NIST Grant/Contractor Report
NIST GCR 23-046

Intelligent Flexible Manufacturing Cell

Ethan Hollingsworth
Eric Holterman

This publication is available free of charge from:
https://doi.org/10.6028/NIST.GCR.23-046

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Grant/Contractor Report
NIST GCR 23-046

Intelligent Flexible Manufacturing Cell

Ethan Hollingsworth
Software Engineer, CCAM

Eric Holterman
Automation Researcher, CCAM

This publication is available free of charge from:
https://doi.org/10.6028/NIST.GCR.23-046

December 2023

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

NIST GCR 23-046
December 2023

Disclaimer

This publication was produced as part of cooperative agreement 70NANB19H083 with the National Institute of Standards and Technology. The contents of this publication do not necessarily reflect the views or policies of the National Institute of Standards and Technology or the US Government.

NIST Technical Series Policies

How to Cite this NIST Technical Series Publication

Hollingsworth E, Holterman E (2023) Intelligent Flexible Manufacturing Cell. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Grant/Contractor Report (GCR) NIST GCR 23-046.
https://doi.org/10.6028/NIST.GCR.23-046

Abstract

Designed around Industry 4.0 based digital technology, flexible manufacturing cells are envisioned to comprise a variety of machines capable of peer-to-peer communication, learning, and adaptation with the end goal of creating a standards-based, brand-agnostic, autonomously managed work cell free of any human influence. In this project, CCAM developed a physical and digital flexible manufacturing cell built primarily on open standards and open-source technologies to address the opportunities and challenges of development of a strong communication framework, as well as the semantics and context to drive individual decision making down to the machine or controller and enable brand-agnostic interoperability. This reference implementation of a full digital and flexible manufacturing cell can be used for the further development of standards and technologies surrounding the digital thread and distributed manufacturing.

Keywords

Automation; discrete manufacturing; flexible manufacturing; Industry 4.0; Kafka; manufacturing; MTConnect; networks; open standards; QIF; robotics; smart manufacturing; standards; STEP.

Table of Contents

List of Figures

Acknowledgements

CCAM wishes to thank NIST, Mitutoyo, Hurco, Cisco, and Simplimatic Automation for all of their help and support during the course of this project.

Executive Summary

The CCAM Intelligent Flexible Manufacturing Cell endeavors to provide a test-bed for accelerating and transitioning emerging digital manufacturing technology from the laboratory to the factory floor. By delivering a system which is standards-based and brand-agnostic we hope to increase American manufacturing competitiveness and encourage small and medium product providers to enter the Intelligent Factory workspace.

Introduction

- Designed around Industry 4.0 based digital technology, flexible manufacturing cells are envisioned to comprise a variety of machines capable of peer-to-peer communication, learning, and adaptation with the end goal of creating a standards-based, brand agnostic, autonomously managed work cell free of any human influence.
- In this project, CCAM developed a physical and digital flexible manufacturing cell built primarily on open standards and open-source technologies to address the opportunities and challenges of development of a strong communication framework, as well as the semantics and context to drive individual decision making down to the machine or controller and enable brand-agnostic interoperability.
- This reference implementation of a full digital and flexible manufacturing cell can be used for the further development of standards and technologies surrounding the digital thread and distributed manufacturing.

Methodology

- The cell contains a CMM, CNC, and mobile manipulator. All devices were locally networked and secured with Cisco industrialized IoT equipment.
- Firewall rules, network isolation, and edge compute devices were utilized at the cell level to ensure that all work was performed locally. The only external access to the cell was enabled by a Cisco firewall and the only external traffic which influenced the cell was the job orders coming from the Digital Factory MES.
- MTConnect was chosen as the data exchange standard of choice for the cell. Apache Kafka was deployed as the publish/subscribe messaging middleware through which all cell communication was routed.
- CCAM's onsite MES system was utilized for the deployment of process jobs to the cell. An autonomous cell supervisor application is responsible for the monitoring of the job queue and distribution of individual process steps to the machine specific driver agents. • Each machine in the cell is monitored, commanded, and controlled by a low level autonomous driving agent responsible for the execution of jobs assigned to it by the cell supervisor application.
- The MTConnect process data coming from the machines is augmented with job specific parts and process data conforming to MTConnect parts and process vocabulary before being sent back to the MES. This enables more robust and granular job and process

- tracking/cross-referencing.
- All applications are running locally at the edge of the machines. This decreases communication latency and increases the reaction time of the disparate equipment.

Results

The major achievements in this project included:

- The development and deployment of a heterogeneous set of secured, networked equipment built around the MTConnect standard of communication.
- The development and deployment of autonomous equipment agents running at the edge, including machine specific controllers, and an overarching cell supervisor application capable of managing the cell's workload and workflow.
- The development and testing of job specific parts and process data objects conforming to the MTConnect standard and capable of being inserted into the MTConnect streams being published from the machines.
- The successful testing and showcasing of the core FMC functionality, including equipment heterogeneity, and cell flexibility based around autonomous job management and execution.

Conclusions and Recommendations

- CCAM is now ideally positioned to continue investigating the best methods and standards for edge communication and control of disparate manufacturing equipment.
- Kafka and MTConnect can serve as robust messaging and middleware services capable of handling large amounts of traffic in a secure, robust, and elastic manner.
- Device agnostic machine control driver agents and an overarching cell supervisor have been developed for the command and control of the FMC cell.
- Future work could involve the exploration of alternative communication architectures for this type of deployment. It would ideally comprise a robust, secure, and pluggable *device agnostic* middleware communication framework capable of handling multiple standards.
- A deeper investigation into the task assignment semantics of a cooperative production floor could also be useful.

1. Introduction and Background

1.1. Manufacturing in the Digital Age

Terms such as “digital thread”, “Industry 4.0”, and “Industrial Internet of Things” (IIoT) are emblematic of the technological advances in sensing, computing, digitalization, data storage, and analytics. At the cell level, Industry 4.0 is pushing manufacturing to become more distributed, intelligent, and autonomous. By utilizing the digital thread, processes can become more easily and more tightly controlled and optimized, resulting in better quality parts, increased productivity, and increased efficiency.

The utilization of these technologies is encouraging research and innovation in what has been termed the flexible manufacturing cell (FMC) of the future [1], [2]. Designed around 4.0 based digital technology, these cells are envisioned to comprise a variety of machines capable of peer-to-peer communication, learning, and adaptation with the end goal of creating an autonomously managed work cell free of any human influence. For example, an autonomous machining cell comprised of a CNC for cutting, a CMM for measuring, and a mobile manipulator cobot for inbound/outbound part movement, could be virtually independent and capable of mass part production once the cell is deployed.

There are still numerous FMC related challenges to be studied and addressed though, including the development of a strong communication framework for carrying data, as well as the semantics and context to drive individual decision making down to the machine or controller level. Some of these challenges can be addressed by standards (e.g., MTConnect [3], OPC UA [4], STEP AP242 [5], and QIF [6]) that provide common semantics, meta-data, and an established baseline for sharing various data types between manufacturing systems. Additionally, it is important to develop an understanding of the collected data in the context of workflow and production orders and how this pairing of data and context could drive intelligent and autonomous decisions at each unit or component. Other aspects of autonomy requirements remain to be explored, including the number and fidelity of data sources, rate of storage, data aggregation, and bandwidth.

For this project, CCAM developed a physical and digital testbed to address these opportunities and challenges.[1] The specific questions evaluated were:

1. How to develop a secure IT/OT environment with micro-service capabilities at the edge?
2. Which technology, data standards, semantics, context, and protocols are best for enabling digital thread technology in a flexible manufacturing cell?
3. How best to increase autonomy, decision making, and situational awareness of the individual units at the cell level?

This reference implementation of a full digital and flexible manufacturing cell can be used for the development of new standards and technologies surrounding the digital thread and distributed manufacturing. It is a self-contained manufacturing cell with a dedicated digital infrastructure and capabilities for running micro-services at the edge. The cell is comprised of a heterogeneous set of equipment that can perform a set of individual process steps and be automatically adaptable to variations in part designs, process flows, etc. The cyber-physical testbed will enable industry

[1] CCAM Flexible Manufacturing Cell

partners, standards organizations, and government partners to develop, test, and demonstrate new technologies, prove functional concepts, and test new concepts that relate to flexible and distributed manufacturing processes. Anticipated use cases for the cell include distributed manufacturing, cooperative processing, refurbishment of complex parts, small part assembly, and seamless or adaptive manufacturing.

2. Methodology

2.1. Physical and Digital Architecture

The development cell contained a Hurco vm10i CNC machine, a Mitutoyo MiSTAR 555

CMM machine, and a custom-built mobile manipulator comprised of a UR5 arm and a MiR 200 mobile base. As seen in Figure 2.1, the cell was locally networked and isolated from the remainder of the CCAM infrastructure by a Cisco ISA 3000 industrial firewall running in ”Transparent Mode” (works at Layer 2 of the IP stack without any need to segment the network up or downstream). A Cisco IE4000 Layer 3 switch with IoX capability was used for connecting the cell elements. A Layer 2 NAT was utilized for routing external traffic to the cell.

A mobile networking cabinet, see Figure 2.2, was constructed for the containment of all digital connections and physical devices. The CNC, CMM and mobile robot were all placed on the local network. Additionally, a Linux based NUC, network attached storage (NAS) drive, and Cisco IC3000 Industrial Compute Gateway were added to the cell network. The IC3000 was initially intended to be the sole edge computing device placed in the cell, but space and memory constraints required the installation of the NUC for additional compute capabilities. The NAS drive was mounted to the CMM, CNC and NUC for convenient

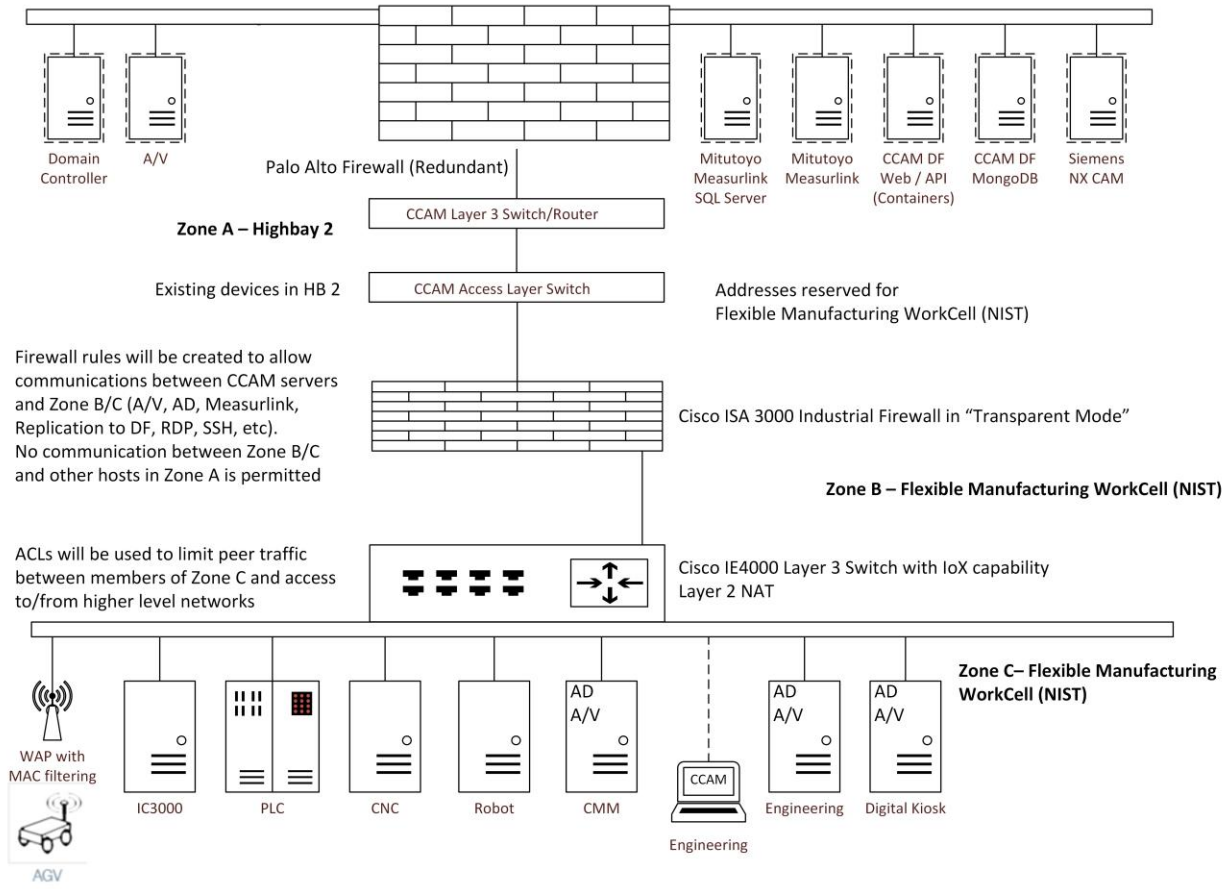

Figure 1: Flexible manufacturing cell integrated with CCAM networks.

transfer of large files and part programs.

The CMM and CNC were positioned in a manner where they could be easily and quickly managed by the mobile manipulator. In Figure 2.3 it can be seen that the robot is capable of removing the part from the front of the CNC and quickly placing it at the CMM. In this layout, the inbound and outbound conveyors would also be nearby. It is of course understood that the cell may not always be so conveniently laid out, a problem that is outside the scope of this research.

Many challenges remain with the adoption, deployment and implementation of standards and technologies associated with Industry 4.0 and the digital thread. Many of these standards promise greater access to data with benefits for improving efficiency, flexibility, process control, and quality control [7]. It was desired that the FMC be designed around a single

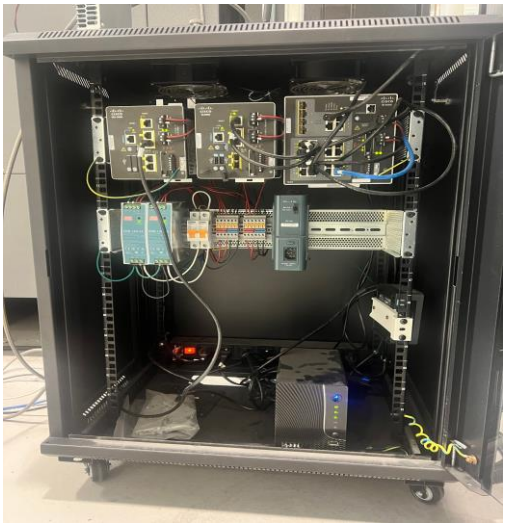

Figure 2: Local network cabinet

Figure 3: Flexible manufacturing cell

data model/standard of communication. For this project, MTConnect was selected as the data standard around which the FMC software would be constructed.

MTConnect provides a standard vocabulary for differing pieces of manufacturing equipment to provide structured, contextualized data, that can be captured and analyzed by nearby control systems [8].[2] Both the CMM and CNC machines in the FMC cell were configured with MTConnect version 1.3 adapters/agents capable of providing the necessary data for the machine control systems to be built around. The mobile manipulator, as an amalgamation of two cobots, was not equipped for providing its data in this manner. The MTConnect version 1.3 agents running on both machines published the data in XML format and without any information about the part being produced or the process being run. Part of the investigation on this project involved what the injection of the parts and process data into the MTConnect schema looked like, and how it could be utilized by the control systems and manufacturing execution system (MES) at the cell.

With the decision to utilize MTConnect as the data standard of choice at the cell, it was also necessary to select a uniform communications system that all devices/agents would adhere to. Apache Kafka is an open source publish/subscribe distributed event streaming

[2] mtconnect.org

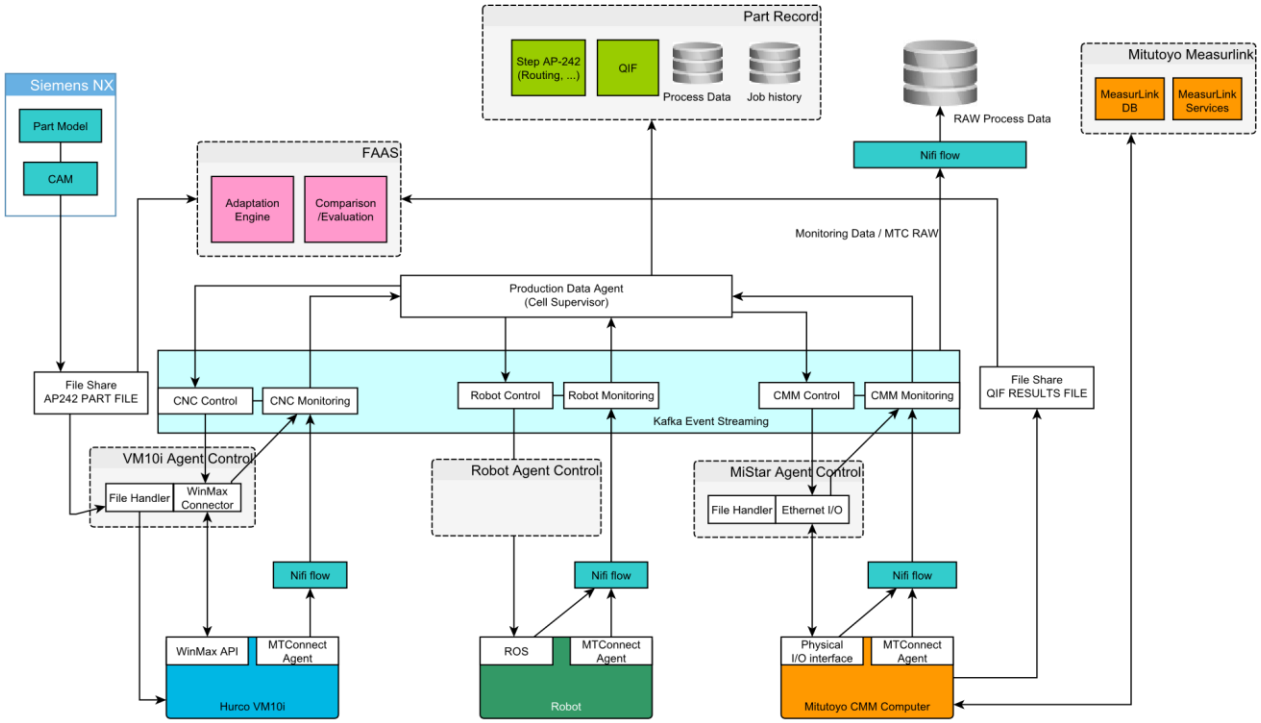

Figure 4: System architecture

platform offering high throughput, scalability, permanent storage, and high availability.[3]

Figure 2.4 offers a high-level view of the cell's system architecture. Figure 2.5 illustrates the various software and hardware components, along with their connections, which comprise the CCAM flexible manufacturing cell.

A description of the most relevant system components now follows.

2.1.1. FMC machines

All devices in the cell must be capable of producing their process data in an MTConnect standardized manner. For data production, the Hurco CNC, and Mitutoyo CMM arrived as MTConnect enabled devices. Both machines produce data in the version 1.3 XML schema of MTC. Additionally, the CMM produces a QIF file for the tracking and examination of process data related to the part.[4] The robot does not produce MTC data and requires a

[3] kafka.apache.org
[4] qifstandards.org

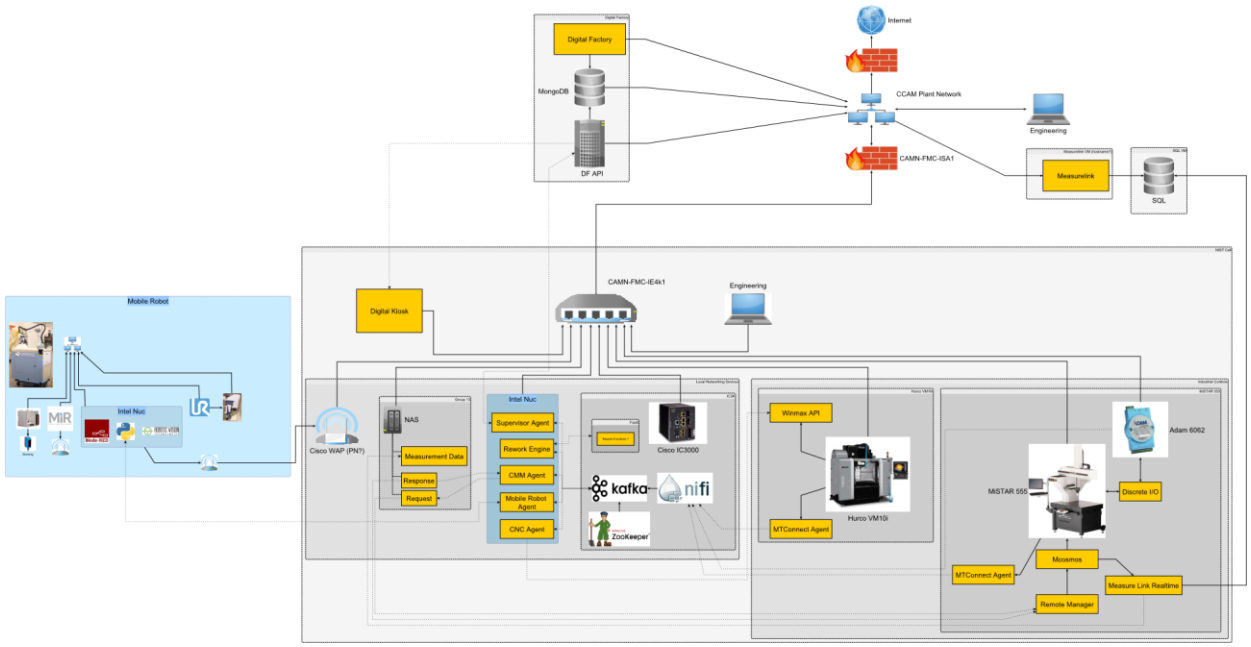

Figure 5: Flexible manufacturing cell components

plugin capable of translating its device information into an MTC standard which can then be published by the MTC agent.

The machines must also be capable of being controlled remotely. The robot is ROS1 enabled and controlled through driver libraries. The CNC provides a remote-control solution developed by Hurco called the Winmax API. This is is a RESTful service running on the machine controller which allows remote control of the machine with functionality such as program load/unload and machine start/stop. The CMM provides a Remote Manager service which allows external loading and starting of measurement programs.

2.1.2. Extract transform load tool

A portion of the digital thread research in this project involved augmenting the base MTConnect stream coming from the machines with additional job specific parts and process data. As the data storage unit utilized in the MES system was a JSON based Mongo database, it was desired that the XML output of the MTConnect stream be immediately transformed into a JSON string. The flow based ETL tool Apache NiFi was utilized due its quickly deployable, secure, and extensible nature.[5]

Flows were set up so that the MTConnect data from each machine was constantly being extracted into NiFi. Once extracted, the XML was then transformed into JSON via a custom script. The JSON formatted MTConnect data was then published into machine specific topics contained within the Apache Kafka broker running at the cell.[6]

[5] nifi.apache.org

[6] This work was performed before MTConnect support for JSON was available. Future projects could potentially replace the XML to JSON conversion with native JSON from the MTConnect agents.

2.1.3. Kafka event streaming broker

Apache Kafka is an open-source distributed event streaming platform" built on a publish subscribe architecture. Kafka offers the ability to write to and read from streams of events, store streams of events, and process events as they occur or after they occur. A fault tolerant, highly scalable broker manages all client connections/disconnections and topics. Each event consists of a key, value, timestamp and is published on a client defined topic.

Each NiFi flow responsible for the extraction, transformation and loading of the MTC data to the Kafka broker, publishes the information to the broker on a machine specific topic that all other devices in the cell have access to. This centralized messaging system allows the easy connection/disconnection of machines from the manufacturing cell. Each new machine can publish its MTC data through NiFi to be added to the broker on its own custom topic which all other devices can subscribe and react to.

2.1.4. Manufacturing execution system

The manufacturing execution system should be responsible for making production jobs available to the cell, and storing back all of the job associated metadata when the job is complete. CCAM utilizes an on-premises MES called the Digital Factory. This is a full stack architecture with a MongoDB backend through which new jobs can be advertised for the FMC cell to run. Though the CCAM MES is on premises, this could just as easily be a cloud deployment from which jobs are published and to which job run data is saved. Kafka streams connectors were set up at the cell level as MES monitoring middleware. When a job is made available in the Digital Factory, it is automatically pulled from the MES and placed into a job specific Kafka topic within the cell broker.

2.1.5. Cell supervisor

The higher level management application running at the FMC cell level is titled the "Cell Supervisor". This application is responsible for the monitoring, management, and state assessment of the entire cell. The job specific Kafka topic is monitored by the application, as well as the state of all the machines in the cell.

The supervisor responsibilities are as follows:

1. When a new part enters the cell, the supervisor queries the DF for the "production order" associated with that part. It is the supervisor's responsibility to assess which portions or "jobs" can be performed inside the cell.
2. The supervisor then assigns each job the cell can perform to a capable and available machine.
3. Each machine is assigned only one job at a time.
4. As each job is completed, the supervisor publishes the updated order with all associated metadata back to the digital factory.
5. The supervisor then waits for a new part or moves on to the next one available.

2.1.6. Device driver agents

The device driver agents are responsible for the control of the machines in the cell. Each machine has a command and control ”Agent” which runs at or near the device. The agent is responsible for picking up and running a job when the supervisor assigns one to it. Each agent, regardless of machine, is driven by a state machine and transition table, Figures 2.6

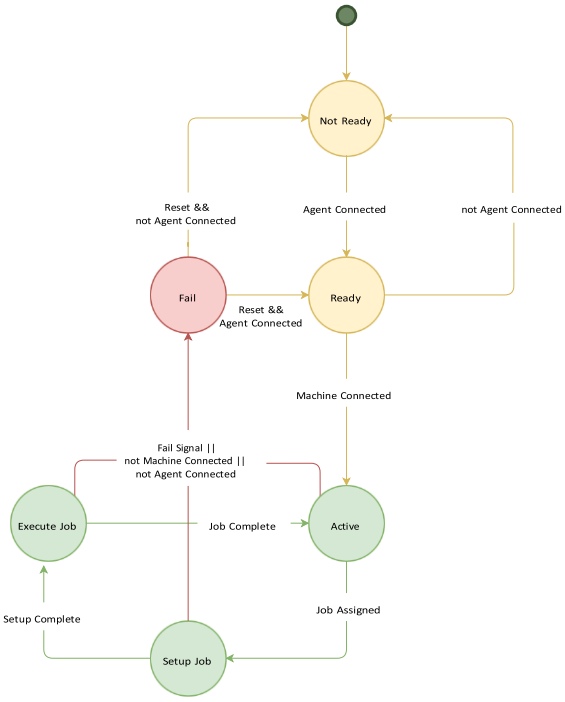

Figure 6: Agent state machine

and 2.7, respectively.

Except for the specific commands required to remotely actuate and control its machine, (i.e., the CMM, CNC, and robot all require different forms of control), the driver code for every agent is the same. Provided that each agent environment file (specific kafka topic, etc) is correctly configured, this enables the easy swapping out of differing machines with the only requirement being that the integrator implement the standard machine specific control functions themselves.

The agent responsibilities are as follows:

1. Always ensure connection to their machine and the Kafka broker. Move into Fail state if either is not true.
2. Monitor their job specific topic. If a job is assigned to them and they pick it up, publish the job as accepted so that the supervisor is aware of the cell state.
3. Place machine into proper state for running of the job and begin executing it. Publish job as executing so that the supervisor is aware of cell state. If something fails moveinto fail state and publish job as failed.
4. When job is complete publish job as complete so supervisor is aware and move back into state such that new jobs can be accepted in the future.

Transition Requirement Conditions for Each Agent					
	Agent Connected	Machine Connected	Job Assigned	Setup Complete	Job Complete
OperatorAgent	Connected to Kafka	Operator UI Connected	Received operator job as READY	Operator prompted to perform job	Operator confirms job COMPLETE
vm10 Agent	Connected to Kafka NAS share access Connected to Winmax API	AVAILABLE in MTC master	Received CNC1 job as READY	Program loaded on Winmax	Machine done running Program completed successfully
Mistar Agent	Kafka connected NAS share access Note: Remote manager should be running on Mistar	AVAILABLE in MTC master	Received CMM1 job as READY	Part detected on table REMOTE.ASK copied to cmm share	REMOTE.RESPONSE written out MTC master showing job completion

Figure 7: Agent state transition table

2.1.7. Adaptation engine

The adaptation engine is responsible for the evaluation of the machined part once it is measured by the CMM. The engine is deployed as a function service which compares the CMM QIF results to the actual part file specifications. If they are determined to be out of tolerance, calculations are performed by the adaptation engine which will move the part back into tolerance. Siemens NX is utilized for the generation of the post-processing instructions required to bring the part back into tolerance.[7]

2.2. FMC Workflow

The order that the cell supervisor receives from the Digital Factory is comprised of five normal flow of execution steps plus one adaptive step. The part flow is inherently contained within the order description as an array of ”job steps”. Each job step also contains the name of an agent capable of performing it. The cell supervisor reads these job steps and delivers them in order to the manufacturing cell agents capable of carrying out the job. The logic of the machine execution and flow can be viewed in Figure 2.8 and is described here.

[7] siemens.com

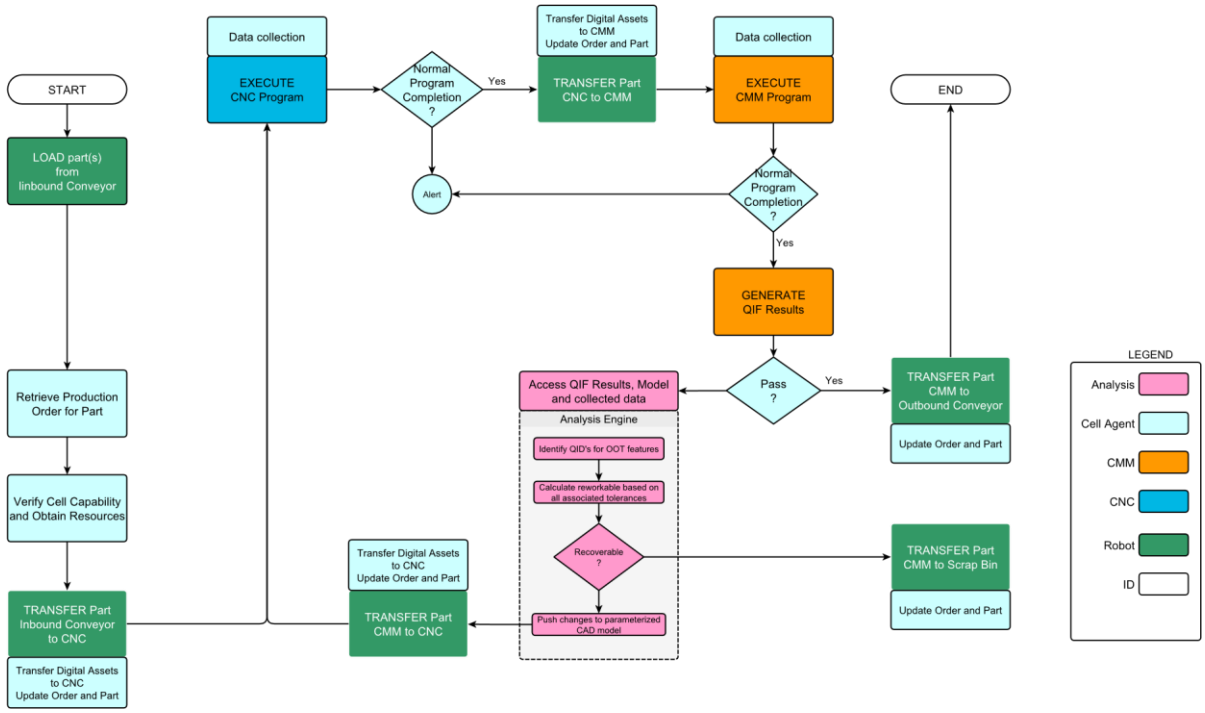

Figure 8: Part workflow in the FMC cell

2.2.1. Move workpiece from inbound conveyor to CNC

This step would generally be performed by a mobile manipulator or fixed base robot. In case of an emergency, however, a human operator could and should be able to be placed in the cell to carry out this job function. The cell supervisor would first evaluate the job to determine what the capable resources within the cell are. This evaluation is done by examining the resources current state as last published within the Kafka broker. If no resource is available the supervisor will wait until one is, or in a more distributed setting, identify a job that could actually be carried out while putting the other one on hold.

Once the resource is identified, the job will be published to Kafka with the resource it is targeted for embedded within the topic. The machine agent with that resource tag will then pick up the job and publish over Kafka that they received the job. They will then remove the relevant parts and process data from the job and embed it into the MTConnect stream coming from the machine, before republishing the entire stream as an ”MTConnect Master” topic which is little more that the base MTConnect stream embedded with the relevant parts and process info. Examples of the job order and parts and process data can be viewed in Figures 2.9 and 2.10, respectively.
The agent will update the supervisor on its job progress throughout the task. Once the workpiece is successfully placed into the CNC machine, the agent will publish the job as ”COMPLETE”. The supervisor will then proceed to the next job step.

```
"startOfJobConfig": [
    {
        "types": [
            "6079edfecb6e990017dc980b"
        ],
        "propertyName": "primaryEquipmentResource",
        "propertyId": "5f22cd4999de2c00134d7182",
        "required": true,
        "objectType": "reference",
        "processPropertyType": "equipment",
        "item": {
            "equipmentClasses": [],
            "pointOfContact": [],
            "_id": "607f3b20d449c640c291a6b4",
            "uniqueName": "CAMP-FMC-CMM1",
            "name": "CAMP-FMC-CMM1",
            "displayName": "Mistar 555",
            "active": true,
            "instanceDerivation": {
                "instanceType": "equipment",
                "parentDef": null,
                "initialParentDefVersion": 0,
                "syncedWithParentDefVersion": 0
            },
```


Figure 9: CMM job order example

```
"PartsAndProcess":[
    {
        "component": "PartOccurrence",
        "name": "PartOccurrence",
        "componentId": "PartOccurrence",
        "Events": {
            "PartId": {
                "dataItemId": "PartId_MongoID",
                "timestamp": "2001-12-17T09:30:47Z",
                "sequence": 1,
                "text": "UNAVAILABLE"
            },
            "ProcessOccurrenceId": {
                "dataItemId": "ProcessOccurrenceId",
                "timestamp": "2001-12-17T09:30:47Z",
                "sequence": 1,
                "text": "UNAVAILABLE"
            },
            "ProcessAggregateId": [
                {
                    "dataItemId": "ProcessAggregateId_WO_UUID",
                    "subType": "UUID",
                    "timestamp": "2001-12-17T09:30:47Z",
                    "sequence": 1,
                    "text": "UNAVAILABLE"
                },
```


Figure 10: Parts and process example

2.2.2. Execute CNC program

This step involves the actual machining of the part. Once the robot/operator has confirmed successful placement of the piece into the CNC, the supervisor will publish the next job step over Kafka targeted for that CNC machine. Upon receipt of the job, the CNC machine agent will embed the job specific parts and process data into the MTConnect stream.

The CNC agent will extract the part program to be run from the job. If the machine is in a functional state, it will remotely load and execute the program. When the machining is done it will publish the job as ”COMPLETE”. If any adverse condition occurs, the agent will publish the job as ”FAILED”, after which the supervisor will set an alert which should trigger emergency intervention.

2.2.3. Transfer part from CNC to CMM

Once the part has been made in the CNC, it must be transferred to the CMM for measurement. This transfer is again done by a robot/operator. The workflow is the same as in step 1. The supervisor identifies a capable entity, publishes the job to it, which is then picked up and executed while the MTC stream is augmented with the parts and process data. When the job is COMPLETE, it is published as such. The agent then moves back into a waiting state, and the supervisor proceeds to the next step.

2.2.4. Execute CMM program

In this step, the measurement of the part is performed by the CMM machine to ensure that the workpiece was cut to tolerance. Once the robot/operator has confirmed successful placement of the piece at the CMM, the supervisor will publish the CMM targeted job, after which the agent will follow the same *receipt→confirmation→MTC augmentation* flow.

Once the part is measured, it will generate a QIF result file indicating whether the part passed or failed. If the part passed, then the agent will publish the job as COMPLETE, and the supervisor

will transition to the final job step. If the part fails, the supervisor triggers the adaptation engine, which calculates the rework ability of the part based on all available tolerances. If the part is not reworkable, it is transferred to the scrap bin by the robot/operator. If it is reworkable, then the part specific G-Code is regenerated in NX, and the supervisor signals the robot/operator to move the part back to the CNC machine. Execution then re-proceeds from the" Execute CNC Program" step.

2.2.5. Transfer part from CMM to outbound conveyor

This is the final step. Reaching this point requires that the part passed all tolerance checks. Once this job step is signaled as COMPLETE, the cell supervisor will assign the appropriate metadata to the job order and write it back to the Digital Factory. If there is a new order containing cell-specific jobs available in the Kafka job queue, the supervisor will pick this up and begin the process again. Else, all components will sit in an idle state until a new job is ready to be run.

3. Findings and Results

3.1. Analysis

There were several architectural hurdles encountered in this project which merited discussion. The original research questions embarked upon in this study and the relevant answers now follow.

How to develop a secure environment at the edge?

CCAM primarily leveraged the existing Cisco IoT edge technology stack for cell securement. As discussed previously, firewall rules, network isolation, and edge computer devices were utilized at the cell level to ensure that all work was performed locally. The only external access to the cell was enabled by a Cisco firewall and the only external traffic which influenced the cell was the job orders coming from the Digital Factory. Similarly, the completed job information was the only data flowing out.

All communication traffic within the cell was directed through the Kafka broker, a secure and robust middleware service. This ensured that no applications or machines directly communicated with each other. The Kafka streams connector responsible for pulling jobs from the Digital Factory and placing them in the broker was designed to be the sole point of contact outside the cell.

One problem encountered with this deployment was that the Cisco IC3000 was sized too small for the application. Since it could not run all the required applications effectively, a far less secure Linux NUC was placed at the cell level for running applications such as the machine agents and cell supervisor. In theory, the network isolation of the cell could still secure the environment even with the unhardened NUC, but this was never tested or explored. The NUC being placed into the cell was an unanticipated need.

Which data standards best enable digital thread technology in an FMC?

When trying to bring a disparate number of machines together, which are all speaking a different" language", there must be a standard transactional format which all devices understand. By leveraging the MTConnect standard, it was hoped that the integration of disparate devices could be easily accomplished. Additionally, by injecting the job specific parts and process data into the

MTC stream coming from the machine, an evaluation was sought as to whether this improved job and process specific contextualized metadata at the end of the production run.

Results on both fronts were mixed. Injecting the parts and process data into the MTConnect stream definitely enhanced the information coming from the machine and going into the MES. However, MTConnect is an open standard that enforces no strict adherence as to how much data the manufacturer inserts into the stream. As such, it was found that there were large differences between the MTConnect stream from the CMM and the one from the CNC. Fortunately, both machines were utilizing the version 1.3 MTC standard, so these differences did not affect the command and control of the cell. However, at the end of the project, the CMM was updated to version 2.0 of the MTConnect standard. This change forced a slight modification of the control for the CMM agent due to the differences between the 1.3 and 2.0 schemas. While not a breaking change, any flexible manufacturing cell attempting to utilize MTConnect as the communication standard of choice would need

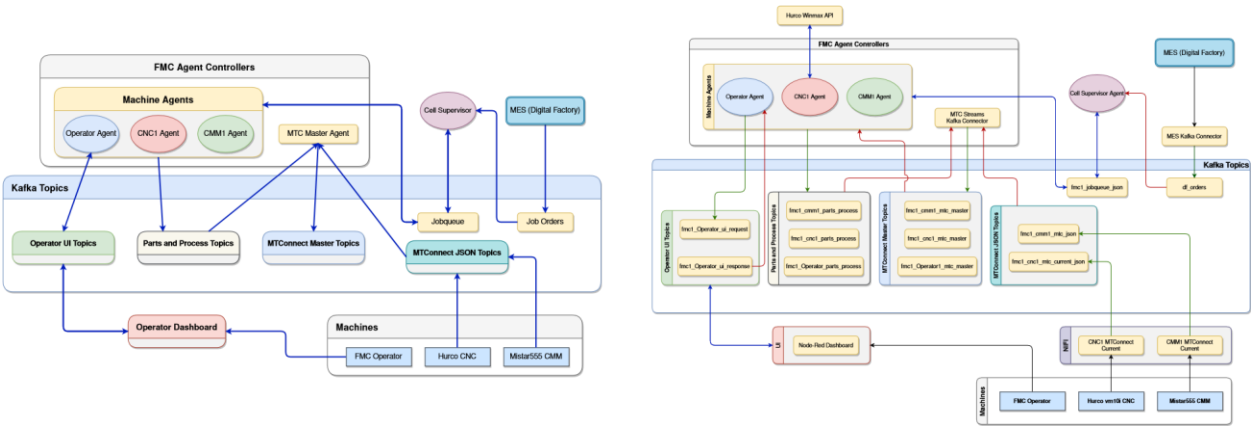

Figure 11: Messaging overview within the cell

to be aware of the potential version disparities.

How best to increase autonomy of individual cell units?

This project implemented an overarching cell supervisor application which ultimately had responsibility for the cell. All machine agents were autonomous, but the management of the cell was left to the supervisor. The other option investigated was a voting deployment, whereby the machines advertised themselves as available and voted on job assignments based on proximity, availability, etc. One reason this route was not pursued is that the number of machines in the cell at CCAM was not conducive to it. Even in the current deployment though, the supervisor has a very narrow defined responsibility and no direct contact with any of the agents as all communication travels through the Kafka broker.

It is this middleware communication framework running at the edge of the device that improves both agent decision making and autonomy. Figure 3.1 illustrates the manner in which the agents and supervisor are all autonomous microservices each responsible for their small piece of the FMC pie which makes the entire thing whole. The broker in the middle of all these services allows them all to be aware of what the other one is doing and to act accordingly. And as an edge deployment, these reactions occur quickly and with little latency, increasing not only the autonomy within the cell, but its safety as well.

3.2. Achievements

The major achievements in this project included:

1. The development and deployment of a heterogeneous set of secured, networked equipment built around the MTConnect semantic standard of communication.
2. The development and deployment of autonomous equipment agents running at the edge, including machine specific controllers, and an overarching cell supervisor application capable of managing the cell's workload and workflow.
3. The development and testing of job specific parts and process data objects capable of being inserted into the MTConnect streams being published from the machines.
4. The successful testing and showcasing of the core FMC functionality, including equipment heterogeneity, and cell flexibility based around standards-based autonomous job management and execution.

4. Conclusions and Recommendations

4.1. Conclusion

In this project, CCAM developed a flexible manufacturing cell to investigate the opportunities and challenges of development of a strong communication standard for improving security, autonomy, and decision making at the machine or controller level. Specifically, a heterogeneous set of equipment was deployed, networked, and secured for an exploration of communication and messaging standards in industry. MTConnect was leveraged as the semantic standard within the cell that all equipment utilized. Apache Kafka was selected as the middleware publish/subscribe broker that all MTConnect, job messages, and agent states were routed through. With the investigation, challenge, and ultimately successful testing and deployment of an autonomous machining cell structured around the MTConnect messaging standard and Kafka middleware, there are several conclusions and recommendations that can be drawn.

First, it was demonstrated that Kafka and MTConnect can serve as robust messaging and middleware services capable of handling large amounts of traffic in a secure, robust, and elastic manner. This makes them an excellent choice for an FMC cell deployment.

Second, the injection of the parts and process data in the MTConnect standard helped to facilitate better job tracking and association. As it is known that MTConnect has been moving in this direction for the past couple of years, proving out the utility of the part and process data in the MTC stream has inherent value.

Finally, a device agnostic machine control framework comprised of low level machine driver agents and an overarching cell supervisor has been developed for command and control of the FMC cell. This framework enables the management of the cell by the supervisor agent through the assigment of jobs to the individual machine agents. The agents were developed in such a way that a different machine could be placed in the cell without major modifications to any of the pertinent pieces. This flexibility could ultimately enable the efficient addition/subtraction of machines and services from a larger manufacturing cell.

4.2. Future Work

With the development and deployment of a CNC, CMM, and robot/operator based flexible manufacturing cell, CCAM is ideally positioned to continue investigating the best methods and standards for edge communication and control of disparate manufacturing equipment. While the standards explored here proved effective, there are other methods that could be investigated as well. In particular, the deployment of a more device and resource agnostic middleware to the cell could be extremely useful. There are other pub/sub standards such as MQTT and OPC UA which could be leveraged in a similar manner if the machines were equipped with the requisite clients/servers. An improved architecture for this type of deployment would involve a robust, secure, and pluggable *device agnostic* middleware communication framework capable of handling multiple standards in a single framework. Running this service at the edge would further facilitate the modularized command and control necessary for disparate, networked machinery.

In that vein, a deeper investigation into the task assignment semantics of a cooperative production floor could also be useful. A higher-level engine with a well-designed task assignment framework could enable more granularized and better-defined methods of targeting machines in a more standardized, efficient, and clear manner.

References

[1] S. Zhang, S. Li, H. Wang, and X. Li, “An intelligent manufacturing cell based on human–robot collaboration of frequent task learning for flexible manufacturing,” *The International Journal of Advanced Manufacturing Technology*, vol. 120, no. 9, pp. 5725–5740, 2022.

[2] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, “Enabling flexible manufacturing system (fms) through the applications of industry 4.0 technologies,” *Internet of Things and Cyber-Physical Systems*, vol. 2, pp. 49–62, 2022.

[3] B. Edrington, B. Zhao, A. Hansel, M. Mori, and M. Fujishima, “Machine monitoring system based on mtconnect technology,” *Procedia Cirp*, vol. 22, pp. 92–97, 2014.

[4] S.-H. Leitner and W. Mahnke, “Opc ua–service-oriented architecture for industrial applications,” *ABB Corporate Research Center*, vol. 48, no. 61-66, p. 22, 2006.

[5] R. Wardhani and X. Xu, “Model-based manufacturing based on step ap242,” in *2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. IEEE, 2016, pp. 1–5.

[6] Y. Lu, X. Xu, and L. Wang, “Smart manufacturing process and system automation– a critical review of the standards and envisioned scenarios,” *Journal of Manufacturing Systems*, vol. 56, pp. 312–325, 2020.

[7] T. Margaria and A. Schieweck, “The digital thread in industry 4.0,” in *International Conference on Integrated Formal Methods*. Springer, 2019, pp. 3–24.

[8] S. Atluru and A. Deshpande, “Data to information: can mtconnect deliver the promise,” *Transactions of NAMRI/SME*, vol. 37, no. 2009, pp. 197–204, 2009.