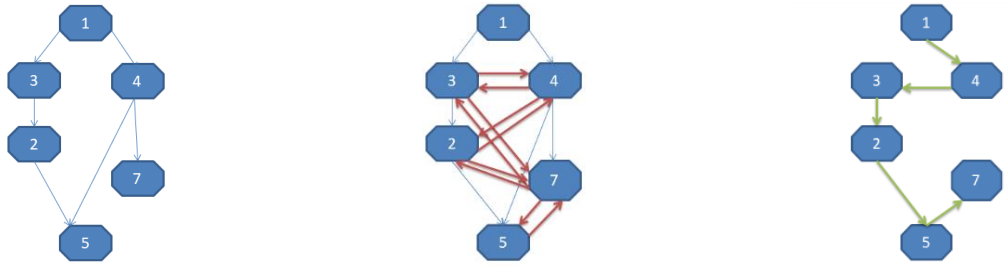
Capability Editor: A multi-agent adaptation architecture has been followed, relying on worker capabilities to match workers to specific manufacturing tasks. These capabilities are characterized by certain ranges and constraints referred to as the capability parameters [13]. In order to efficiently match the capabilities of the workforce with the capability requirements set by a new product, these functional

capabilities should be formalized in a common representation format. For this, we follow the approach described in [13], which utilizes a capability concept name and a number of parameters. We likewise employ a Capability Editor, a web-based interface module in which a supervisor is able to assign capabilities to workers, add, remove or change parameters of these capabilities or even add new capabilities and workers to a centralized repository. Eventually, through the Capability Editor, a user is able to add new workers to the factory's Resource Ontology, define and store new capabilities, and make them available to the other modules of the DSS.

Pre-process Plan Generator: In a similar way to the Capability Editor, a Pre-process Plan Generator (PPG) is used to define the product requirements from the product model, and describe a set of capability requirements per process in the form of a pre-process plan [16]. A pre-process plan is a way to describe how to manufacture a product through a step-by-step manner. A user is able to define the required capabilities for each task using the Capability Model previously defined through the Capability Editor. The pre-process plan can thus only contain capabilities already defined in the Capability Editor, and any new capabilities required must be defined and assigned to the workers using that module. The PPG gives foremen and supervisors the ability to define additional requirements, such as the number of workers needed for a task, or even the required tools for a task. The module is based on a graph editing user interface, its output viewed as an ordered graph of multiple sub tasks, which lay out all possible recipes (e.g. sequences) for manufacturing a product. The module also supports automatic generation of combined capability requirements. For instance, if a task requires more than one worker, the PPG will automatically create the necessary fields and new combined capability requirements will be created.

DSS Engine: As described in the Introduction section, a DSS component has been developed for finding the best sequence among all the possible sequences that can be derived from the output graph of the PPG, which minimizes the overall assembly time and sequence dependent setups. This DSS mostly targets industrial environments in which a computationally cheap approach is much appreciated for time and scalability requirements. Our approach is based on finding all the possible sequences of a pre-process plan (Figure 1a) by turning the graph into a Directed Acyclic Graph (DAG) [16]. The developed algorithm generates a graph structure by using the pre-process plan generator data, automatically filling up edges that don't exist and which would allow sequencing between tasks at the same level (Figure 1b). After that the algorithm computes all the possible permutations using topological search based on a combination of Kahn's algorithm for topological sorting [17] and backtracking. Executing Kahn's algorithm by reducing the pre-process plan into a DAG enforces the hard precedencies described in the Introduction section. In addition, a cost per path is calculated for each optimal sequence permutation, using the setup time for each tool, defined in the PPG. This will effectively group tasks using the same tools, which as mentioned

in the Introduction, is expected to reduce the number of visits/walking distance by sequencing tasks in a way that those which require the same tool are suggested sequentially (Figure 1c). After selecting one of the optimal sequences suggested by the Sequencing module, the Engine uses a Matching algorithm to match resources to specific tasks based on their capabilities, which bears similarities to the approach described in [13]:



(a) (b) (c)

Figure 1: Initial, intermediate and final stage of pre-process plan a) Pre-process plan as defined in the PPG; b) Graph structure with edges automatically added, displayed with red color; c) optimal sequence (one of many) forcing the process hierarchy.

Algorithm 1 Capability Coarse Matching

```
Input ← A list of resources considered for a process.
      The process data.
set resourcePool = []
for each required capability in process.required_capabilities do
        for each resource in resourcePool do
                for each capability in resource.capabilities do
                        if required_capability.lowercase() === capability.lowercase()
                                resourcePool.push(resource)
                end for
        end for
end for
Return resourcePool
Output: A list of resources with capabilities matching the capability requirement resourcePool.
```

Algorithm 1 details the process of Capability Coarse matching. It first goes through the list of required capabilities defined for a specific task and looks for matches in the list of capabilities for each worker in the currently considered resource pool. If a match is found, the worker can be considered in later stages of the reasoning process. If not, the worker is omitted.

Algorithm 2 Extract Rules from Process Parameters

```
Input ←The process data.
set rules = []
for each parameter in process.parameters do
        set rule {
                requirement = parameter
                condition = false
                confidence = 0.0
        }
end for
```

Return rules
Output: A list of rules against which to match resources.

Algorithm 2 demonstrates how Rules are defined from process parameters to drive the reasoning process. Each rule is characterized by a rule requirement (parameter to match against), condition (whether the rule is actually matched or not – expressed as a Boolean value) and confidence (the extent to which the rule is relevant expressed as 1.0 for fully relevant and 0.0 for fully irrelevant rules, in that the resource capabilities specifically designate variability among instances in handling the rule).

Algorithm 3 Reasoning
Input: A list of resources considered for a process.
A list of rules against which to match resources.

```
set resourcePool = []
for each resource in resources do
    set product = 0
    for each rule in rules do
        set param = lookFor_matching_parameter_name(rule.requirement.name)
        if param set rule.confidence = 1.0
        if check (rule.requirement,param) set rule.condition = true
        if (rule.confidence == 1 && rule.condition == false) product += -1000
        else product += ruled.condition * rule.confidence
    end for
if product > 0 push resource to resourcePool
else block (resource)
end for
Return resourcePool
```

Output: A list of resources matching the rule requirements resourcePool

Algorithm 3 shows how the Rules generated using Algorithm 2 drive the reasoning behind matching a resource from the resource pool satisfying the capability coarse matching (Algorithm 1) against the requirements set by the task defined in the PPG. It describes how Rule condition and confidence play a key part in determining whether the resource can be considered a candidate for ultimately assigning the task to a worker. Algorithm 4 below describes the operation of the Engine's Matching component, utilizing the aforementioned algorithms:

Algorithm 4 DSS Engine Resources to Task matching algorithm
Input: The ResourceOntology created using the Capability Editor.
The PreProcessPlan created using the Pre-process Plan Generator.

```
set resourcePool = ResourceOntology
for each process in the PreProcessPlan do
    if process has Sub-Processes && process has no LoA Profiles then continue
    if process is LoA Profile then continue
    resourcePool = Capability Coarse Matching(resourcePool, process)
    if resourcePool.length = 0 then continue
    for product-related and other parameter types in process do
        set R = Extract Rules From Process Parameters (process)
        if(R.length > 0) resourcePool = Reasoning (resourcePool,R)
```

```
            end for
        end for
        Return resourcePool
Output: A list of resources matching the rule requirements resourcePool
```

APPLICATION

This section presents a comprehensive description of the proposed DSS' application in a HVAC industry instance of AHU assembly. For this case study, 20 workers have been represented in the Resource Ontology using the Capability Editor. The core capability of interest to the use case consists of the workers' *Qualifications*, and its parameters and possible values are presented below.

- ***Role:*** *{Worker | Electrical worker}*
- ***Preferred Shift:*** *{Day | Night}*
- ***Experience Level:*** *{Novice | Experienced | Expert}*
- ***Availability:*** *{Available | Unavailable}*

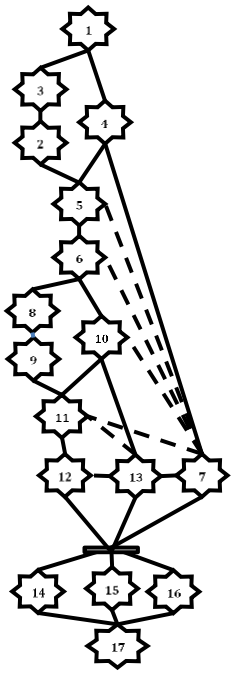


Figure 2: Use case process plan with 17 processes. Solid lines represent hard precedence relations, while dashed lines represent soft precedence constraints.

The process plan consisting of 17 sub tasks shown in Figure 2 is created with the use of the PPG. For each task the user can define its requirements, which are the number of workers, the various tools and all the parameters related to the Qualifications capability. As mentioned before, the PPG automatically generates combined capability requirements. Therefore, a task defined with a requirement of being assigned 2 workers, of which one should be a novice and the other an expert (with respect to *Experience Level*), automatically generates a requirement for assigning combination of workers, whose Qualifications group dynamic matches the "novice-expert" requirement. This will notify the Matching module to procedurally generate new worker resources by grouping single worker resources together to form groups matching the number in the task requirement. This will automatically reject all single worker resources during the matching stage later on, and pick only those groups that satisfy the requirement.

Once the pre-requisites for the DSS Engine have been defined, the latter finds the optimal sequences and can be issued by a supervisor to match the workers to specific tasks for any selected optimal sequence of operations. Table 1a presents results for one optimal case and with all the workers available. Note that Task with ID 17 has no requirements defined, and therefore all workers are equally capable for completing it alone. The DSS Engine visualizes the above results for each task, and logs the reasoning process as is shown in Figure 3.

Task ID	Resource IDs
1	11-17,11-20,3-11,17-4,4-3,4-20
3	12-13,12-14,12-15,12-16,12-1,5-12,7-12,9-12,9-2,13-2,14-2,15-2,16-2,2-1,5-2,7-2
2	12-13,12-14,12-15,12-16,12-1,5-12,7-12,9-12,9-2,13-2,14-2,15-2,16-2,2-1,5-2,7-2
4	11-17,11-20,3-11,17-4,4-3,4-20
5	17,20,3
6	11-17,11-20,3-11,17-4,4-3,4-20
8	11-17,11-20,3-11,17-4,4-3,4-20
9	11-17,11-20,3-11,17-4,4-3,4-20
10	11-17,11-20,3-11,17-4,4-3,4-20
11	11-17,11-20,3-11,17-4,4-3,4-20
12	11-17,11-20,3-11,17-4,4-3,4-20
7	11-17,11-20,3-11,17-4,4-3,4-20
13	11-17,11-20,3-11,17-4,4-3,4-20
14	11-17,11-20,3-11,17-4,4-3,4-20
15	11-17,11-20,3-11,17-4,4-3,4-20
16	11-17,11-20,3-11,17-4,4-3,4-20
17	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19,20

(a)

Task ID	Resource IDs
1	11-17,3-11,17-4,4-3,20-3,17-20
3	12-13,12-14,12-15,12-16,12-1,5-12,7-12,9-12,9-2,13-2,14-2,15-2,16-2,2-1,5-2,7-2
2	12-13,12-14,12-15,12-16,12-1,5-12,7-12,9-12,9-2,13-2,14-2,15-2,16-2,2-1,5-2,7-2
4	11-17,3-11,17-4,4-3,20-3,17-20
5	17,3
6	11-17,3-11,17-4,4-3,20-3,17-20
8	11-17,3-11,17-4,4-3,20-3,17-20
9	11-17,3-11,17-4,4-3,20-3,17-20
10	11-17,3-11,17-4,4-3,20-3,17-20
11	11-17,3-11,17-4,4-3,20-3,17-20
12	11-17,3-11,17-4,4-3,20-3,17-20
7	11-17,3-11,17-4,4-3,20-3,17-20
13	11-17,3-11,17-4,4-3,20-3,17-20
14	11-17,3-11,17-4,4-3,20-3,17-20
15	11-17,3-11,17-4,4-3,20-3,17-20
16	11-17,3-11,17-4,4-3,20-3,17-20
17	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19,20

(b)

Task ID	Resource IDs
1	11-17,3-11,17-4,4-3,20-3,17-20
3	12-13,12-14,12-15,12-16,12-1,5-12,7-12,9-12,9-2,13-2,14-2,15-2,16-2,2-1,5-2,7-2
2	12-13,12-14,12-15,12-16,12-1,5-12,7-12,9-12,9-2,13-2,14-2,15-2,16-2,2-1,5-2,7-2
4	11-17,3-11,17-4,4-3,20-3,17-20
5	11-20-3,11-20-17,11-4-3,11-4-17,17-20-4,20-3-4
6	11-17,3-11,17-4,4-3,20-3,17-20
8	11-17,3-11,17-4,4-3,20-3,17-20
9	11-17,3-11,17-4,4-3,20-3,17-20
10	11-17,3-11,17-4,4-3,20-3,17-20
11	11-17,3-11,17-4,4-3,20-3,17-20
12	11-17,3-11,17-4,4-3,20-3,17-20
7	11-17,3-11,17-4,4-3,20-3,17-20
13	11-17,3-11,17-4,4-3,20-3,17-20
14	11-17,3-11,17-4,4-3,20-3,17-20
15	11-17,3-11,17-4,4-3,20-3,17-20
16	11-17,3-11,17-4,4-3,20-3,17-20
17	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19,20

(c)

Table 1: DSS results: a) Optimal sequence task allocation plan; b) Adapted case after worker 20 "Experience Level" parameter is set to "expert"; c) Adapted case after Task 5 is modified to require three workers, with specific "Qualifications" capabilities.

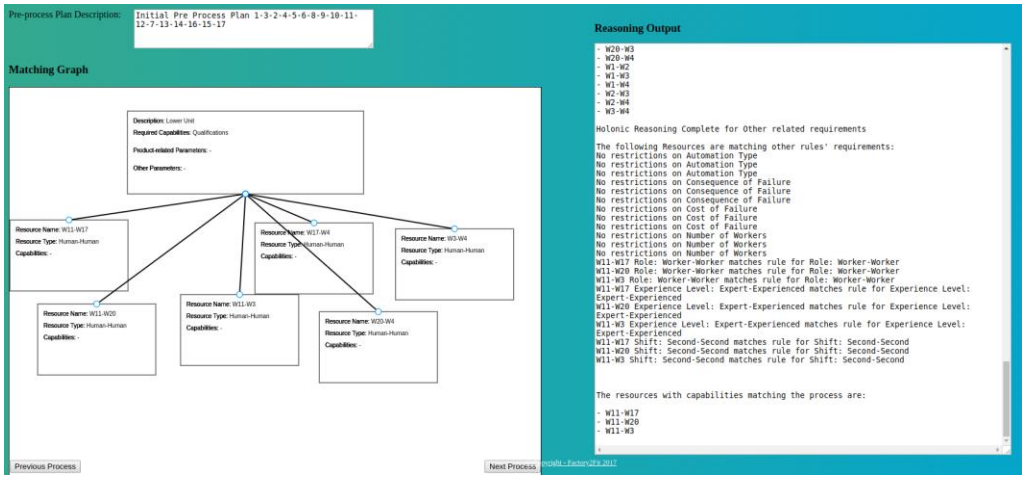


Figure 3: Visualization of matching and reasoning output

A powerful concept of the proposed implementation concerns re-adaptability. In case that some of the resources become unavailable, or some value defining their capabilities is changed over the course of time (e.g. workers assume different shifts, in our example), only one minor amendment should be made in the Capability Editor module, which will update the Repository with the new values. Similarly, considering the flexibility required in tailoring the production to a customizable product, a task's requirements can be changed upon request through the PPG instead of having to recreate a new pre-process plan from scratch. The DSS can store multiple pre-process plans in a centralized repository, providing the supervisors and foremen with an easy-to-use interface to make any minor or major changes over the course of production changes. The matching algorithm can then be executed again, using the DSS Engine, providing updated suggestions based on the changes made. Table 1b presents the results for the same sequence used in our application example, where a change is made to a resource description through the Capability Editor. Table 1c presents the results for changes made to the task requirements.

In order to prove our concept a simulation script has been developed for comparing the sequence which is currently used in an AHU assembly process and one of the produced optimal sequences. The total number of travels to the tool magazine and for viewing the CAD drawings is 7 in the optimal sequence and 9 in the corresponding default and as a consequence of this the total assembly time has been reduced by 9.87%. Prior to real-life deployment in near-operational environments, as a next step, we are planning to validate the results of our simulation using professional-grade 3D manufacturing simulation software.[1]

CONCLUSION & FURTHER WORK

In this paper, a DSS for optimally sequencing assembly operations and matching each with best eligible workers is presented. The developed solution is demonstrated via a representative use case from the HVAC industry, particularly AHU assembly. Deploying the proposed DSS technology, significant improvement in reducing non-value added time in AHU assembly due to setup/tool changing times and reducing total walking distance of workers is expected. The presented solution is highly flexible, in that it can be applied in different assembly systems of various industries. Furthermore, it is also modular: regarding to the necessity of the manufacturing system, only one or some of the components of the proposed DSS can be used. Besides advantages, our framework has also some limitations. At the moment real time information, such as unexpected events in the shop floor is not considered. Stochastic programming or simulation optimization can be integrated to cover real-time occurrences of unexpected events. Furthermore, albeit our framework generates process sequences automatically, it is still required to construct the initial process plan manually. A machine learning method that extracts ontology from

[1] https://www.visualcomponents.com/

technical data is a research direction to automate generation of the initial process plan. We are planning also to deploy the developed technology in different assembly systems, and develop mathematical and constraint programming based exact methods to evaluate the solution's quality and time performance of the DSS in hopes of improving the cases even further [18]. Finally, other combinatorial optimization methods to traverse the search space will be explored in combination with machine learning methods. A feature can be added to order possible groups for supervisors so that the list is ranked. The methodology can be coupled with task profiling and Levels of Automation (LoAs) in order to also suggest alternate ways of executing task parameters tailored to worker preferences [19], which is expected to significantly boost the attractiveness of the works as well as the satisfaction of the workers.

ACKNOWLEDGEMENT

The research leading to this work has received funding from the EU Horizon 2020 Framework Programme under grant agreement no. 723277 (Factory2Fit project).

REFERENCES

[1] Öztürk, C. (2013). Simultaneous balancing and scheduling of flexible mixed model assembly lines. Izmir: Dokuz Eylül University, The Graduate School of Natural and Applied Sciences.

[2] Koren, Y., & Shpitalni, M. (2010). Design of reconfigurable manufacturing systems. Journal of manufacturing systems, 29(4), 130-141.

[3] ElMaraghy, H., & ElMaraghy, W. (2016). Smart adaptable assembly systems. Procedia CIRP, 44, 4-13.

[4] Arai, T., Aiyama, Y., Maeda, Y., Sugi, M., & Ota, J. (2000). Agile assembly system by "plug and produce". CIRP Annals-Manufacturing Technology, 49(1), 1-4.

[5] Li, S., Wang, H., & Hu, S. J. (2013). Assembly system configuration design for a product family. In 41st North American Manufacturing Research Conference 2013, NAMRC 2013, June 10, 2013-June 14.

[6] Spath, D., Müller, R., & Reinhart, G. (Eds.). (2013). Zukunftsfähige Montagesysteme: wirtschaftlich, wandlungsfähig und rekonfigurierbar. Fraunhofer IRB Verlag.

[7] Tsourma, M., Zikos, S., Drosou, A., & Tzovaras, D. (2018, April). Online task distribution simulation in smart factories. In Small-scale Intelligent Manufacturing Systems (SIMS), 2018 2nd International Symposium on (pp. 1-6). IEEE.

[8] Leitão, P., & Karnouskos, S. (Eds.). (2015). Industrial Agents: Emerging Applications of Software Agents in Industry. Morgan Kaufmann.

[9] Barata, J., Camarinha-Matos, L., & Cândido, G. (2008). A multiagent-based control system applied to an educational shop floor. Robotics and Computer-Integrated Manufacturing, 24(5), 597-605.

[10] Alsafi, Y., & Vyatkin, V. (2010). Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. Robotics and Computer-Integrated Manufacturing, 26(4), 381-391.

[11] Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. Procedia Cirp, 17, 9-13.

[12] Antzoulatos, N., Castro, E., de Silva, L., Rocha, A. D., Ratchev, S., & Barata, J. (2017). A multi-agent framework for capability-based reconfiguration of industrial assembly systems. International Journal of Production Research, 55(10), 2950-2960.

[13] Järvenpää, E., Lanz, M., & Tuokko, R. (2016). Application of a capability-based adaptation methodology to a small-size production system. International Journal of Manufacturing Technology and Management, 30(1-2), 67-86.

[14] Backhaus, J., & Reinhart, G. (2017). Digital description of products, processes and resources for task-oriented programming of assembly systems. Journal of Intelligent Manufacturing, 28(8), 1787-1800.

[15] Järvenpää, E., Siltala, N., Hylli, O., & Lanz, M. (2018). Product Model ontology and its use in capability-based matchmaking. Procedia CIRP, 72, 1094-1099.

[16] Järvenpää, E., Luostarinen, P., Lanz, M., Garcia, F., & Tuokko, R. (2011, May). Dynamic operation environment—Towards intelligent adaptive production systems. In Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on (pp. 1-6). IEEE.

[17] Kahn, A. B. (1962). Topological sorting of large networks. Communications of the ACM, 5(11), 558-562.

[18] Öztürk, C., Tunalı, S., Hnich, B., & Örnek, A. (2015). Cyclic scheduling of flexible mixed model assembly lines with parallel stations. Journal of Manufacturing Systems, 36, 147-158.

[19] Chen, X., Bojko, M., Riedel, R., Apostolakis, K.C., Zarpalas, D. & Daras, P. (2018). Human-centred Adaptation and Task Distribution utilizing Levels of Automation. In 16th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2018) 11-13 June 2018, Bergamo, Italy.