UNIVERSITÀ DI TRENTO
Dipartimento di Ingegneria Industriale

eit Digital MASTER SCHOOL

UNIVERSITÉ CÔTE D'AZUR

Master's Degree in

Mechatronics Engineering

FINAL DISSERTATION

CLOUD WORKFLOW CONTROLLER FOR IIOT SOLUTIONS IN ADDITIVE MANUFACTURING

In the context of the POLYLINE project

Supervisor UniTrento
Daniele Fontanelli

Supervisor Company
Simone Dal Poz

Student
Zaida Brito Triana, [MAT. 230400]

ACADEMIC YEAR 2021/2022

Nesting /Slicing

3D Data

MES
Scheduling

ERP
Order Management

Monitoring

Laser Sintering

Cooling /Unpacking

Powder Preparation /Recycling

Cleaning

Buffer

Dyeing /Finishing

POLYLINE Project

Automated Laser Sintering Production Line

Graphics by 3YOURMIND,
based on concept by G. Katsimitsoulias, Fraunhofer IML

Abstract

This master thesis presents the research and implementation of a cloud workflow controller for Industrial Internet of Things (IIoT) solutions in Additive Manufacturing (AM). This thesis has been carried out in the context of the POLYLINE project, which aims to automate the entire manufacturing process in the production of polymeric components in additive manufacturing plant of BMW in Munich.

This research is centered around exploring potential models of computation that can be leveraged to construct a logic-based model of the production line designed in the project, in order to do so, the research presented in this thesis investigates some of the most commonly used models for these purposes and examines the advantages that Behavior Trees (BTs) have offered for similar projects to this one in the fields of robotics, manufacturing, and gaming. Specifically, the research delves into the ways in which BTs demonstrate potential for effectively organizing and managing the complex processes and decision-making situations involved in the operation of a production line. Via a comprehensive examination of the capabilities and limitations of BTs and other relevant models, this thesis aims to provide insight into the most suitable approach for constructing a logic-based model of the production line in question.

Furthermore, this research underwent a comprehensive study of the principal communication protocols implemented in the Industrial Internet of Things. This evaluation was performed in order to provide a more comprehensive assessment of the proposed connections for the successful implementation of the workflow and give a comprehensive understanding of the underlying automation functionality intrinsic to the project. While the primary protocol implemented in the project was based on the OPC UA protocol, a comparison with all the current state-of-the-art developments in communication protocols in this field was also undertaken. The results of this extensive analysis provided valuable insight into the most suitable communication protocols for future research.

Additionally, a thorough data analysis was conducted to gain a comprehensive understanding of the initial performance of the implementation site. This analysis aimed to identify key performance indicators (KPIs) and determine their initial values. These analytics proved to be a valuable resource in comprehending the performance of whole

system, thus enabling us to make informed decisions regarding the most appropriate approach for future development.

The workflow controller has been tested in a mock environment prior to its deployment in the BMW plant. The tests have been focused on the well development and safety of the workflow and the ability of the system to support the workload generated by the production process. The results have shown that the workflow controller is able to support the workload generated by the production process, ensuring a smooth transition between the different stages of the production line.

This Master's thesis presents a comprehensive overview of the current state of research in the field of industrial automation, including a novel cloud workflow controller design and its capabilities. The research contained within this thesis represents a significant contribution to the field, providing valuable insights and advancements in the understanding and application of IIoT in additive manufacturing.

Contents

List of figures

Keywords

Additive Manufacturing (AM)
Automated Guided Vehicles (AGVs)
Behavior Tree (BT)
Cloud Manufacturing (CM)
Finite State Machine (FSM)
Flexible manufacturing systems (FSM)
Enterprise Resource Planning (ERP)
Hierarchical Finite State Machine (HFSM)
Industry 4.0
Industrial Internet of Things (IIOT)
Industrial Message Queuing Telemetry Transport (MQTT)
Key Performance Indicators (KPIs)
Machine-to-machine (M2M)
Manufacturing Execution System (MES)
Non-player Character (NPC)
OpenAPI Specification (OAS)
Open Platform Communications United Architecture (OPC UA)
Probabilistic Finite State Machine (PFSM)
Protocol Buffers (Protobufs)
Representational State Transfer (REST)
Universal Machine Type Interface (UMATI)

1 Introduction

In the years to come, the number of additively manufactured products and parts will continue to grow, AM technologies are becoming increasingly important and have the potential to shape the world of production in the future (Horstkotte et al., 2021). Research into the optimal implementation of additive manufacturing technology has the ability to improve the manufacturing processes that run the general production globally, enabling more local production and reducing transport costs (Thomas & Gilbert, 2014), as well as allowing for greater adaptability for OEMs and manufacturers (Bhatt et al., 2020).

Additive manufacturing integration into conventional lines can only be achieved to a limited extent at present, both vertically and horizontally ("Horizontal and vertical integration in industry 4.0," 2019) (Butt, 2020). Initially, this is attributed to the specific AM production steps (e.g., curing, post-processing, slicing, batch production time), but a closer examination reveals it's related to the overall low implementation of automation techniques in AM manufacturing and post-production processes. (Butt, 2020). Due to this, production intervals are elongated, and there is a great need for manual work involved not only to mass-produce but to prototype as well. The discontinuity of the digital data chain along the horizontal process chain results in a lack of comprehensive monitoring along the processes, hindering integration into relevant production controls. This impediment significantly reduces the full potential of additive manufacturing processes in the integration into existing series production and assembly lines. The state of the art in the industry shows that there is not enough automation in the additive process chain to fully exploit its potential, as mentioned in (Horstkotte et al., 2021) in the same remote technologies to enable automatization are still yet to be implemented in many industries remains restricted in a general sense, as evidenced by the findings of the study conducted by Seiffert et al. (2022).

Until now, the research and implementation of digital solutions for managing production lines in Additive Manufacturing have been quite limited (Mies et al., 2016). The integration and collaboration with other technologies needed, such as autonomous robots, cloud control, and integration, are shown to be key for the optimal implementation of Industry 4.0; the path to smart and flexible manufacturing goes through the development and integration with automation (Butt, 2020). The ongoing process of globalization faces the challenge of meeting the increasing worldwide demand for capital and consumer goods while ensuring sustainable development across social, environmental, and economic dimensions. To address this challenge, industrial value

creation must prioritize sustainability. The current landscape presents numerous opportunities for Additive Manufacturing, as studied in (Stock & Seliger, 2016).

The rise of Industry 4.0 and the increasing popularity of emerging technologies has created a unique space for the transformation of traditional manufacturing methods. (Gao et al., 2019). The current industrial value creation in early industrialized countries is influenced by the development towards Industry 4.0, the fourth stage of industrialization. This development presents substantial opportunities for achieving sustainable manufacturing. This paper will conduct a state-of-the-art review of Industry 4.0, based on recent advancements in research and practice, and provide an overview of the various opportunities for sustainable manufacturing in Industry 4.0. The prediction of Industry 4.0 has opened up opportunities for developing roadmaps for manufacturing operations, especially for manufacturing IT systems. The traditional centralized and monolithic production monitoring and control applications will be replaced by solutions that support the decentralized and connected vision of production and supply chain processes (Almada-Lobo, 2016).

It is worth noting as well that the COVID-19 pandemic situation brought attention to the importance of reacting quickly and efficiently to real-time changes as the traditional manufacturing methods are getting behind in their adaptability horizons (Petch, 2020). Overall, the trend is clear: it is very important to adopt efficient, flexible, and agile manufacturing systems to overcome the challenges of real-time changes and volatility in demand. The adoption of advanced technologies foreseen in Industry 4.0, such as automatization, data management, and the Internet of Things, will help to improve manufacturing efficiency and reduce lead time.

1.1 What is the polyline project?

The POLYLINE project is a collaboration of German industry and research partners working to establish an automated factory for additive manufacturing of automotive parts. The objective is to fully automate the BMW AM production line, encompassing every aspect from CAD model to 3D printing and quality assurance. 3YOURMIND's Agile Manufacturing Suite will manage the entire AM workflow, and the resulting factory will serve as an example of Industry 4.0 for other future implementations. The goal of the project is to create an efficient and streamlined production line for end-use parts using laser sintering technology. Each partner will handle a specific aspect of the manufacturing process, from 3D printing to post-production and quality control. The project also focuses on integrating the entire

additive production process into existing mass production. The project also aims to address issues such as long processing times and low degrees of automation for physical handling and transport processes by implementing a digital and continuous data chain along the entire production process.

1.2 Goals of this thesis

The main goal of this research work is to create a cloud workflow controller within the POLYLINE project, which aims at the development of a fully digitalized additive manufacturing production line at BMW Germany. 3YOURMIND, one of the companies involved in this research project, is in charge of implementing the manage the entire AM workflow composed by the MES(Manufacturing Execution System) and the ERP(Enterprise Resource Planning) mainly, highlighted in Error: Reference source not found, and has developed and commercialized a solution called 3D Pulse Control (3YOURMIND, 2023) for monitoring additive manufacturing production through IIOT. The objective is to expand this solution's capacity to not only monitor the machines but also to remotely control multiple machines, potentially in different sites.

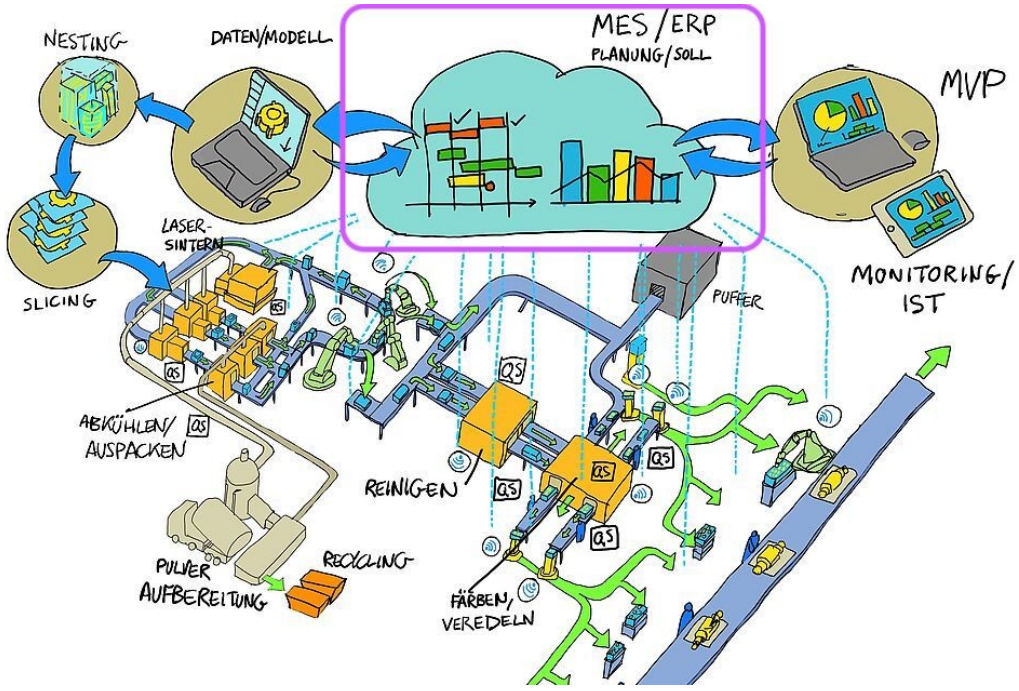


Figure 1. Schematic representation of an example line production of the laser-sintering process, highlighting the Manufacturing Execution System and the Enterprise Resource Planning, center focus on the thesis. © G. Katsimitsoulias, Fraunhofer IML (edited)

In order to achieve this goal, it is necessary to conduct research on the technologies needed to develop an automated production line. The main focus of this research will be the workflow trigger utility and the workflow controller development, as well as the research on the necessary models of computation. This thesis aims to explore potential models of computation that can be leveraged to construct a logic-based model of the production line outlined in the project, specifically investigating the advantages that Behavior Trees (BTs) offer for projects similar to this one in the fields of robotics, manufacturing, and gaming. Through a thorough analysis of the capabilities and limitations of BTs and other relevant models, this thesis aims to provide insight into the most suitable approach for constructing a logic-based model of the production line in question.

In the second segment of this dissertation, an in-depth examination will be carried out to assess the communication protocols utilized in the Industrial Internet of Things (IIOT) in order to scrutinize the proposed connections for completing the workflow. The principal protocol implemented in the research will be based on the OPC UA protocol; however, a comparative analysis with the current state-of-the-art advancements in this domain will also be executed. Furthermore, experiments designed to measure performance will be conducted in order to determine the optimal protocol for future research within the project. The outcomes of this comprehensive analysis will furnish invaluable insight into the most appropriate communication protocols for future research.

In order to evaluate the performance of the workflow and determine the system's capacity to sustain the workload generated by the production process, a mock environment will be created for this purpose. The workflow controller will undergo testing in this simulated environment prior to its deployment in the BMW plant. The results of these tests will demonstrate that the workflow controller possesses the capability to support the workload generated by the production process, thereby ensuring a seamless transition between the various stages of the production line.

Furthermore, a comprehensive data analysis was performed with the goal of conducting analysis with the available data to gain a deep understanding of the initial performance of the implementation site. The objective of this analysis was to recognize key performance indicators (KPIs) and establish their initial values. These analytics proved to be a useful tool in comprehending the overall performance of the system, thereby allowing informed decisions to be made regarding the best approach for future development.

The overarching objective of this thesis is to give an overview of the state of the art in Industry 4.0, with a specific focus on the domain of industrial automation and additive manufacturing. The goal is to conduct a comprehensive analysis of the current advancements and developments in this field, in order to provide a clear understanding of the state of the art. The research aims to provide valuable insights and information that can be used to guide future development and innovation in the field of industrial automation and additive manufacturing.

2 Literature review/theory

This section aims to present a comprehensive examination of the current state-of-the-art in the automation of additive manufacturing. The section will begin by highlighting the current challenges and possibilities in the field, followed by an in-depth analysis of the theoretical principles and concepts of behavior trees and a comparison with other commonly used methodologies in the realm of manufacturing automation. Furthermore, the current state-of-the-art of behavior trees in similar applications will also be examined. Finally, the section will conclude with a discussion of the most prevalent communication protocols currently utilized within the Industrial Internet of Things (IIoT) and their current state of advancement. In sum, this section aims to provide a thorough understanding of the current state-of-the-art in the field of additive manufacturing automation and related areas, with a specific focus on the utilization of behavior trees and communication protocols within the IIoT. This serves as the theoretical foundation upon which this research is based.

2.1 Automation in Additive Manufacturing

Additive Manufacturing is a process of building 3D objects by adding layer-upon-layer of material until the desired shape is achieved. The material can be plastic, metal, or other composites, and the process is controlled by computer software. AM processes refer to techniques that build up a surface by adding material, resulting in varying part quality, density, and geometric accuracy (Ibrahim et al., 2018), (Nematollahi et al., 2019). AM includes various technologies such as 3D printing, laser sintering, and fused deposition modeling. It differs from traditional subtractive manufacturing methods, which involve cutting and shaping material from a larger block, in that technologies like CNC there are some steps in automation mainly under Flexible manufacturing systems (FSM)(Koštál et al.,2019). AM provides greater design freedom, improved efficiency, and increased sustainability compared to traditional methods.

It is crucial to analyze the advancements in automation within additive manufacturing in relation to the broader trend of Industry 4.0, as the two are closely interrelated. The evolution of industrialization has established the foundation for the emergence of the fourth industrial revolution, commonly referred to as Industry 4.0. First introduced in 2011 with the goal of promoting economic growth in Germany (Industrie 4.0 - BMBF, 2011), this industry is characterized by the integration of cyber-physical systems (CPS). These systems, which constitute the third generation of

control systems, consist of embedded computers and complex software applications that are interconnected through wired and wireless connections. These CPS possess the capability for autonomous decision-making and communication, with the purpose of improving industrial efficiency, productivity, safety, and transparency. There is a significant correlation between the concept of Industry 4.0 developed in Germany and the Industrial Internet concept. The term "Internet of Things" was first introduced in 1999 and encompasses interconnected devices present in various settings such as homes, businesses, and industries. Empirical studies have demonstrated that the implementation of wireless communication network intelligent automatic 3D printing machinery increases printing efficiency by 15% and saves economic cost by 20%, highlighting the practicality of the research findings (Zhang et al., 2021). Additionally, there have been noteworthy advancements in the field of additive manufacturing, such as the incorporation of machine learning, as demonstrated in (Delli & Chang, 2018).

The concept of Industry 4.0, also known as the fourth industrial revolution, is a technology-driven shift in manufacturing that aims to improve various aspects of human life. It involves the integration of smart automation, decision-making, knowledge, problem-solving, self-diagnosis, self-configuration, and self-automation in industries (Shah, 2022). This technology has the potential to influence various levels of the manufacturing process, the end-users, designers, managers, and all employees involved in the manufacturing process and supply chains. This work describes the use of a decision tree algorithm for monitoring energy consumption in machines and appliances, predicting future behavior, and detecting anomalous behavior. The system is evaluated and compared with existing methodologies, and it offers an efficiency of 78%. However, the implementation of Industry 4.0 also faces standardization issues, security issues, resource planning challenges, legal issues, and changes in business paradigms. The success or failure of Industry 4.0 depends on the entire production chain and all the participants, from manufacturers to end-users.

As seen in this article (Kellett, 2012) advancements in automation technologies such as motion control components, robotic arms, and gantry robots will enable 3D printers to overcome size constraints and increase the types of objects that can be "printed", and that new market opportunities will open up for manufacturers of these technologies. Studies have been conducted exploring the acceleration of production processes, with a particular emphasis on cloud-based additive manufacturing as a means of facilitating rapid development. (Wang et al., 2019). Recently, there has been an increase in collaboration between robotics and additive manufacturing for the purpose of enhancing production (Bhatt et al., 2020). Additionally, additive manufacturing has opened up promising sustainable prospects, as demonstrated by (Stock & Seliger, 2016).

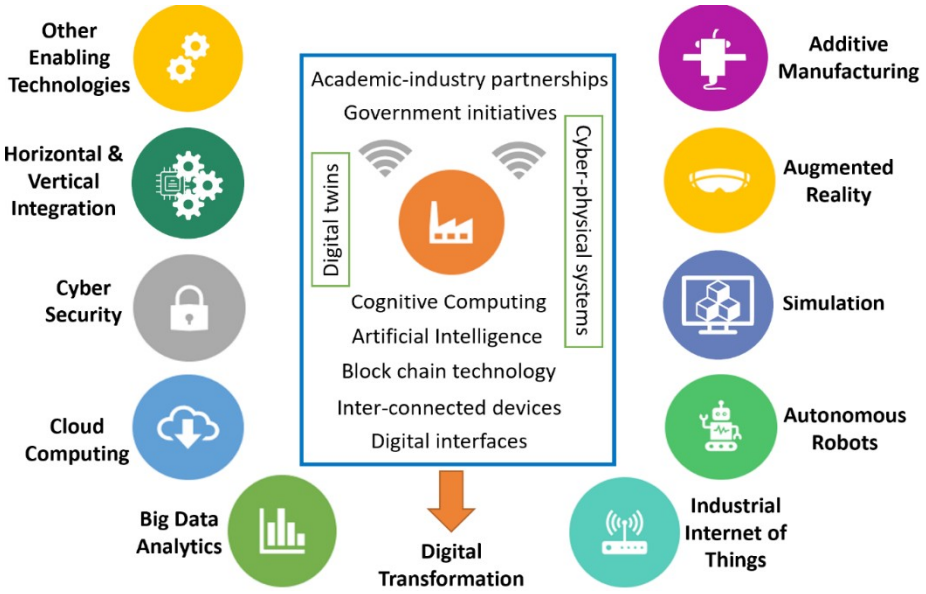


Figure 2. Pillars of Industry 4.0, retrieved from (Butt, 2020)

There is limited research available on the automation of additive manufacturing facilities, however, this study from 2015 (Freens et al., 2015) is highly relevant to the focus of the present investigation. This study aimed to demonstrate the feasibility of automating the production planning of a 3D printing factory by modeling it as an optimization problem and testing it in a Shapeways factory. The production planning process involves composing printer trays by allocating objects to batches and determining their positions in the tray using a 3D packing algorithm. The study proposed an extension of the classical one-dimensional bin packing problem that takes into account specific requirements of high-volume 3D printing environments. The method was tested in a simulation study using one month of production data from Shapeways, which showed an increase in printer capacity and shorter waiting times. The main drawback of the method is difficulty in accurately estimating the print time of batches. Future work should focus on designing a more elaborate simulation study to optimize object size mixtures for more accurate print time estimation.

2.2 Behavior Trees

Behavior Trees (BTs) are a hierarchical decision-making system that provides a structured approach to designing and implementing autonomous systems in computer programming and robotics. The system was created with the intention of providing a method for describing the behavior of non-player characters (NPCs) in computer games. These NPCs need to respond to complex and uncertain situations in real-time, making decisions based on a set of rules and conditions.

The concept of BTs was first introduced by R. G. Dromey in 2001, with the publication of his paper defining the key concepts and applications of the system. However, one of the first significant implementations of BTs was in the development of the Halo 2 and Halo 3 video games (Isla, 2005, Isla, 2008). This demonstrated the potential of BTs in creating intelligent, autonomous systems that can react to changing environments.

Recently, the use of BTs in the robotics and control community has gained significant attention, as seen in the publication of research papers and books dedicated to the topic. Two Ph.D. theses, "Onboard Evolution of Human-Understandable Behaviour Trees for Robot Swarms" by Jones.SW and "Behavior Trees in Robotics" by Colledanchise. M, will be used as the basis for discussion and analysis in this research paper. The latter thesis served as a precursor to the highly regarded book "Behavior Trees in Robotics and AI," which will also be referenced in this research.

The example presented in this research paper was implemented using the "BehaviorTree. CPP" framework (Faconti, 2022), written in C++ 17, and the "Groot" editor (Faconti, 2019). Both of these programs are maintained and regularly updated by Faconti.D and have been instrumental in the implementation of BTs in this field. They provide a powerful toolset for creating and visualizing behavior trees, making the process of designing and implementing autonomous systems more efficient and user-friendly.

2.2.1 Behavior Tree theory

Behavior trees are a type of decision tree used in artificial intelligence and robotics to specify the behavior of a system. They are composed of a hierarchy of nodes, each of which represents a specific behavior or action. The tree is traversed from the top down, and the actions of the nodes are executed in order based on the results of previous actions and the conditions specified in the tree.

Behavior trees are often used in robotics because they allow the robot to make decisions based on its current situation and the available information. For example, like

in the POLYLINE project, a behavior tree might specify that a robot should pick a printed component if there is one available, but if the robot encounters that the piece is stuck, it should try to go around the obstacle or seek help from a human operator.

Behavior trees are a useful tool for specifying complex behaviors because they allow for a clear and concise representation of the decision-making process and enable easy modification of the behavior by modifying the tree structure. They are also easy to understand and debug, making them a feasible choice for specifying the behavior of robots and other intelligent systems

Corresponding to the matter in this research, we can define that a Behavior Tree is a hierarchical tree of nodes, where the leaf nodes correspond to executable programs that correspond to robot actions, such as picking up an object, stop, or perform a gesture. A behavior tree (BT) begins by running its root node, which sends out signals at a specific frequency called ticks. These ticks are then passed on to the root node's children. A node can only be executed if it receives these ticks. When the root of a tree sends a tick signal, it travels towards the leaves of the tree. Each tree node that receives the signal will then execute a callback function. This callback can return one of three possible results: success, failure, or running. If the running result is returned, it indicates that the action is still in progress and needs more time to be completed.

Tree nodes with children are responsible for passing on the tick signal to them. The rules for when and how many times the tick is passed down to the children may vary between tree nodes. The actual commands of the behavior tree are found at the leaf nodes, and it is there that the tree interacts with the rest of the system. Commonly, these leaf nodes are Actions nodes.

2.2.1.1 Common node types

A review of the most commonly used node types in industry projects and research is presented below.

Outline of the most common control flow nodes:

Sequence nodes: A sequence node is the most common type of node in a behavior tree. It is used to execute a series of actions consecutively. When a sequence node with more than one child receives a tick, it will first tick its first child; if that child returns a result of success, the sequence node will then tick its next child, continuing this process until all of its children have been ticked.

The sequence node will return a result of success if all of its children return a result of success. If any of the children return a result of failure, the sequence node will immediately return a result of failure and stop executing the behavior tree. Above, the pseudo code representation that elucidates the underlying logic of the node. This pseudo code serves as a guide for understanding the functionalities and decisions made by the node in executing its tasks.

Algorithm 1 Sequence node

```
for i ← 1 to N_child do
    childReturn ← Tick(C_i)
    if childReturn = RUNNING then
        return RUNNING
        if childReturn = FAILURE then
            return FAILURE
        end if
    end if
end for
return SUCCESS
```

Fallback or Selector nodes: The series of actions specified by these nodes is ticked in order. If a child returns failure, it ticks the next one. If success or running is returned by one child, the selector does not tick any more of its children, and the same result is returned. In the event that all of the actions fail, a failure result is returned by the selector node. Above, the pseudo code representation that elucidates the underlying logic of the node.

Algorithm 2 Selector or Fallback node

```
for i ← 1 to N_child do
    childReturn ← Tick(C_i)
    if childReturn = RUNNING then
        return RUNNING
        if childReturn = SUCCESS then
            return SUCCESS
        end if
    end if
end for
return FAILURE
```

Parallel nodes: These nodes indicate that a series of actions should be carried out at the same time. If any of the child nodes are still in progress, the node will return a running status. If the number of child nodes that have successfully completed is above a certain threshold, which is set by the user, the node will return a success status. If it is below the threshold, it will return a failure status. Although they were not used in this research, parallel nodes are mentioned here for completeness.

Decorator nodes: These nodes have a single child node, and they modify the return status of the child node according to a rule specified by the user. They also selectively execute the child node based on a predetermined rule. In every case, if the child node returns a running status, the decorator node will also return running. Algorithm 3 presents the pseudocode of the commonly used decorator node *RetryUntilSuccesful.*

Outline of the general characteristics of control execution nodes:

Algorithm 3 Decorator node
RetryUntilSuccessful with N attempts

```
1: for i ← 1 to N_child do
2:     for n ≤ N do
3:         if childReturn = FAILURE then
4:             n = n + 1
5:             if childReturn = SUCCESS then
6:                 n = N
7:                 return SUCCESS
8:             end if
9:         end if
10:    end for
11: end for
12: return FAILURE
```

Action nodes: These nodes are the most common type of leaf node, and they represent a specific action or behavior that should be performed. When it receives a tick, an action node will execute a command. If the action is completed successfully, it will return a success status. If the action fails, it will return a failure status. While the action is in progress, it will return a running status. An action node may read from the blackboard (as described 3.4 Section) and modify the values of certain variables by writing to them.

Condition nodes: Control execution nodes determine the success or failure of a particular action based on the evaluation of a specific condition. They are frequently used with decorator nodes to specify the circumstances under which a specific behavior should be executed. It is important to note that a control execution node will never return a running status. They may read from the blackboard, but they cannot modify its contents.

The table above, inspired by (Colledanchise & Ögren, 2017), summarizes the characteristics of the most common node types

Node type	Symbol	RUNNING	SUCCESS	FAILURE
Sequence	→	If one child returns RUNNING	If **all** children return SUCCESS	I **one** child returns FAILURE

Fallback		If one child returns RUNNING	If **one** child return SUCCESS	If **all** child returns FAILURE
Parallel		Any other situation	If ≥ N children return SUCCESS	If > n - N return FAILURE
Decorator		Custom	Custom	Custom
Action		During completion	On conclusion	If unfeasible
Condition		Never	If true	If false

Table 1. Most frequently encountered node types in a Behavior Tree, outlining their main characteristics

2.2.1.2 Blackboard

In a behavior tree, leaf nodes interact with the environment through a set of variables known as the blackboard. The blackboard mechanism facilitates the integration of events by allocating memory for specific tasks or by translating events into lower-level mission plan execution status. These statuses can be stored in the blackboard and reset after processing appropriate behavior tree tasks. (Agis et al., 2020)

In the context of behavior trees, a blackboard is a shared memory space that is used to store information that is relevant to the behavior tree and its nodes. The blackboard is accessible to all nodes in the behavior tree and allows them to share and access information in a centralized manner. This can be used to facilitate communication between different nodes, and to store information such as the current state of the system, the status of various tasks, or the results of sensor data.

The blackboard typically contains several different data structures, such as variables, flags, and data tables, that are used to store information in a structured manner. The blackboard can also include methods for reading, writing, and modifying the data stored within it. This allows nodes to access and manipulate the data in a consistent and predictable way.

Blackboards are useful in behavior trees because they allow the nodes to share information and coordinate their actions without the need for direct communication. This makes it possible to decouple the logic of different nodes and to create a more modular and reusable behavior tree. It also allows the behavior tree to adapt to changing circumstances by modifying the data stored in the blackboard, rather than by modifying the structure of the behavior tree itself.

In the shapes project (Cooper & Lemaignan, 2022), we can see a basic example of the use of blackboard when the robot outputs a string with the name of the identified person after recognizing it.

Blackboards are one of the key elements in the real implementation of behavior trees; it is possible to see it for example in (Jones, 2020) in the explanation of the controller of its system "A behavior tree consists of a tree of nodes and a blackboard of variables which comprise the interface between the controller and the robot. At every controller update cycle, the tree of each robot is evaluated, with sensory inputs resulting in actuation outputs. Evaluation consists of a depth-first traversal of the tree until certain conditions are met. Each agent has its own blackboard, state memory and tree."

2.2.1.3 Challenges with Behavior Trees

Implementing Behavior Trees can present certain challenges, particularly with single-threaded programming. However, once built, the engines can be reused with software libraries. Is also important to mention that BTs are designed around conditions, not states, and the switching occurs between tasks on each tick rather than events. Despite the availability of BT development tools, they are not as advanced as those for finite state machines (FSMs) and in the same line the software for BTs is less sophisticated compared to FSM software. Some of the limitations of BTs include difficulties in evaluating large trees and a lack of decision-making models. However, newer techniques like Utility AIs hold potential for creating advanced artificial intelligence in future applications.

2.2.2 Behavior Trees vs Finite State Machines

FSMs have been for long time the standard choice for constructing structures for task switching, in 2013, Klockner highlighted the current prevalent use of Finite State Machines as a means of managing the diverse capabilities of an Unmanned Aerial Vehicle (UAV), quoting in his paper "The state-of-the-art approach for managing different capabilities of a UAV is to use Finite State Machines (FSMs)." (Klockner, 2013)

In a comprehensive manner, FSMs are a computation model that can be in only one state at any one time. The state changes when specific conditions are met, which

are usually linked to sensor inputs. The outputs of actuators are usually dependent on the current state.

Despite the multitude of possibilities that finite state machines offer, it has been noted that they encounter difficulties in satisfying two essential characteristics that are often sought-after in autonomous agents: the ability to exhibit both reactive and modular capabilities.

The term reactive refers to the ability to rapidly and effectively respond to changing circumstances. This can manifest in various ways as for example shown in the demo developed in this paper, when a robot reroutes to avoid a collision. The modularity allows for a more hierarchical and adaptable approach in designing, which can lead to more flexibility and ease of modification of the agent's behavior.

As explained in (Colledanchise & Ögren, 2017), the problem lies in FSM's trade-off between reactivity and modularity. This compromise can be comprehended by referencing the traditional Goto statement, heavily criticized by Edsger Dijkstra (Dijkstra, 1968), that was utilized in early programming languages. The Goto statement serves as an exemplar of a one-way control transfer, in which the execution of a program diverts to another section of code and continues from that point. In contrast to one-way control transfers, contemporary programming languages tend to favor two-way control transfers, as exhibited in constructs such as function calls. In order for the system to be reactive, many transitions between components are necessary. However, a high frequency of transitions results in an abundance of one-way control transfers. For instance, if a component is removed, every transition to that component must be updated, affecting lastly the modularity. BTs, on the contrary, utilize two-way control transfers regulated by the internal nodes of the tree, which has been shown to mitigate this trade-off.

As the number of capabilities or inputs that a finite state machine is required to process increases, the complexity of the FSM also increases in terms of the number of states and transitions. This results in a scalability issue, as the FSM becomes increasingly difficult to design, analyze, understand, maintain, and verify as the number of states and transitions increases, making it challenging to ensure that the FSM is working as intended.

It is also worth noting that behavior trees offer advantages over finite state machines in terms of reusability. The hierarchical structure of behavior trees enables the reuse of subtrees, allowing for more efficient creation of new behaviors or the extension of existing ones. This modular design allows for a more streamlined

development process, as it enables developers to easily add or modify behaviors without having to make extensive changes to the overall structure of the system.

It can be highlighted that while behavior trees provide an advantage over finite state machines, it has been demonstrated through the research of Colledanchise [2017] and Jones [2020] that both BTs and FSMs are mathematically equivalent in terms of their capabilities.

The validity of these advantages has been substantiated through various contemporary studies, including (Cooper & Lemaignan, 2022) (Tadewos et al., 2019), and more insights on the topic will be examined in greater detail within 2.2.3 Section.

2.2.2.1 Hierarchical Finite State Machines

Hierarchical Finite State Machines (HFSMs), were created to address some limitations of traditional FSMs. In an HFSM, a state can contain one or more sub-states and a state that contains two or more states is known as a super-state. In an HFSM, transitions between super-states are referred to as generalized transitions. These transitions can reduce the number of transitions needed by connecting two super-states instead of connecting a larger number of sub-states individually. Each super-state has a designated starting sub-state that is activated when transitioning to that super-state.

One potential disadvantage of using Hierarchical Finite State Machines (HFSMs) over Behavior Trees is complexity. As the number of states and transitions in an HFSM grows, the system can become increasingly complex and difficult to understand. This can make it harder to maintain and update an autonomous system, especially as the system becomes more intricate.

Additionally, the rigid structure of HFSMs can limit their ability to handle more complex or dynamic behavior, like in a robot swarm. Behavior Trees are designed to be more flexible in handling complex and dynamic behavior. They use a hierarchical tree structure, which allows for more intuitive visualization of the flow and logic of the system. The tree structure also allows for a more modular approach the design of the system, making it easier to add, remove, or modify individual behaviors without affecting the rest of the system. Furthermore, BTs are well suited for parallel execution, where multiple tasks can be run at the same time, whereas HFSMs are not able to perform parallel execution.

It is worth noting that despite these potential disadvantages, HFSMs can still be a suitable choice for certain AI systems depending on the requirements of the system. However, for more complex or dynamic systems, like the one presented in this research, BTs are often considered to be a more flexible and scalable option.

2.2.3 State of the art in Behavior Trees

Even though BTs have been around for more than a decade now, not too many years ago the complaint was that in the field of Behavior Trees "The available literature lacks the consistency and mathematical rigor required for robotic and control applications" (Marzinotto et al., 2014). Since then, attempts were made to standardize the basis of behavior trees, as seen in (Champandard & Dunstan, 2019). The purpose of this review is to provide a comprehensive overview of the current state of the art in the application of behavior trees in autonomous systems.

In this research conducted in 2021 by Legarda.H studied a methodology for the collective transport of heavy objects through the utilization of an industrial swarm in a simulated setting is presented. The method was developed and tested in a simulated environment, using a combination of freely available open-source libraries. The robots employed controllers that incorporate a behavior tree architecture, which were optimized using genetic programming (GP) techniques to generate actuator commands. Coordination was achieved through a decentralized negotiation strategy that employed direct communication. The findings indicated that the genetic programming algorithm is capable of generating controllers that are able to safely lift and convey multiple loads concurrently towards a designated area, referred in the project as nest region. However, further refinement would be needed to implement these advancements in a real-world environment in order to assess its sensitivity to the reality gap.

The first research (Jones, 2020), linked as well with another earlier publication (S. Jones et al., 2018), puts the focus on the issues of designing optimal and readable systems. In order to design swarm robot systems with desired collective behavior, the challenge lies in designing controllers for individual robots that will produce the desired communal behavior through interaction. The current solutions studied for the mentioned research often relied on offline automatic discovery using artificial evolution of robot controllers, which are then transferred to the swarm, but that approach had limitations such as the need for additional supporting infrastructure for evolution and communication and also the evolved controllers being opaque and difficult to understand which could compromise safety and explainability. This research addresses

both these issues by using behavior trees. Behavior trees were used as the individual robot controller architecture with great results, focusing on them being modular, hierarchical, and human-readable. Automatic tools were developed to simplify large evolved trees, making it possible to understand, explain, and improve the evolved controllers.

The research highlights three main topics to research in the future; firstly, adaptability represents a crucial aspect to be considered, for the system described to possess the ability to better adapt its behavior in response to changes in the environment. Another important area of investigation is the reality gap and the impact of the chosen architecture. Additionally, expanding the system to operate in three-dimensional environments is also worth exploring, since the research was made in a two-dimensional setting. The videos of the implementation of this research can be consulted.

2.3 Communication Protocols in IIOT

The present era of society finds itself on the brink of a new epoch, as the processing power and connectivity of industrial embedded devices continues to increase, paving the way for a multitude of innovative applications. These advancements, encapsulated inside the concept of Industry 4.0 and more specifically Industrial Internet of Things promise to alter the way we manufacture and consume. The success of these applications is largely dependent on the communication between devices. One of the obstacles to achieving effective communication is the existence of a plethora of protocols developed in various domains, as stated in (Rodríguez et al., 2016). This section will focus on the protocols that are most commonly used in industrial environments, particularly in the area of additive manufacturing, such as OPC-UA (in conjunction with Umati), MQTT, and REST API (along with HTTPS).

The objective of this section is to provide a comprehensive examination of the underlying principles behind the most widely employed communication protocols, which will be subsequently subjected to comparative analysis in the experimental section of this report.

2.3.1 OPC-UA

OPC UA (Unified Architecture) is a communication protocol used in additive manufacturing (3D printing) that enables the seamless exchange of data between different devices and systems, allowing for efficient and accurate control of the manufacturing process. It allows for the integration of various equipment and software, such as 3D printers, scanners, and CAD software, to communicate in real-time, allowing for coordination of the entire manufacturing process. Overall, OPC UA is essential for the success of additive manufacturing, allowing for efficient control, coordination and automation of the manufacturing process, with added security [chapter to be completed].

2.3.1.1 Umati

UMATI (Universal Machine Type Interface) is a standardized protocol for communication between industrial machines and controllers. It was developed as a collaboration between the OPC Foundation and the VDMA (German Engineering Federation) with the goal of creating a universal interface for machine communication that is open, standardized, and easy to use. It is based on OPC UA, a widely-used open industrial communication standard, which enables secure and reliable communication between machines and controllers in industrial environments. UMATI enables interoperability between different machines and controllers, regardless of the manufacturer or communication protocol they use. This makes it easier to integrate new machines into production processes and to connect different types of equipment together. In the context of the IIoT, UMATI is an important building block for enabling communication and data exchange between devices and systems in an industrial environment.

2.3.2 MQTT

MQTT (Message Queuing Telemetry Transport) is a publish/subscribe-based messaging protocol designed for use in low-bandwidth, high-latency, or unreliable network environments. It is commonly used for Internet of Things applications where devices and sensors need to send and receive data over a network. MQTT provides high performance and low latency for IoT and M2M (Machine-to-Machine) communication, due to its lightweight and efficient design.

In MQTT, clients and servers communicate by publishing and subscribing to topics, with the broker acting as the intermediary. The client publishes a message to a topic and the broker distributes the message to all subscribers to that topic, allowing

for efficient one-to-many communication. The use of binary encoding and efficient compression techniques helps to minimize the overhead of MQTT and improve its performance, while the support for different levels of Quality of Service (QoS) for messages allows clients to control the level of reliability they require for their messages.

2.3.3 Http

The Hypertext Transfer Protocol (HTTP) is a commonly employed communication protocol for the transfer of data across the Internet. Within the field of the Industrial Internet of Things, HTTP can be leveraged to facilitate communication and data exchange among devices and systems within a network. One implementation of HTTP in IIoT applications is through the utilization of Representational State Transfer Application Programming Interfaces (REST APIs). REST APIs are web services that conform to the REST architectural style and leverage HTTP to transmit and receive data. The REST API is an architectural pattern for building web services that adheres to the principles of Representational State Transfer. These principles aim to create scalable and flexible web services that can be easily consumed by a broad range of clients, including web browsers, mobile devices, and other applications.

The HTTP protocol serves as the underlying communication mechanism for REST APIs, with HTTP methods such as GET, POST, PUT, and DELETE utilized for sending requests and receiving responses between the client and server. REST APIs typically utilize HTTP status codes to indicate the success or failure of a request and HTTP headers to carry supplementary information, such as authentication tokens, content types, and other metadata.

In the following section, we will delve further into the topic of REST API, as it serves as the framework in which we have conducted our tests for the implementation of a potential triggering mechanism for the machines in the production line of the project.

2.3.3.1 REST API

As explained in the section above, it is imperative to discuss the underlying theory of the REST API, as the framework for the trigger mechanism was tested using Markforged 3D printers. These printers are equipped with the Eiger API V3 (Eiger API V3, n.d.), which operates on the REST API architecture. As will be further elaborated upon later on, the testing involved the retrieval of reports from the Markforged printers. The Markforged company provides a REST API for their

industrial 3D printing systems. This API allows developers to integrate the functionality of the 3D printing systems into other applications and automate various tasks such as printing and monitoring the print progress.

It is important to understand the distinction between RESTful and REST API. A RESTful API is an API that adheres to the principles of REST. It is a web service that follows the REST architectural style and provides a set of operations that can be performed on resources identified by URIs. A REST API, on the other hand, is simply an API that uses REST principles. The REST API uses HTTP requests to send and receive data and is designed to be easy to use and flexible for integration with a wide range of systems. The REST architecture, was introduced by computer engineer Dr. Roy Fielding in his doctoral thesis in 2000, (Fielding, 2000). REST is a popular method for connecting components and applications within a microservices architecture due to its flexibility and autonomy for developers. REST API is a way of interacting with web and cloud services using HTTP requests, as described by Schiekofer et al. (2018). It uses either JSON or XML to send and receive data and is based on the client-server model, where the client triggers actions on the server and the server reacts.

Talking about the main characteristics of REST we can say that it`s characterized by its uniform interface, with its four constraints of resource identification, resource manipulation, self-descriptive requests and responses, and hypermedia as the engine of application state. REST is also based on a layered system, where each layer provides services to the layer above and uses services from the layer below. The client-server interaction in REST is stateless, meaning no session state is allowed and all necessary information must be included in each message. In the context of REST APIs, statelessness means that each request made to the API contains all of the information necessary to complete the request, and the server does not store any client context or session information between requests. This means that the server does not have any memory of previous requests, and each request is treated as an isolated transaction.

This design principle is in contrast to stateful systems, where the server maintains a session state, and each request is dependent on the state maintained by previous requests. The stateless design of REST APIs makes them more scalable, cacheable, and easier to maintain, as there is no need to store session data on the server. Additionally, it also allows for greater flexibility, as the client can make requests to any server at any time, without having to worry about maintaining a session or maintaining state information on the server.

There has been a significant amount of research and technological progress in the implementation of REST APIs in various contexts. One example is the use of REST

APIs in Android mobile apps, as studied by Abdellatif et al. (2020). Another example is the use of REST APIs in cloud testing, as explored by Wolde & Boltana (2021). REST APIs are widely utilized in a variety of applications, including web applications, mobile apps, IoT devices, and the Industrial Internet of Things (IIoT). They are employed for tasks such as data retrieval, creation and updates, user authentication, and access control to resources. In the field of 3D printing, REST APIs are utilized to transfer 3D models, automate printing processes, and allow for remote control of 3D printers.In the context of the IIoT, REST APIs play a crucial role in enabling communication and data exchange between devices and systems over a network. This is due to the simplicity, flexibility, and scalability of REST APIs. By using REST APIs, devices and systems can effectively communicate and exchange data, allowing for greater efficiency and effectiveness in the IIoT.

The performance of a REST API implementation can vary depending on several factors, including the complexity of the operations being performed, the efficiency of the server-side code, the speed of the network and the devices being used, and the number of concurrent users accessing the API. In general, REST APIs are designed to be lightweight and fast, as they use the HTTP protocol and rely on the client-server model.

3 Methodology and results

The present study will be conducted within the framework of Machine Connectivity 3D Pulse Control (3YOURMIND, 2023). This IIOT solution is based on the utilization of the OPC UA (Ladegourdie & Kua, 2022) communication standard and the Universal Machine Technology Interface (UMATI). UMATI is a relatively novel standard that is recently being implemented.

First, a research on the possible models of computation that will be used to model the logic of the production line has been conducted. Once the model is chosen, it will be necessary to extend the available debugging tools as well as implementing a trigger utility to start the workflow create a trigger utility that will start the workflow controller, this trigger utility will also be used for pulling production reports from a Markforged machine based using Representational state transfer (REST) a standard for internet data exchange this data transfer with a real machine is a key component to simulate and test the project beforehand.

After developing the trigger utility, the next step will be to continue with the development of the workflow controller (as said before based on Behavior Trees) implementing the machines belonging to the production line of the project. The software development approach chosen in this research is Test Driven Development through automated mock testing with GoogleTest (also known as gtest) and Doctest frameworks in C++. This approach has proven successful over time and is now considered to be a good practice (Petrovic et al.,2021)

To ensure the achievement of the determined objectives in this research the implementations will be tested and simulated. In addition, the complete POLYLINE project will be integrated and tested in the BMW production structures. Simulation was chosen as the primary testing method. However, in Section 3.6 the industrial testing process will be discussed to some extent, the data collected from the on-site results of this testing will be analyzed in order to derive Key Performance Indicators that provide a general understanding of the performance trends in this implementation.

3.1 Design and Functionality of the Machine Connectivity System

Before embarking on a discussion of the various implementations and testing that have been conducted, it is imperative to first gain a thorough understanding of the fundamental design of the system. This is because the backbone of the system's design serves as the foundation for all subsequent developments and is critical to the system's overall effectiveness and efficiency. The schematic representation of the system's functionality, as depicted in Figure 3 , provides a comprehensive illustration of the operations that take place within the framework of Machine Connectivity. It is through this representation that the underlying design principles, communication channels, and component interactions can be fully comprehended.

The trigger utility, developed as part of this research, is situated within the Manufacturing Execution System framework. Its purpose is to transmit the required files and initiate the execution of the workflow. The trigger utility plays a pivotal role in ensuring that the workflow is executed in a timely and efficient manner, serving as the starting point for the various processes and operations that take place within the system, meaning it first send the files and starts the workflow. The signal aggregator receives input signals which serve as stimuli for the activation of Behavior Trees.

Upon activation, the Behavior Trees then execute their nodes by issuing specific command instructions to the designated drivers of the machines, thereby facilitating the execution of the intended actions. The Behavior Tree dispatches commands to the various machines (including Falcon, Grenzebach, EOS,…). Upon receipt of the commands, the machines execute the instructions and send the results back to the Behavior Tree, which reacts accordingly. The signal aggregator then forwards the results to the Manufacturing Execution System.

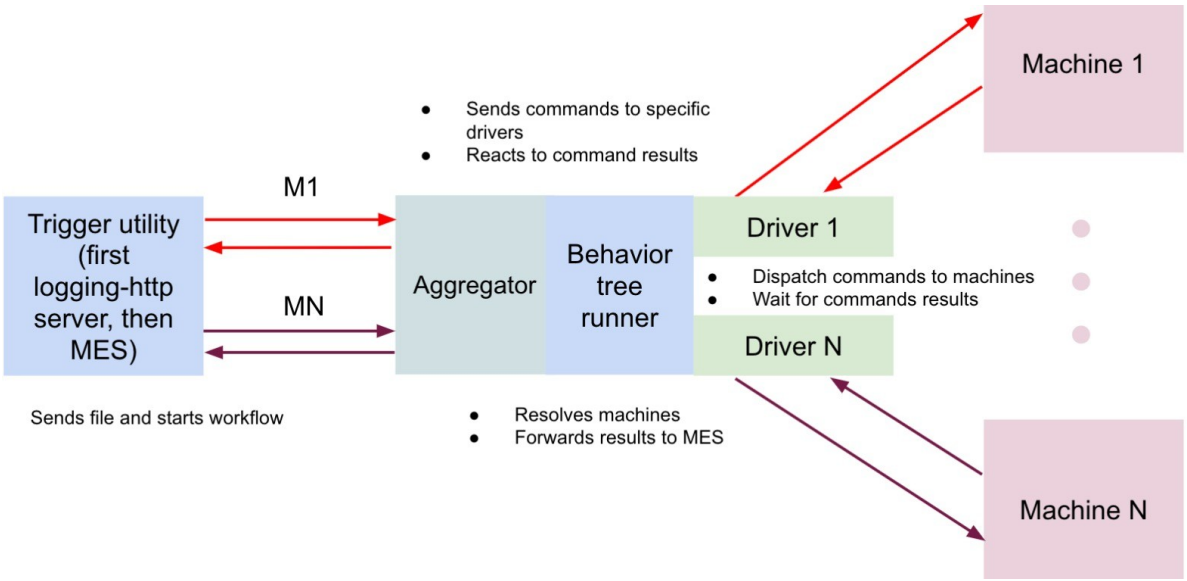


Figure 3. Schema of the connections and functionalities in the project.

3.2 Trigger utility - Markforged reports

Initially, between the goals of this research project there was to conceive a trigger utility to initiate the operations of the machines that would be part of the workflow (like starting the job printers, or starting the postprocessing movements,...). The trigger utility was envisioned as the responsible to start the jobs in the workflow, and it was tested using a Markforged printer. The objective of this implementation was to initiate a printer job and generate and save production reports from a Markforged machine. This implementation would serve as a framework for the eventual in site deployment for the Polyline project.

The trigger utility would act as the starting point of the production process, initiating the workflow and setting the wheels in motion for the rest of the tasks to be executed, based on the deployed framework in this section, the MES would send a switching on signal to the machines included in the project. This deployment was also designed with the intention of extracting production reports from the Markforged machine, which would provide a comprehensive overview of the production process. The reports generated would include key metrics such as output, efficiency, and quality, thereby enabling a more thorough analysis of the production process.

The Eiger API, developed by Markforged, is the software development tool that allows to integrate Markforged 3D printing solutions into custom applications, workflows, and systems. The API is defined by an OpenAPI Specification and features interactive documentation generated from the specification file, making it easy to understand and use. OpenAPI Specification (OAS) is a standard for designing and describing RESTful APIs. It is used to define the structure of an API, including its endpoints, operations, parameters, and responses. The OpenAPI Specification is a language-agnostic standard, meaning that it can be used to describe APIs written in any programming language. The Eiger API is structured around resources and collections of resources that are accessible through unique URIs. Interaction with these resources is performed using standard HTTP methods and it is important to include both "https" and "www" in the request syntax to ensure secure communication. The API server returns responses in JSON format.

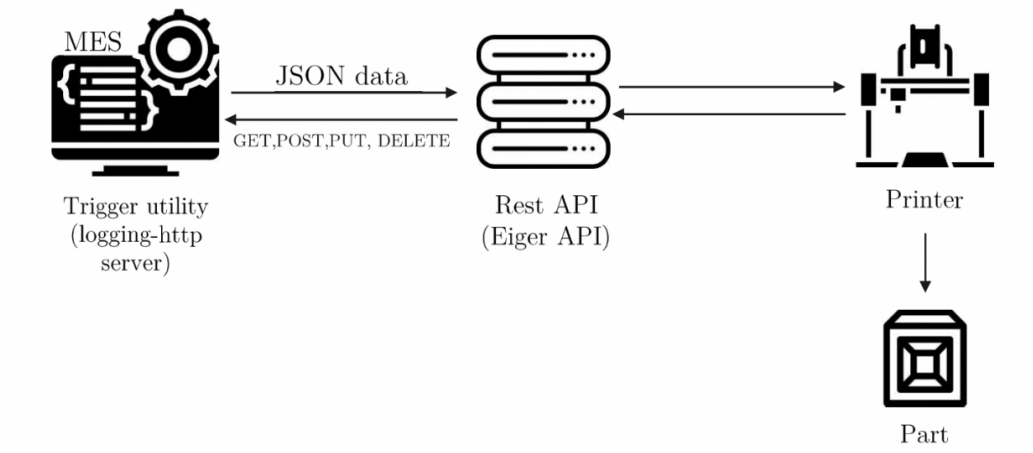


Figure 4. Communication architecture of the implementation with the Markforged printer

The JSON file from the Eiger API from Markforged uses Protocol Buffers (Protobufs) to exchange information with the Manufacturing Execution System. Protocol Buffers are a data serialization format that is language- and platform-agnostic, making them well-suited for use in REST APIs like the Eiger API. Protobufs are defined in a .proto file, which specifies the structure of the data. Code generation tools can then be used to generate code in the desired programming language, allowing

developers to encode and decode the data. This makes it easy to transmit complex data structures over a network in a compact binary format, improving the performance and efficiency of the system.

In the context of the Eiger API, the use of Protocol Buffers enables the efficient exchange of information between the API and the Manufacturing Execution System, making it easier to automate and streamline the 3D printing process. Access to the Eiger API is secured through authentication, which is used to identify the API consumer and verify their authorization to use the API. The authentication process is accomplished through HTTP Basic Authentication and the use of API keys. API keys for Eiger consist of two components: an API Access Key and an API Secret Key.

The illustration below provides a visual representation of a portion of the contents contained within the .proto file. The creation of this image utilized was done using an open-source tool[1]. It is noteworthy to emphasize that this graph only encompasses a fraction of the .proto file, as constructing a graph with all the implemented numerous nodes and arrows would be impracticable. The aim of this graph was to exhibit a selection of the interconnections within the file, however, it is important to consider that the original file is significantly more extensive and intricate. This provides a glimpse into the complexity and interconnections within the .proto file, further emphasizing the benefits of using Protobufs in the Eiger API to efficiently exchange information with the MES.

[1] https://github.com/seamia/protodot

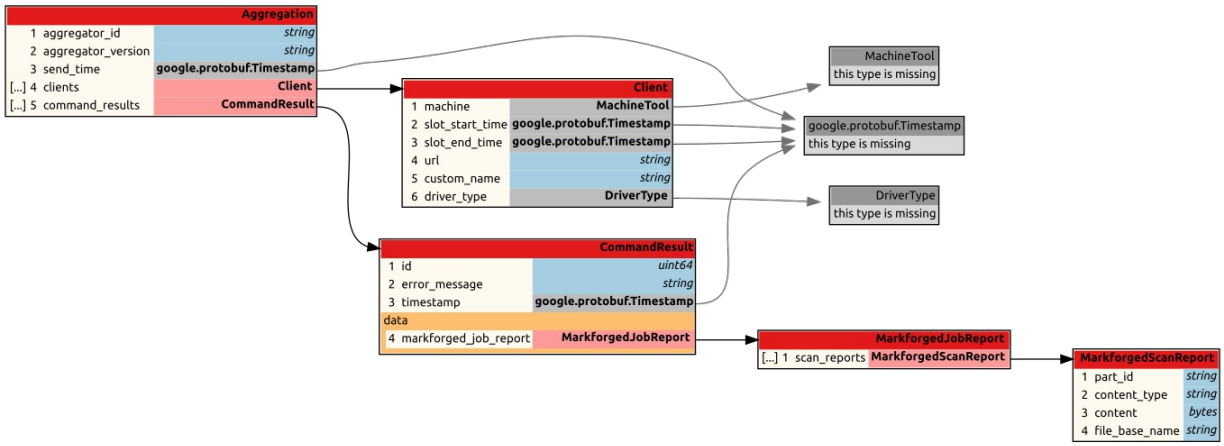


Figure 5. Schema of the Markforged communication

This implementation was put to the test and demonstrated satisfactory outcomes. It facilitated the activation of a designated machine, and also enabled the generation of a PDF document containing information on both the print process and communication data. The successful results of the testing indicate that this implementation is effective in achieving its intended triggering purpose. The ability to initiate a machine and generate a comprehensive record of the print and communication data in a portable and easily accessible format could enhance the overall efficiency and effectiveness of the workflow.

The development of the utility underwent testing using the company's tools. However, the initiation of the process remotely was not pursued further in the project due to the restrictions imposed by industrial regulations and policies. These regulations and policies served as barriers that prevented the implementation of remote starting for the industrial machines. The outcome of the testing phase using the company's tools provided valuable insights for future considerations and advancements, However, the implementation of remote starting is still subject to the resolution of regulatory and policy restrictions.

3.3 Comparison between protocols

OPC UA is an industrial communication standard for machine-to-machine or industrial Internet of Things communication. It has several advantages over MQTT, one of which is its ability to connect machines to each other. OPC UA provides a secure and reliable means of exchanging data between machines, allowing for real-time data exchange with a high degree of semantic interoperability. This means that data

can be easily and accurately interpreted by different systems, even if they have different underlying technologies or protocols. Also provides a hierarchical structure for data representation, which makes it easy to manage and organize large amounts of data generated by multiple machines. It also supports multiple data types, including complex structures and arrays, which makes it suitable for a wide range of industrial applications.
In contrast, MQTT is primarily designed for small-scale, low-bandwidth IoT devices and provides only a basic publish/subscribe messaging model. It does not provide the level of interoperability or data management capabilities that OPC UA offers, making it less suitable for industrial M2M communication.

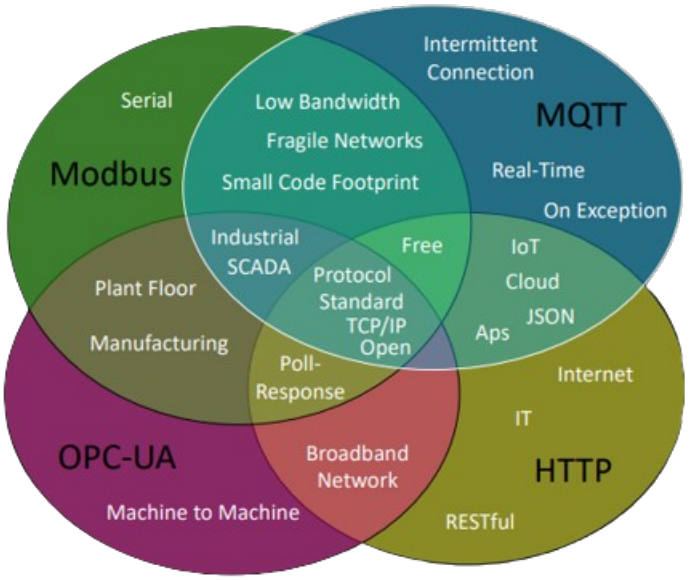


Figure 6. Schema outlining the typical application areas of prevalent communication protocols.

In summary, the capacity of OPC UA to establish interconnectivity among machines, in tandem with its dependable and secure data transmission, sophisticated data depiction, and semantic interoperation, positions it as a superior alternative for industrial machine-to-machine communication, as opposed to MQTT. [missing data]

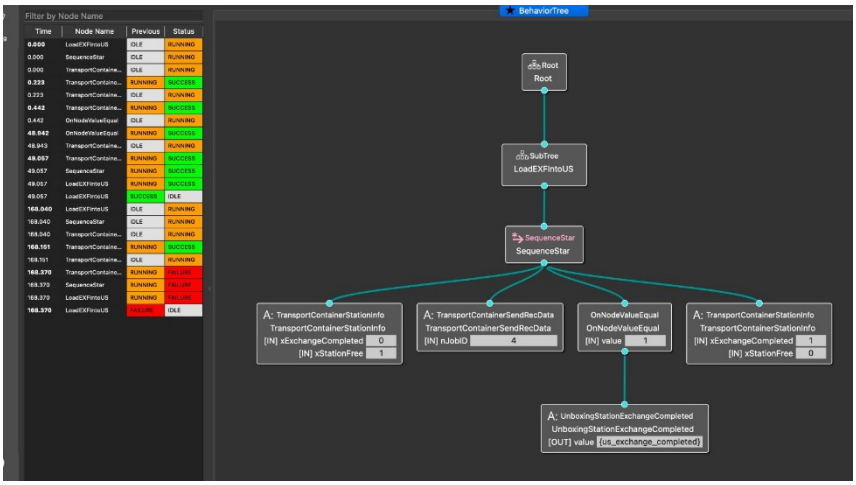
3.4 Behavior Tree implementation

In this study, the C++ implementation of BTs employs the BehaviorTree.CPP library developed by Faconti and Colledanchise (Faconti, 2022). It is worth noting that

the compatibility of Groot 1.0 is exclusively with BehaviorTree.CPP version 3.8.x, thus the demonstration presented in this paper, is constructed upon the framework provided by BehaviorTree.CPP 3.8.x.

The implementation of the behaviortree.cpp library for this project is motivated by several factors. Firstly, as discussed before, the library features a graphical user interface, Groot, (Faconti, 2019) for the visual description of workflows, which can then be loaded directly into the application, meaning if the nodes are already defined the behavior tree can run directly, in contrast to common state-machine libraries that require the manual translation of UML diagrams into C++ code.

Additionally, it is possible to highlight that BTs are generated using the XML language, which promotes readability of the trees. Moreover, the library is capable of recording, replaying, and displaying the live state of workflows through the GUI, as it can be seen in Figure X.



The dynamic loading of the Behavior Tree structure at runtime allows for on-the-fly adjustments to the system's behavior, eliminating the need for program recompilation. This feature is especially advantageous when testing the system's workflow with physical machines in real-world environments, as the one described in this research. As described in the last section this flexibility also enables developers to create more robust and adaptable systems, as they can easily adjust and fine-tune the system's behavior to meet specific requirements or handle unexpected situations.

Moreover, as illustrated in Figure 7 and expounded upon by Ghzouli et al. (2022), the BehaviorTree.cpp library has recently gained substantial traction and adoption in the field. This trend can be attributed to the library's capacity to efficiently model and

regulate the behavior of autonomous agents by means of a hierarchical tree structure, as well as its provision of numerous exemplary cases that facilitate the establishment of research settings.

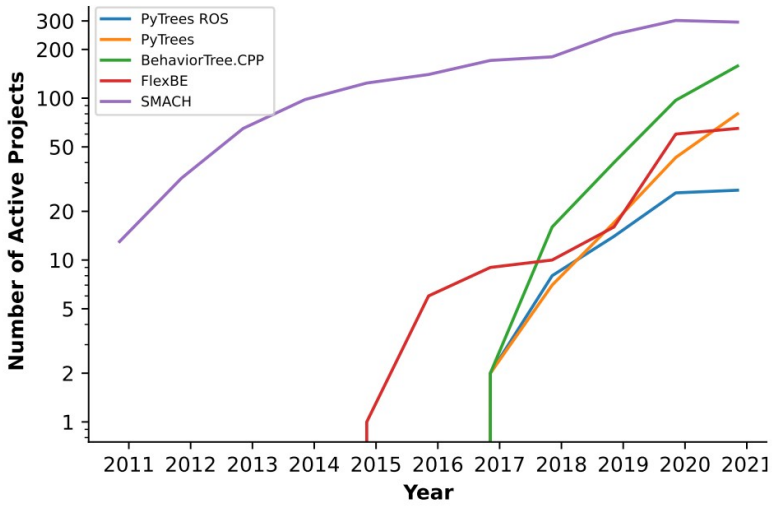


Figure 7. A study of the trend of usage of state machine and behavior tree languages in open-source projects on GitHub over time. Extracted from (Ghzouli et al., 2022)

After a detailed exposition of the tools and considerations utilized in the project, the subsequent step entails describing the definitive implementation of the workflow. By delineating the key steps and procedures involved in the workflow, the project can be comprehensively analyzed and understood.

One of the key features of Behavior Trees is that they are executed sequentially, meaning that each node in the tree is visited in a particular order, typically from top to bottom. Despite being executed sequentially, the final behavior tree file [*tree.xml*] is divided in subtrees due to its sheer size, this division is primarily for the purpose of enhancing the file's readability and facilitating its creation and debugging. The main behavior tree is compartmentalized into four distinct sections, each with a specific purpose based on different parts of the workflow (printing, post-processing, robot selection,…). As seen in the picture below, Figure 3, this four-sectioned structure constitutes the primary behavior tree and represents the main logic behind the workflow.

The opening section of the study focuses on delineating the essential preconditions necessary for the activation of the printing process. The subsequent three sections outline the number of programmed days necessary to complete the entire procedure. It is important to note that the division into days is merely for the purpose of

organization and does not necessarily indicate that each of the three sections will take a full day to complete.

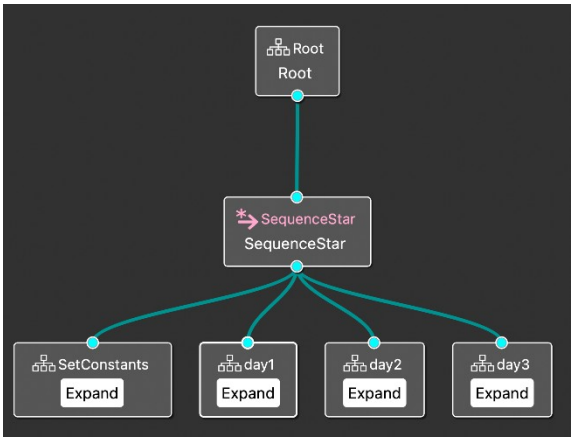


Figure 8. Main behavior tree of the project

The initiation of the workflow involves the configuration of necessary parameters for the complete workflow. The initial step in the tree employs the utilization of a node referred to as the SetBlackboard node, which is already available in the BehaviorTree.cpp library. This node serves the purpose of defining the values of the signals utilized for communication between the tree and the machines. The initial phase of the workflow is encapsulated within a subtree referred to as the "SetConstraints." This subtree serves as a foundational step to establish the parameters required for the overall workflow. The first step of this tree involves the use of the "SetBlackboard" node, which is part of the BehaviorTree.cpp library. The function of this node is to set the values of signals utilized for communication between the tree and the machines.

The SetBlackboard node is a type of Action node in Behavior Trees that is used to set a value on the Blackboard. As explained before, the Blackboard is a data structure that is used to share information between different parts of the Behavior Tree. It enables nodes in the tree to read and write data, which can be used to make decisions and control the system's behavior. Upon execution, the SetBlackboard node takes in a key-value pair and sets the corresponding value on the Blackboard. The key represents a specific variable, while the value can be any type of data, such as a number, string, or object. The value set on the Blackboard can be accessed by other nodes in the tree through the use of the GetBlackboard node.

The SetBlackboard node is particularly useful for storing and accessing data that needs to be shared across different parts of the Behavior Tree. For example, it can be used to store the location of a target object, which can be accessed by other nodes in

the tree to make decisions about the system's behavior. Additionally, it can be used to store information about the system's state, such as whether it is currently in a "searching" or "attacking" mode.

It is important to note that this initial phase of the workflow will be improved and refined in the future by a graphical user interface (GUI) that goes beyond the scope of this thesis. The aim is to provide a user-friendly and intuitive interface for setting the parameters of the workflow.

To talk about the different steps in the process of constructing the decision tree was initiated with the intention of considering each company process in isolation. This approach was chosen due to its simplicity in terms of signaling and ease of interpretation. To exemplify this, the photo below showcases the Nabertherm furnace

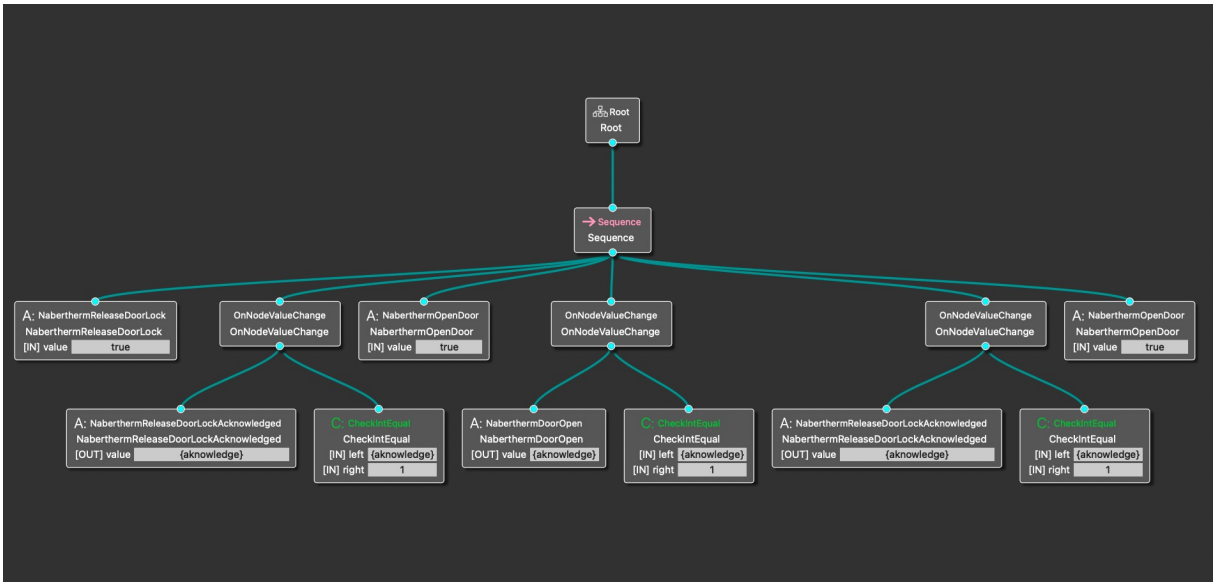


Figure 9. Nabertherm OpenDoor behavior tree

as an example.

However, as the construction of the decision tree progressed, it became evident that this approach was not the best course of action. The lack of collaboration between different processes and machines resulted in a limited scope of operations and hindered the overall effectiveness of the decision tree. It was determined that the approach needed to be altered to incorporate a broader perspective that incorporated the collaboration between different processes and machines.

Therefore, it was decided to construct the decision tree based on specific operations that involve the integration of various machines from multiple companies. This approach allowed for a comprehensive analysis of the operations and allowed for a

more in-depth examination of the processes involved. Examples of these specific operations include bin-picking, falcon post-processing, and others.

The revised approach resulted in a more comprehensive and effective decision tree that improved the efficiency and accuracy of the operations analysis. The integration of multiple processes and machines allowed for a more holistic approach that resulted in a better understanding of the relationships between the different operations and their impact on the overall outcome.

In conclusion, the initial approach of considering each process in isolation was found to be insufficient for the construction of an effective decision tree. The revised approach of constructing the tree based on specific operations that involve the integration of various machines from multiple companies resulted in a more comprehensive and effective analysis of the operations.

Now I would give a The following section provides a summary of the activities carried out during the schedule. On day1, the printer exchange frame (where the parts are printed on) is situated inside the transfer station (TS). The virtual controller sends all the necessary commands to move the exchange frame into the final state where it sits inside the printer transfer. The transport container is placed by the fleet management system and this TS is used to transport the exchange frame. At the start of Day1, the exchange frame, where the part is printed, is expected to be inside the transfer station, but it will eventually be moved to the next step which is the 3D printer (an EOS printer). Once the print is completed, the part cools down inside the printer.

On Day2, the printed part is inside the printer and will be moved to the transport container. The transport container will then be moved to the unboxing station.

On Day3, the box with the parts is in the unboxing station (Station 1). At the end of the day, all the separated parts are ready to be picked (Station 7). The process starts by moving the box with the parts to the Falcon (post-processing). Station 3 must have clean boxes as a precondition. The parts are then sent to Station 4 where the bin picking scans and recognizes them before sending them to different stations. All stations must have a small clean box. The robot must be instructed on how many parts it should expect to find and then separate them. The large empty clean box is moved from Station 4 back to the buffer in Station 3 for the next cycle. The small boxes are then moved to the outfeed where they are picked by people. Once the boxes are empty, they get picked and returned to the system.

3.4.1.1 Personalized nodes

This section will delve into the discussion of personalized controller and condition nodes that have been incorporated into the workflow. It is possible to highlight the nodes OnNodeValueEqual, OnNodeValueDiF, This code is a part of a behavior tree node called "OnNodeValueEqual". This node is used to compare the value of a child node with a given value. It checks the node's status and if it is successful, stores the value of the child node in a variable called "child_value". The node then evaluates whether this stored value is equal to the given value. If it is, the node will return a success status, otherwise it will return a running status [to be completed].

3.5 Mock environment results

To ensure that the implementation is capable of handling the workflow objectives, comprehensive testing and simulation will be conducted on the developed implementations. The complete POLYLINE project will also be integrated and tested within the BMW production structures to verify its functionality and performance within a real-world production environment, however, testing within BMW's facilities necessitates the collaboration of numerous companies, and it was forecasted that it may not be feasible to complete the testing prior to the submission of this research. Hence, simulation has been deemed as the primary testing tool to evaluate the this research viability and potential for implementation.

In this research, the software development methodology adopted is Test-Driven Development through automated mock testing using the GoogleTest (gtest) [2] frameworks in C++. This approach has become widely popular in the software engineering community, due to its demonstrated efficacy and effectiveness over time. TDD emphasizes writing automated tests before writing actual code, which serves as a guide for development and ensures that the code meets the desired specifications. As a result of its widespread success and proven track record, TDD has now become the de-facto standard in software development.

Google Test provides a framework for writing and running automated tests for C++. They can be used to automate mock testing, which is a technique used to simulate the behavior of dependent components in a system under test. This allows developers to test individual units of code in isolation, and to verify their behavior under various conditions, without having to rely on the actual implementation of the dependent components. In gtest, automated mock testing can be performed by using

[2] http://google.github.io/googletest/

mock objects to simulate the behavior of dependent components in the system under test. By using these mock objects, developers can write tests that verify the behavior of their code in isolation, without having to worry about the implementation of the dependent components.

An experimental environment was established with the purpose of assessing the performance of behavior trees in an industrial context. The evaluation involved conducting a suite of exercises that were designed to simulate real-world scenarios and elicit both successful and unsuccessful outcomes. These exercises were carefully crafted to challenge the behavior trees and test their ability to respond appropriately in different situations. The results of these exercises were used to determine the overall functionality of the behavior trees and to identify any areas for improvement. The ultimate goal of this evaluation was to ensure that the behavior trees were fully operational and capable of meeting the requirements of the industrial setting.

As part of the testing process, various scenarios were evaluated to ensure the functionality and reliability of the system being developed. This involved the evaluation of situations that required the leaf nodes to communicate failure status to the root node, such as the door of the transfer station not being able to open, the unboxing station not being ready, and the 3D printing not being completed. These scenarios were selected as they represent common failure conditions that may arise in the system and provide insight into the system's ability to respond to and recover from such situations. It is important to emphasize that the most important scenario to undergo testing has been the “happy path” scenario, a widely recognized concept in the realm of design and software in general (Atanasijevic, 2020), in which all connections operate efficiently within the allotted time frame and the tree, as a whole, returns a value indicating successful completion following the callback.

To accomplish this, mock objects of the nodes were created. These mock objects simulate the signal reception through the blackboard between the nodes and the real machines in the system, allowing for testing without the need for actual physical machines. The mock objects also simulate the signals received through the blackboard, which serves as a shared memory space for storing signals from the machines. This allows for testing of the system's ability to process and store signals correctly.

This approach to testing provides several benefits. It allows for faster and more efficient testing, as well as a controlled environment for testing, as the mock objects can be programmed to simulate specific scenarios and conditions. Additionally, it makes it easier to identify and isolate problems in the system, as the mock objects can be programmed to simulate specific failure conditions. Overall, creating mock objects of

the nodes is an effective and efficient way to test the functionality and reliability of a system.

In conclusion, GoogleTest (gtest) provide efficient and adaptable options for automating mock testing. The utilization of TDD through these frameworks serves as a robust basis for this research, with the thorough testing and simulation ensuring the validity and reliability of the developed implementations and the successful integration of the POLYLINE project within BMW's production structures.

3.6 Analytics - On site results

Although the project is still in its testing phase, this section aims to elucidate the preliminary discoveries and findings made during the initial testing stages of the project. Furthermore, it serves to verify the theories formulated prior to the commencement of the project's development and highlight areas that require further improvement.

A significant discovery was the recognition of the potential offered by behavior trees in this project, particularly with regards to their modular and easily interpretable nature. This was observed due to the fact that the real-world environment in the factory was still undergoing development and not all machines could be tested, which resulted in multiple revisions to the final [tree.xml] file as various sequences of machines were implemented and individuals external to the research project became involved. This case study serves to illustrate the advantages of the modular structure of behavior trees in interdisciplinary environments, as they enhance human readability and manipulation of the system in general.

The study identified a key advantage of using Behavior Trees in this project, specifically in their modular and highly interpretable nature. This attribute proved to be especially useful as the real-world environment in the factory was still undergoing development and not all machines could be tested. As a result, multiple revisions were made to the final Behavior Tree file (tree.xml) to incorporate various sequences of machines and accommodate external contributors to the project.

The flexibility of Behavior Trees enabled the research team to make these adjustments without significant disruptions to the overall system's structure. This modularity allowed for greater adaptability and ease of interpretation of the system, as the team was able to manipulate the tree's structure to reflect the necessary changes in a timely and efficient manner. Furthermore, the team noted that the hierarchical

nature of Behavior Trees provided a clear overview of the system's behavior and facilitated the isolation and resolution of issues, leading to more streamlined development and debugging of the system.

The case study presented herein demonstrates the advantages of using Behavior Trees in interdisciplinary environments. Their modular and interpretable structure enables human readability and manipulation of the system, leading to improved development and ease of maintenance. As such, Behavior Trees can be a valuable tool for designing complex systems that require a high level of adaptability and modularity, as they can be easily adjusted and fine-tuned to meet specific requirements and handle unexpected situations.

Although the workflow controller is still undergoing testing, it has been possible to perform preliminary data analysis by extracting information from the log files generated by the machines. At present, the ability to generate aggregated analytics that would enhance the speed and efficiency of data analysis has not been achieved. Nevertheless, a Python code was developed to scrutinize the log files acquired from the aggregator with the intention of eliciting insights into the preliminary data. The code was written to extract meaningful insights from the log data and present them in a readable format. The data from the log files from the aggregator contained information about various events that took place during the testing phase of the project, including the time stamps of event and different components, any errors or issues that occurred, and the overall behavior of the system. Daily log files are generated for each test conducted.The code was written in Python to extract insights from the processes leveraged.

The research main execution testing was done through a simulation in a mock environment. While the mock environment provided valuable information of the feasibility of the implementation, the recent on-site testing revealed additional insights into the project and verified certain theoretical expectations that were established, even though it was not a primary objective of the research.

The log analysis provided valuable insights into the performance and behavior of the aggregator during the testing phase. For instance, it provides a prediction of the ultimate efficiency of the additive manufacturing system. It is noteworthy that the data from these printings were obtained from the production of small pieces with a high level of detail 3. It also helped identify any bugs or issues that needed to be addressed, ensuring that the final product would be of high quality and perform as expected. The creation of this Python code was a crucial step in the development and testing of the project, as it allowed the team to make informed decisions and ensure that the final product would meet the needs and expectations of its users.

In order to thoroughly analyze the project, the findings from the on-site testing phase will be discussed in this thesis. It is important to remember that these findings are from the testing phase and may vary in the final outcome. However, they still offer valuable insights into the project's Key Performance Indicators and the potential benefits of automation in the additive manufacturing field. [missing the gaphs]

The purpose of this analysis is to identify key performance indicators (KPIs) that can be extracted from log files of machines to assess their performance and provide insights into their operations. The specific KPIs that can be obtained depend on the type of machine and the purpose of the analysis.

One of the most basic KPIs that can be extracted from machine log files is the downtime and uptime. By analyzing log files, it is possible to determine the duration of time a machine was idle (downtime) and the duration of time it was operational (uptime). This information is crucial for understanding the overall efficiency of the machine and identifying areas where improvements can be made to reduce downtime.Cycle time is another important KPI that can be extracted from machine log files. By determining the time taken for a machine to complete a single cycle of operation, it is possible to identify bottlenecks in the production process and optimize machine operations to improve cycle time. Log files can also provide insights into the production rate of the machine. By analyzing log files, it is possible to determine the number of units produced by the machine over a given period of time. This information is useful for identifying areas where production can be increased to meet demand.

Another important KPI that can be extracted from machine log files is the reject rate. Log files can be analyzed to determine the number of units rejected by the machine due to defects or other issues. This information is useful for identifying areas where improvements can be made to reduce the reject rate and improve product quality.

The amount of energy consumed by the machine during operation can also be extracted from log files, allowing for energy efficiency analysis. By analyzing log files, it is possible to identify areas where energy consumption can be reduced, leading to cost savings and environmental benefits. Finally, the frequency of errors or malfunctions occurring during machine operations can be extracted from log files. This information is crucial for identifying areas where maintenance is required to reduce the frequency of errors and improve machine reliability.

In conclusion, by analyzing machine log files, it is possible to extract a range of KPIs that provide insights into machine performance and operations. The specific KPIs

that can be obtained depend on the type of machine and the purpose of the analysis. However, some common KPIs that can be extracted from log files of machines include downtime and uptime, cycle time, production rate, reject rate, energy consumption, and error frequency.

4 Future work

The research presented in this Master's thesis represents a significant step forward in the application of IIoT and behavior trees in the context of additive manufacturing. However, there is still significant room for further exploration and development in this field. In future work, some of the following areas could be considered:

Deployment and Validation: The cloud workflow controller developed as part of this research needs to be deployed and tested in real manufacturing scenario in a real-world scenario for validation of its performance and effectiveness. This will provide further insights into the practical applications and limitations of the controller in an industrial setting.

Enhancements to the Behavior Tree Model: Further exploration of the behavior tree model could be undertaken to identify ways to improve its performance and efficiency in managing complex processes and decision-making. This may include investigating alternative approaches to constructing the behavior tree or exploring new technologies and techniques for enhancing the performance of the controller.

Integration with Other Technologies: The cloud workflow controller developed in this research could be integrated with other technologies and systems to enhance its functionality and potential applications. For example, the integration with machine learning algorithms or predictive maintenance systems could provide valuable insights and optimizations in the production process.

Scalability and Sustainability: As the number of connected devices and systems in the IIoT continues to grow, it is important to consider the scalability and sustainability of the cloud workflow controller. Research into approaches to address these challenges will be critical to ensure the long-term viability and success of this technology in industrial settings.

5 Conclusions

In conclusion, the findings of this master thesis demonstrate the potential for improving the efficiency and automation of the Additive Manufacturing process through automation, particularly in this case the implementation of a cloud workflow controller utilizing Industrial Internet of Things solutions. The research conducted in this thesis explored various models of computation, specifically focusing on the advantages of using Behavior Trees as a means of organizing and managing the complex processes involved in the production line. Additionally, a comprehensive study of communication protocols was carried out to evaluate the suitability of the OPC UA protocol for the intended workflow. The results of this study provided valuable insight into the most suitable communication protocols for future research in this field.

The workflow controller was tested in a mock environment and the results showed that the controller was able to support the workload generated by the production process, ensuring a smooth transition between the various stages of the process. Furthermore, the results also indicated that the proposed cloud workflow controller has the potential to increase efficiency and reduce manual intervention in the Additive Manufacturing process, thus contributing to the advancement of Industry 4.0. Overall, this research provides an important step towards optimizing the implementation of Additive Manufacturing technology and highlights the need for further research and development in this field.

6 Bibliography

1. (3YOURMIND, 2023)
3YOURMIND. (2023). 3YOURMIND. 3yourmind.com. Retrieved January 17, 2023, from https://www.3yourmind.com/production-and-quality-management
2. (Abdellatif et al., 2020)
Abdellatif, M., Tighilt, R., Belkhir, A., Moha, N., Guéhéneuc, Y.-G., & Beaudry, É. (2020). A multi-dimensional study on the state of the practice of REST APIs usage in Android apps. Automated Software Engineering, 27(3–4), 187–228. https://doi.org/10.1007/s10515-020-00272-9
3. (Agis et al., 2020)
Agis, R. A., Gottifredi, S., & García, A. J. (2020). An event-driven behavior trees extension to facilitate non-player multi-agent coordination in video games.
Expert Systems with Applications, 155(113457),113457. https://doi.org/10.1016/j.eswa.2020.113457
4. (Almada-Lobo, 2016)
Almada-Lobo, F. (2016). The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES). Journal of Innovation Management, 3(4), 16–21. https://doi.org/10.24840/2183-0606_003.004_0003
5. (Bhatt et al., 2020)
Bhatt, P. M., Malhan, R. K., Shembekar, A. V., Yoon, Y. J., & Gupta, S. K. (2020). Expanding capabilities of additive manufacturing through use of robotics technologies: A survey. Additive Manufacturing, 31(100933), 100933. https://doi.org/10.1016/j.addma.2019.100933
6. (Butt, 2020)
Butt, J. (2020). Exploring the interrelationship between additive manufacturing and Industry 4.0. Designs, 4(2), 13. https://doi.org/10.3390/designs4020013
7. [2] (Champandard & Dunstan, 2019)
Champandard, A. J., & Dunstan, P. (2019). The behavior tree starter kit. In Game AI Pro 360 (pp. 27–46). CRC Press.
8. [3] (Colledanchise, 2017)
Colledanchise, M. (2017). Behavior trees in robotics. Diva-portal.org. Retrieved January 9, 2023, from https://www.diva-portal.org/smash/get/diva2:1078940/FULLTEXT01.pdf
9. [4] (Colledanchise & Ögren, 2017)
Colledanchise, M., & Ögren, P. (2017). Behavior trees in Robotics and AI: An introduction. ArXiv [Cs.RO]. https://doi.org/10.48550/ARXIV.1709.00084
10. [5] (Cooper & Lemaignan, 2022)
Cooper, S., & Lemaignan, S. (2022). Towards using behaviour trees for long-term social robot behaviour. 2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 737–741.
11. (Delli & Chang, 2018)
Delli, U., & Chang, S. (2018). Automated process monitoring in 3D printing using supervised machine learning. Procedia Manufacturing, 26, 865–870. https://doi.org/10.1016/j.promfg.2018.07.111
12. [6] (Dijkstra, 1968)
Dijkstra, E. W. (1968). Letters to the editor: go to statement considered harmful. Communications of the ACM, 11(3), 147–148. https://doi.org/10.1145/362929.362947
13. [7] (Dorigo et al., 2013)
Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A. L., Decugniere, A., Di Caro, G., Ducatelle, F., Ferrante, E., Forster, A., Gonzales, J. M., … Vaussard, F. (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. IEEE Robotics & Automation Magazine, 20(4), 60–71. https://doi.org/10.1109/mra.2013.2252996

14. [8] (Dromey, 2001)
R.G.Dromey, (2001).Genetic Software Engineering – Simplifying Design Using Requirements Integration, IEEE Working Conference on Complex and Dynamic Systems Architecture, Brisbane.
15. (Eiger API V3, n.d.)
Eiger API V3. (n.d.). Eiger.Io. Retrieved February 4, 2023, from https://www.eiger.io/developer
16. (Faconti, 2022)
Davide Faconti. 2022. BehaviorTree.CPP library Documentation. https://www.behaviortree.dev
17. (Faconti, 2019)
Faconti,D.Groot. https://github.com/BehaviorTree/Groot. 2019.
18. (Fielding, 2000)
Roy Fielding. 2000. "Architectural Styles and the Design of Network-based Software Architectures," 128. University of California, Irvine, PhD Dissertation. https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation_2up.pdf
19. (Francesca et al., 2014)
Francesca G, Brambilla M, Brutschy A, Trianni V, Birattari M. 2014. AutoMoDe: a novel approach to the automatic design of control software for robot swarms. Swarm Intelligence 8(2):89–112 DOI 10.1007/s11721-014-0092-4.
20. (Freens et al., 2015)
Freens, J. P. N., Adan, I. J. B. F., Pogromsky, A. Y., & Ploegmakers, H. (2015). Automating the production planning of a 3D printing factory. 2015 Winter Simulation Conference (WSC), 2136–2147.
21. (Gao et al., 2019)
Gao, W., Zhang, Y., Ramanujan, D., Ramani, K., Chen, Y., Williams, C. B., Wang, C. C. L., Shin, Y. C., Zhang, S., & Zavattieri, P. D. (2019). The status, challenges, and future of additive manufacturing in engineering. Computer Aided Design, 69, 65–89. https://www.academia.edu/38520195/The_status_challenges_and_future_of_additive_manufacturing_in_engineering
22. (Ghzouli et al., 2020)
Ghzouli, R., Berger, T., Johnsen, E. B., Dragule, S., & Wąsowski, A. (2020). Behavior Trees in action: A study of robotics applications. ArXiv [Cs.RO]. https://doi.org/10.48550/ARXIV.2010.06256
23. (Ghzouli et al., 2022)
Ghzouli, R., Dragule, S., Berger, T., Johnsen, E. B., & Wasowski, A. (2022). Behavior Trees and state machines in robotics applications. In arXiv [cs.RO]. http://arxiv.org/abs/2208.04211
24. (Heckel et al., 2010)
Heckel, F. W. P., Youngblood, G. M., & Ketkar, N. S. (2010). Representational complexity of reactive agents. Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games.
25. (Horstkotte et al., 2021)
Horstkotte, R., Bruning, B., Prümmer, M., Arntz, K., & Bergs, T. (2021). Determination of the level of automation for additive manufacturing process chains. Hannover : publish-Ing. https://doi.org/10.15488/11257
26. (Ibrahim et al., 2018)
Ibrahim, H., Jahadakbar, A., Dehghan, A., Moghaddam, N., Amerinatanzi, A., & Elahinia, M. (2018). In vitro corrosion assessment of additively manufactured porous NiTi structures for bone fixation applications. Metals, 8(3), 164. https://doi.org/10.3390/met8030164
27. (Isla, 2005)
Isla, D. (2005, March 11). GDC 2005 proceeding: Handling complexity in the Halo 2 AI. Game Developer. https://www.gamedeveloper.com/programming/gdc-2005-proceeding-handling-complexity-in-the-i-halo-2-i-ai
28. (Isla, 2008)
Isla, D (2008). "Building a Better Battle: Halo 3 AI Objectives". In: Game Developers Conference. 2008.
29. [16] (S. Jones et al., 2018)

Jones, S., Studley, M., Hauert, S., & Winfield, A. (2018). Evolving behaviour trees for swarm robotics. In Distributed Autonomous Robotic Systems (pp. 487–501). Springer International Publishing.

30. [17] (Jones, 2020)
Jones, S. W. (2020). Onboard evolution of human-understandable behaviour trees for robot swarms. University of Bristol. Retrieved January 9, 2023, from https://research-information.bris.ac.uk/en/studentTheses/onboard-evolution-of-human-understandable-behaviour-trees-for-rob
31. (Industrie 4.0 - BMBF, 2011)
Industrie 4.0 - BMBF. (2011). Bundesministerium für Bildung und Forschung - BMBF. Retrieved January 20, 2023, from https://www.bmbf.de/bmbf/de/forschung/digitale-wirtschaft-und-gesellschaft/industrie-4-0/industrie-4-0
32. (Kellett, 2012)
Kellett, P. (2012). Additive Manufacturing and Automation Will 3D Printing Displace Automation Technologies? Association for Advancing Automation. https://www.automate.org/userAssets/riaUploads/file/Additive_Manufacturing_and_Automation.pdf
33. (Košťál et al.,2019)
Košťál, Peter & Mudriková, Andrea & Michal, Dávid. (2019). Group technology in the flexible manufacturing system. MATEC Web of Conferences. 299. 02001. 10.1051/matecconf/201929902001.
34. (Kim et al., 2015)
Kim, D. B., Witherell, P., Lipman, R., & Feng, S. C. (2015). Streamlining the additive manufacturing digital spectrum: A systems approach. Additive Manufacturing, 5, 20–30. https://doi.org/10.1016/j.addma.2014.10.004
35. [18] (Klockner, 2013)
Klockner, A. (2013). Behavior Trees for UAV Mission Management. Dlr.de. Retrieved January 9, 2023, from https://elib.dlr.de/91679/1/kloeckner2013behavior.pdf
36. [19] (Kuckling et al., 2021)
Kuckling, J., van Pelt, V., & Birattari, M. (2021). Automatic modular design of behavior trees for robot swarms with communication capabilites. In Applications of Evolutionary Computation (pp. 130–145). Springer International Publishing.
37. (Ladegourdie & Kua, 2022)
Ladegourdie, M., & Kua, J. (2022). Performance analysis of OPC UA for industrial interoperability towards Industry 4.0. IoT, 3(4), 507–525. https://doi.org/10.3390/iot3040027
38. (Legarda Herranz, 2021)
Legarda Herranz, G. (2021). Evolution of behaviour trees for collective transport with robot swarms. Universitat Politècnica de Catalunya.
39. (Ligot et al., 2020)
Ligot, A., Kuckling, J., Bozhinoski, D., & Birattari, M. (2020). Automatic modular design of robot swarms using behavior trees as a control architecture. PeerJ. Computer Science, 6(e314), e314. https://doi.org/10.7717/peerj-cs.314
40. (Marzinotto et al., 2014)
Marzinotto, A., Colledanchise, M., Smith, C., & Ogren, P. (2014). Towards a unified behavior trees framework for robot control. 2014 IEEE International Conference on Robotics and Automation (ICRA), 5420–5427.
41. (Liu et al., 2018)
Liu, Y., Wang, L., & Vincent Wang, X. (2018). Cloud manufacturing: latest advancements and future trends. Procedia Manufacturing, 25, 62–73. https://doi.org/10.1016/j.promfg.2018.06.058
42. (Mies et al., 2016)
Mies, D., Marsden, W., & Warde, S. (2016). Overview of additive manufacturing informatics: "A digital thread." Integrating Materials and Manufacturing Innovation, 5(1), 114–142. https://doi.org/10.1186/s40192-016-0050-7
43. (Nematollahi et al., 2019)

Nematollahi, M., Toker, G., Saghaian, S. E., Salazar, J., Mahtabi, M., Benafan, O., Karaca, H., & Elahinia, M. (2019). Additive manufacturing of Ni-rich NiTiHf20: Manufacturability, composition, density, and transformation behavior. Shape Memory and Superelasticity, 5(1), 113–124. https://doi.org/10.1007/s40830-019-00214-9

44. (Atanasijevic, 2020)
Atanasijevic, Tatjana. (2020). HOW TO CREATE A SUCCESSFUL USER EXPERIENCE (UX) ON THE HAPPY PATH AND THE UNHAPPY PATH. 10.13140/RG.2.2.10387.71207.
45. (Tadewos et al., 2019)
Tadewos, T. G., Shamgah, L., & Karimoddini, A. (2019). Automatic safe behaviour tree synthesis for autonomous agents. 2019 IEEE 58th Conference on Decision and Control (CDC), 2776–2781.
46. (Thomas & Gilbert, 2014)
Thomas, D. S., & Gilbert, S. W. (2014). Costs and cost effectiveness of additive manufacturing. National Institute of Standards and Technology.
47. (Petch, 2020)
Petch, M. (2020, April 29). 3D Printing Community responds to COVID-19 and Coronavirus resources. 3D Printing Industry. https://3dprintingindustry.com/news/3d-printing-community-responds-to-covid-19-and-coronavirus-resources-169
48. (Petrovic et al.,,2021)
Petrovic, Nenad & Radenković, Maša & Cvetkovic, Stevica & Rancic, Dejan. (2021). Model-driven automated gMock test generation for automotive software industry.
49. (Rodríguez et al., 2016)
Rodríguez, C., Baez, M., Daniel, F., Casati, F., Trabucco, J. C., Canali, L., & Percannella, G. (2016). REST APIs: A large-scale analysis of compliance with principles and best practices. In Lecture Notes in Computer Science (pp. 21–39). Springer International Publishing.
50. (Schiekofer et al., 2018)
Schiekofer, R., Scholz, A., & Weyrich, M. (2018). REST based OPC UA for the IIoT. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), 1, 274–281.
51. (Seiffert et al., 2022)
Seiffert, L., Sczodrok, J., Ghofrani, J., & Wieczorek, K. (2022). Remote technologies as common practice in industrial maintenance: What do experts say? Telecom, 3(4), 548–563. https://doi.org/10.3390/telecom3040031
52. (Shah, 2022)
Shah, A. (2022). Emerging trends in robotic aided additive manufacturing. Materials Today: Proceedings, 62, 7231–7237. https://doi.org/10.1016/j.matpr.2022.03.680
53. (Stock & Seliger, 2016)
Stock, T., & Seliger, G. (2016). Opportunities of sustainable manufacturing in industry 4.0. Procedia CIRP, 40, 536–541. https://doi.org/10.1016/j.procir.2016.01.129
54. (Wang et al., 2019)
Wang, Y., Lin, Y., Zhong, R. Y., & Xu, X. (2019). IoT-enabled cloud-based additive manufacturing platform to support rapid product development. International Journal of Production Research, 57(12), 3975–3991. https://doi.org/10.1080/00207543.2018.1516905
55. (Wolde & Boltana, 2021)
Wolde, B. G., & Boltana, A. S. (2021). REST API composition for effective testing the cloud. Journal of Applied Research and Technology, 19(6), 676–693. https://doi.org/10.22201/icat.24486736e.2021.19.6.924
56. (Zhang et al., 2021)
Zhang, W., Deng, X., & Cai, J. (2021). Intelligent automatic 3D printing machine based on wireless network communication. Wireless Communications and Mobile Computing, 2021, 1–14. https://doi.org/10.1155/2021/1778