The Digital Manufacturing Institute

MxD Final Report Project 15-06-01

OSCM: Operating System for Cyber-physical Manufacturing	
Principle Investigator / Email Address	Placid M. Ferreira (E-mail:pferreir@Illinois.edu)
Project Team Lead:	University of Illinois at Urbana-Champaign
Project Designation	15-06-01
UI LABS Contract Number	0220160018
Project Participants:	Northwestern University
MxD Funding Value	$ 1,022,231.00
Project Team Cost Share	$ 1,039,493.00
Award Date	August 30, 2016
Completion Date	August 31, 2018

SPONSORSHIP DISCLAIMER STATEMENT: This project was completed under the Cooperative Agreement W31P4Q-14-2-0001, between U.S. Army - Army Contracting Command - Redstone and UI LABS on behalf of the Digital Manufacturing and Design Innovation Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Army.

mxdusa.org
@mxdinnovates
info@uilabs.org

1415 N. Cherry Avenue
Chicago, IL 60642
(312) 281-6900

mxdusa.org
@mxdinnovates
info@uilabs.org

1415 N. Cherry Avenue
Chicago, IL 60642
(312) 281-6900

TABLE OF CONTENTS

I. EXECUTIVE SUMMARY

Industry 4.0, Cloud Manufacturing, Cyber-physical Manufacturing, Digital Manufacturing and Industrial Internet of Things (IIoT) are emerging modern manufacturing paradigms that seek to exploit information and connectivity infrastructure to collect, distribute, use and leverage manufacturing data and information into actionable decision-making so as to realize smart and

responsive manufacturing organizations and ecosystems. However, it is difficult to put these concepts into practice because of the following reasons:

- Complexity – many interacting problems, many stakeholders, heterogeneity in hardware, operations and practices
- Implementation Involves a multiplicity of technologies (e.g., Manufacturing Technology; System Modeling; Operations; Information Technology; Cybersecurity – to name a few) that demand a high diversity of skills
- High initial investment and commitment with deferred payback that is seen only after at-scale implementation.
- Point solution approaches do not work – they only lead to a tangle of 'band-aid' solutions that are difficult to integrate.

The situation seen in manufacturing today is not unlike that seen in computer technology in the 1950s, where each computerization project involved the design of the computer along with writing of customized software to run the computer to address the applications specified in the project. The introduction of operating systems enabled the separation of the hardware design from the application. Eventually, operating systems became ubiquitous and perform several tasks, including shielding the end-user and application programs for the low-level complexity of the computer's hardware and its operations by providing abstraction models, managing process (program) execution and data files, handling interfaces to the user, external hardware and software.

With the emergence of information-intensive manufacturing, it is necessary to take pause and evaluate the possibility of adapting solution approaches that proved so successful in managing complexity and enabling scalability in the information industry to modern manufacturing. Thus, this project seeks to develop an Operating System for Cyberphysical Manufacturing (OSCM). To enable rapid deployment of software and systems for Digital Manufacturing (or Industry 4.0 or IIoT), OSCM provides the following services:

- Shields the users and software from the complexity of low-level operations/hardware
- Provides abstraction models (specifically, a resource allocation model centered around resources or machines and processes or job) and interfaces for multiple classes of users
- Manages & monitors manufacturing jobs (transactions) and resources (machines).
- Manages interaction with application software (called assets in OSCM) and information (data streams) flow to and from a resource/machine.
- Provide services for multiple classes of users, resources, transactions and apps.
- Provides a REST API for apps and extensions

OSCM is implemented as cloud-based platform for spawning and organizing networks of manufacturing resources (here, we consider computer numerical control or CNC machines, i.e., 3-D printers, CNC Mills or Lathes and other such machines as manufacturing resources) provisioned and managed over the internet. The layered architecture developed consists of hardware, virtualization (edge), operating system (cloud), and network (end-point) layers. In this project, using modern web/cloud development technology, we developed an operating system for manufacturing that services and virtualizes manufacturing resources so that they can be attached to a network. This operating system offers a new and much-needed model for physical manufacturing resources and application software distribution in Digital Manufacturing/Industry 4.0. The platform is designed with a micro-service architecture, that is used for manufacturing

nodes and software applications or 'assets', and implemented in a Mongo, Express, Angular, Node (MEAN) stack.

The services provided by OSCM make it possible for a user to

a) Quickly provide a manufacturing resources/machine with a network presence (i.e., make its configuration information, capability, availability accessible over the internet)
b) Manage a manufacturing resource over the internet (i.e., manage who has access to the resource; attach new services to the resource; manage the schedule for the resource; manage the jobs processed the resource)
c) Create and manage IoT services for a resource or machine (gather data streams from a resource and route them to different desired end-points)
d) Create and manage manufacturing transactions in network of distributed manufacturing resources
e) Gain visibility into unit processes to create and document product digital twins.
f) Attach and configure web-based applications to different resources in the network (i.e., deploy applications like digital twins, schedulers, monitors in the network and attach their services to specific machines in it; manage what data-streams from a resource an application has access to; create new end-point applications for the network)
g) Manage users on resources and the network. OSCM assumes that a web-connected resource or machine will have multiple classes of users and manages the authentication of users and their authorizations and privileges.

With these services, OSCM provides new capabilities for manufacturers with the following use cases:

a) Seamlessly combine operations and information technologies to create Industry 4.0/Digital Manufacturing applications. OSCM services allows manufacturers to build an operational context in IoT data streams and information context in operational decision making, something not supported by COTS IoT or MES (Manufacturing Execution Systems).
b) OSCM allows a small shop to configure and restructure IIoT systems around manufacturing resources on the fly. It reduces IIoT development time from months to hours.
c) With its transaction engine, OSCM permits the rapid building and customization of supply-chain management systems. With OSCM it is possible to communicate technical information, availability and schedules, and costs seamlessly in the transaction. Additionally, it is possible to track progress of jobs in the supply network. Most importantly, with appropriate agreements with vendors, OSCM makes it possible to directly access data-streams from suppliers' resources/machines processing a job, thus extending the digital footprint of a product past enterprise boundaries deep into the supply network.
d) OSCM enables enterprise visibility. Installing OSCM within an enterprise and connecting up different CNC resources to it, allows one a manufacturer to provide its personnel direct access to each machine, its configuration, availability and schedule, the jobs it has processed in the past. This use case can be used to co-ordinate enterprise-wide use of prototyping capabilities and other unique capabilities within the enterprise.

OSCM was successfully implemented, deployed and tested and currently runs (at the following URL: https://oscm-il.mechse.illinois.edu/) supporting a cyberphysical manufacturing

network (CyMaN) **integrates around 40 manufacturing resources ranging from 3-D printers to CNC machines, ultrasonic welders and CVD reactors in around 10 laboratories located at 5 different sites**. Two user facilities continue to use its services to make their facilities available to users. Several labs are currently developing applications specific to their needs over OSCM. For example, Professor Sameh Tawfick of Illinois is developing a sub-net of CVD reactors inside OSCM for large-scale community-based experimentation to support and optimize the growth of 2-D materials. OSCM has been proposed to NSF by a multi-university team as the platform over which to support AI in Manufacturing Research. Finally, two of the researchers who worked on this project have started up a company, Cohesive Manufacturing Inc., hoping to commercialize cloud-based manufacturing services shown possible OSCM.

As a team, we have continued to develop and refine OSCM and integrate it into research and education infrastructure. Our recommendations to MxD and member companies for additional development include the following:

a. MxD member companies can have individuals register and log into the live OSCM platform (at https://oscm-il.mechse.illinois.edu/). Our team would be happy to help you navigate through the platform and answer your questions.
b. Prototype one of your Industry 4.0/Digital Manufacturing applications as an OSCM application. Our development team would be happy walk you through the process. The most difficult infrastructure for Industry 4.0/Digital Manufacturing has already been developed in the platform and these prototype projects can be completed in a few weeks and cost very little money.
c. MxD may consider requesting software development projects to use the OSCM platform. This would allow live versions of these application be made available to your membership for testing and evaluation.
d. MxD can consider making unused machine capacity on its shop floor available to its membership and partners using OSCM.
e. From a modeling point of view, the reference model for OSCM can be extended
    a. Compound Transaction. The current transaction engine only considers unit (i.e., indivisible) transactions. The model needs to be extended to compound transactions that might span multiple facilities.
    b. Extension of Resources. The resource model in OSCM only considers processing resources. Extending the resource model to logistical resources (e.g., transportation, warehousing/storage) and verification/certification resources would greatly extend the applicability of the platform.
f. As a cloud manufacturing platform, the following technical improvements would be important next steps:
    a. The data-streams in OSCM are implemented using TCP sockets rather than web sockets. This was done because of speed and security considerations. Now, if speed is not a major consideration, one should consider using web sockets for scalability and containerization.
    b. Containerization of OSCM using Docker is important, especially for large-scale applications.
    c. Implementation of pub/sub services will make OSCM more flexible with respect to its data services.
    d. Any commercial implementation would have to add financial processing services to the platform.

e. 3-legged authentication will make the platform friendlier for users, while 2-factor authentication will make it more secure.
f. For full-scale operation, the hardware-software platform needs to be optimized. In our live deployment, OSCM is sharing a server with a couple of its assets. This makes the service quite slow. If the live version of the platform is to be maintained, we suggest hosting it on a scalable infrastructure platform like AWS, Azure or Google.

II. PROJECT DELIVERABLES

The following list includes all deliverables created through this project. These deliverables will be referenced throughout this final report and can be accessed on the membership portal in accordance with the rights defined in the Membership Agreement.

#	Deliverable Name	Description	Deliverable Type
1	A Reference Model for Cyberphysical Manufacturing Networks	This is a specification of the different software objects needed for creating a cyberphysical manufacturing network	Word document with an html document for the API.
2	OSCM Software Kernel	This is delivered as a software package on GitLab. Based on the architecture developed, it is integrated with deliverables 3, 4 and 5 into a single executable platform. Additional software code is provided for OSCM clients	JavaScript Software
3	OSCM Software Environment for Network Operations, Administration and Monitoring	Software integrated with deliverable 2	JavaScript Software
4	Scheduling App	Software integrated with deliverable 2	JavaScript Software
5	Manufacturing Capability Search App	Software integrated with deliverable 2	JavaScript Software
6	An Expandable Cyberphysical Manufacturing Network	Accessible demonstration of working software running a cyber-physical manufacturing network available at: https://oscm-il.mechse.illinois.edu	Live Demonstration accessible through given link
7	Syllabus and Short Course Materials	User Manuals are developed in the form of tutorials. A laboratory exercise is also included.	Word/PDF document
8	Business Canvas	Strategies for developing a business using OSCM's capabilities	Word/PDF document

III. PROJECT REVIEW

USE CASES & PROBLEM STATEMENT

Industry 4.0/Digital Manufacturing concepts are complex, involve many different technologies and skill sets and are therefore difficult to implement. Hence, the approach has been to undertake pilot implementations that address specific well-bounded applications. As a result, these implementations are of a one-off nature, do not scale and they are not flexible enough to be easily replicated in different applications. Additionally, many services such as authentication and authorization; meta-data design and processing; security; infrastructure services (data storage, web servers) are expensive to replicate in multiple applications. The investment in a flexible and scalable platform, such as OSCM, provides several sharable services over infrastructure that can be shared by applications. Additionally, using the platform's services, the applications can communicate with each other rather than growing as competing silos. Finally, having such a platform establishes a blue-print for applications and reduces application development lead time.

OSCM, being a very general platform, is expected to support a diversity of use cases. This project targeted the following key use cases:

1. Deploying an IIoT application in an enterprise: I am a **plant engineer** and my team and I have been tasked to ensure that our shop floor is Industry-4.0 ready. This means real-time data from every machine on our shop floor is available to authenticated users and applications for analysis and decision-making. My team is using OSCM because it allows us to instantly bring data-streams from machines on-line using standard MTConnect technologies and it ensure safe handling of this data with its authentication services. Additionally, it allows us to route this data to our custom-developed applications and our enterprise databases.

2. Making a prototyping service available throughout the enterprise: I am a **manufacturing manager** and I run a prototyping service for a multinational enterprise. My facility has a number of unique capabilities and my user base consists of engineers from our global facilities that span seven different countries. I spend large amounts of time explaining our capabilities, answering questions of availability and scheduling, and helping my customers/users track the status of a job. I am evaluating OSCM, because it makes generates a web site for my facility, serves out information about my resources, their capabilities and availability to my users. More importantly, it keeps track of my users, their transactions and the information related to each job. This greatly help in billing, communication with my users and identifying new needs that can be serviced by my facility.

3. Enterprise Visibility: I am a **software engineer** in a manufacturing enterprise. I have been tasked to develop an application that give our production department full visibility of all the machines on our shop floor. I have deployed OSCM because it directly communicates with machine to provide status (on, off, idling, which job is running, etc.) information, it has a calendar service for each machine so it allows planners to easily check machines availability. Additionally, it has a reservation system so they can reserve/block time for important jobs. Its asset interface lets me write customized user interfaces for our production planners.

4. Managing my supplier network: I am a **supply chain engineer** for a large OEM, focusing primarily on managing our supply of fabricated components. I deal with several vendors located all over the US to coordinate component supply all our US manufacturing plants. All technical

information (i.e. part models, manufacturing instructions, inspection information, etc.). This is a source of concern as sensitive design information may easily find its way to our competition. Often, I must check with several suppliers to find one with the appropriate capabilities that I am looking for. To track progress of my jobs at vendor sites, I rely on e-mail and telephone calls. Closing the loop in this manner is difficult and time consuming. For critical components, I am often required to record processing conditions. This is virtually impossible to accomplish and I must identify reliable suppliers and rely on their records. I plan to use OSCM because its transaction engine facilitates technical communication with my suppliers. With some cooperation from my suppliers (i.e., getting them to register their facilities and resources in OSCM), the capabilities, location, availability become searchable in my browser. The transaction engine in OSCM lets me obtain quotes, schedules and initiate manufacturing jobs with my suppliers. Additionally, I do not need to e-mail models files and drawings. They are attached to the transaction in OSCM. OSCM allows me to track the progress of my jobs at a suppliers facility. It and even has the capability to stream process conditions directly to my database when my job is being processed.

5. Improving interaction with customers: I am a **sales and application engineer** for a contract manufacturing company. I deal with many customers simultaneously; each of who have several contracts with us. All manufacturing job information including part model files are communicated by e-mail, making it difficult to view, track, maintain and distribute the most current information. I plan to use OSCM because its transaction engine facilitates technical communication with my customers. It permits me customize my interaction with each customer and share information on job status. Finally, with its user authentication and authorization system, it permits me to release job information to the appropriate personnel on our shop floor.

6. Deploying a cloud-based manufacturing app: I am an independent software developer who focuses on developing manufacturing applications such as cutting-tool monitors, CNC tool-path generation, digital twins for CNC machines, etc. OSCM is an amazing platform because I can, not only reach new customers by depositing my application in their repository, but I can have my applications attach to the data streams of my customers' machines with OSCM's API services. This opens up many new possibilities for innovative manufacturing applications.

SCOPE & OBJECTIVES

This project builds and deploys a cloud-based 'operating system' for manufacturing resources (CNC machines, for example) to enable them to function as a node in a network of manufacturing resources. To accomplish this, the operating system provides services to:

- Manage the visibility and access of other network users to the resource/machine (user services);
- Schedule and manage the operation of the machine (transaction/job services);
- Execute, manage and monitor jobs being processed on the machine (internal services);
- Securely manage and route data streams emanating from the resource to different endpoints (IoT services);
- Decorate resources with 3rd party cloud-based software applications from a curated repository to extend their capabilities, improve their productivity and utilization, and make them easy to access and use (asset services).

The scalable and flexible network of manufacturing resources, users and software applications that results is what we call a **Cyber-physical Manufacturing Network (CyMaN)**

and the cloud-platform whose services enables and manages it is what we call **Operating System for Cyber-physical Manufacturing (OSCM)**

Together, OSCM and CyMaN combine cloud computing and IoT technology with elementary manufacturing operations to provide users with a platform for the rapid deployment of integrated Industry 4.0 applications like e-Manufacturing, Virtual Supply Networks, Enterprise Visibility, Condition-based Monitoring and others.

TECHNICAL APPROACH

To address the scope and objectives of this project, the team undertook the following tasks.

1. OSCM Kernel development: This task involved identifying the basic capabilities/services to be configured around a manufacturing resource to make it part of a the following sub-tasks
    a. Architecture Design and Tool Selection:
    b. Implementation of the Kernel
    c. Implementation of the OSCM API and communication interfaces
    d. Design and Implementation of the Resource Driver Husks (these eventually came to the known as OSCM Clients)
2. Network Operations, Administration and Management: This task involved building cloud-services for operating a cyber-physical manufacturing network and involved the following sub-tasks
    a. Specification and Design of the platform
    b. Implementation of Resource Directory Services
    c. Implementation of Transaction Services
    d. Testing
3. Hardware Integration for Manufacturing Cells. This task involved identify how machines, specifically CNC machines would connect into the OSCM platform. The sub-tasks involved here included:
    a. An assessment of needs and available COTS solutions
    b. Implementation of Resource Drives for specific machines
    c. Testing and Integration of real hardware into the OSCM platform
4. Manufacturing App Development. Two apps were proposed for prototyping OSCM
    a. A scheduling app so enable and demonstrate transaction/job services
    b. Service Locator and Selection App which would be able to locate manufacturing capacity in the network with specific characteristics (location, capability, size of workspace, precision level, etc.)
5. Cyber-physical Manufacturing Network (CyMaN) Demonstration and Testing. The testing consisted of several measures
    a. Workability and User Friendliness
    b. Usefulness and application to Laboratory Environment
    c. Creating a full product within the network.

This work plan was generally implemented but with some minor modifications. They are stated below:

1. Tasks 2 and 3 were merged into a single task because of architectural reasons. The reference architecture for our system suggested that it was not necessary to separate local and network services so we merged them into a single cloud platform.
2. Task 4 which involved scheduling and search in CyMaN. These capabilities found to be so central to the use of the platform. Hence, these services were bundled in with the OSCM kernel. To demonstrate the idea of 3rd party apps functioning over OSCM and within the CyMaN ecosystem three additional apps were developed. There were a) Cloud Monitoring App, b) Digital Twin App and c) A Forming Toolpath Generation App. The first app demonstrates how data can be routed from a resource/machine to an end-point application to support IIoT capabilities. The second demonstrates how OSCM enhances enterprise visibility and the third demonstrates how OSCM can be used to make manufacturing software accessible to relevant users by attaching it to appropriate resources.
3. In task 5, we had planned to demonstrate OSCM operational within an OEM industrial partner. This did not take place because of cyber-security concerns (that were articulated at the last minute). Instead, we performed a serious deep evaluation of OSCM as a commercial product by participating in NSF's I-Corps program and have added a business canvas to the list of deliverables.

PLANNED BENEFITS

This project has produced the following benefits.

a) OSCM is a technology-oriented cloud platform that can be flexibly used. Its focus is on exposing capability and capacity of manufacturing resources (machines), assembling the software tools to make them accessible and usable to a wide class of users, and supporting technologically intensive interaction between users. It is the first of its kind in cloud manufacturing.
b) Existing commercial cloud-manufacturing environments tend to be bid-services (Proto labs) or service-brokering platforms (Maker's Row, Xometry). You submit a design and others cost and bid.
c) OSCM is a platform that integrates manufacturing operations (scheduling, process planning, etc.) with cloud computing and IoT services. This provides an integrated context that helps with decision-making and data analytics.
d) Competing commercial services such as Tulip, ThingWorx, Predix, MindShere provide context-free IoT services. Providing an operational context to collected data can be time consuming and expensive.
e) OSCM is the first cloud-manufacturing platform to incorporate a manufacturing app repository into its architecture. Thus, it creates a target platform for focused manufacturing app development and distribution.
f) By subscribing to apps in OSCM's app repository, machine owners or service providers can attach value-added software enhancements to a machine (e.g., a CAM app for generating CNC code specific to that machine; or a CNC-program simulation app for verification) to make it more attractive and usable for customers.
g) OSCM is capable of serving both static and dynamic data from a manufacturing resource to software apps and users. Thus, it forms a platform for innovation for new manufacturing services and new business models.

h) OSCM can be deployed inside or outside an enterprise. By properly choosing user privileges, one can create a sub-net inside CyMaN that is only visible to users within that organization.

Within the next five years, we expect that to see the following

a) Commercial offerings of OSCM. Two of the lead researchers on the project have evaluate the possibility of starting up a company based on offering services and products that package OSCM-like capabilities to address specific applications like cloud-based digital-twin technology; IIoT for critical operations; supply-chain management and visibility tools; resource/machine capacity sharing apps; and virtual gaging apps.
b) OSCM is already being used in new and unusual ways in academia. Current at least two projects are using OSCM as a platform for planning and conducting distributed community-based experimentation. For example, Professor Sameh Tawfick of UIUC is networking CVD reactors in OSCM so that he can develop a community of researchers to explore the vast parameter space for synthesizing graphene and other 2D materials.
c) We expect the demonstration version of OSCM (at https://oscm-il.mechse.illinois.edu) will continue to scale with addition researchers and evolve for a cyber-physical manufacturing network (cyMaN) to an adaptive, learning network as it couples increasing manufacturing data with machine learning and artificial intelligence tools.

IV. KPI'S & METRICS

The table below outlines the key performance indicators and metrics used to evaluate the success of the project outcomes in comparison to the current state and proposed goals.

1. Domain. The domain of a system is its representation power, i.e. what can the system represent. In the case of an OSCM one dimension of the domain is the types of machines that can be represented by OSCM. Many representation systems for machines focus on a single type of machine such as machining centers or molding machines. In OSCM's implementations, we allow configuration to involve user-defined fields, thus making it possible to accommodate any type of machine. This capability is tested by having users configure CNC machining centers, 3-D printers, incremental forming machines, Chemical Vapor deposition chambers, and ultrasonic welding machines into the OSCM network. A second important dimension of the domain is the transaction/job. Here are goals were more limited. We proposed single customer single provider transactions, but wanted to allow providers tailor the transaction to their facilities' and operational needs. Again, we provided configurational flexibility so that besides the information required by OSCM's transaction engine, a provider could require and additional information. To accommodate different levels of flexibility in the transaction, queues, calendars, and automated scheduling were implemented as part of the transaction engine. All these were tested and were found to be useful. Undergraduate classes and user facilities used OSCM's queues while research labs preferred to use more conventional transaction/job services.
2. Flexibility. Flexibility of the system is a measure of how many different use cases. Since we did not get a chance to test the network in an industrial setting, we demonstrated different use cases in simulated mode:
    a) Manufacturing Visibility Tool: Here we tested the ability of OSCM to make manufacturing resources visible and accessible over the network. The test consisted of bring up unique machines from around 10 laboratories with different unique machines and operational modes (user labs; prototyping labs; research labs). Success was measured by the ability to make the resources specified by the laboratory manager a part of the OSCM network. We were able to bring up all the requested resources. Today the network consists of around 40 resources.
    b) Manufacturing Job, Resource and User Management: In this test, we had different users conduct tests to configure their facilities in OSCM, manage user access, and manage transactions on their resources. In all cases, the labs that participated found the service useful. Many continue to use the OSCM platform. User facilities, in general, required customization of accounting and finances, and hence could not undergo full deployment.
    c) Prototyping of New Cloud Manufacturing Services. Two user facilities prototyped cloud access to 3-D printing labs. The platform is still used by these labs for visibility and machine management.
    d) IIoT Applications. Here the test was to determine how quickly users could extract information from their machines with OSCM services, given a machine adaptor. This test was done in three different research laboratories (one and NWU and two at UIUC). In all cases, it took the user about an hour to begin streaming and capturing data from the machine. OSCM's IIoT services are currently being used to stand up a CVD reactor network for distributed experimentation.

e) Deployment of Web-based Software Applications: The test we conducted here was to provide API service to three different types of users i) an experienced application developer; ii) a graduate student with knowledge of programming but no application development experience and iii) an undergraduate student. All three were able to develop applications that could be added to the OSCM repository. The developer created a cloud-based monitoring application, the graduate student created an application for path-planning for incremental forming, and the undergraduate student was able to create a CNC path generation application for laser cutters. Our assessment was that, while writing cloud-based applications is challenging, integrating them with OSCM was not particularly challenging.

f) As a distributed supply network.. To check this capability of OSCM, the team prototyped a product (a puzzle box) that required 3-D printing, machining and incremental forming. Transactions for parts were released through OSCM to laboratories in NWU and Illinois and the box was easily produced in the OSCM network.

g) Customer Interaction: Several of the laboratories have opted to continue to use OSCM as part of their operations.

3. Scalability: OSCM has been written to scale with number of users and resources in the network. Currently, there are around 40 resources and the network has functioned well (within the limits of the hardware infrastructure) with more than 100 users. Scalability over expandable hardware platforms will require the containerization of the OSCM server. This is easily accomplished given that the server is developed over a MEAN (Mongo, Express, Angular, Node) software stack.
4. Ease of Use. To test the ease of use, a laboratory exercise for two undergraduate manufacturing classes (ME451 CAD/CAM and ME452 Numerical Control) at Illinois was constructed. Roughly, 50 students performed this laboratory exercise and were able to configure a manufacturing resource in OSCM, connect it up to the network, deploy IIoT services, and submit and accept jobs/transactions for the resource in an estimated 2 hours of work allocated to the laboratory exercise.
5. Integrating distributed capacity to complete job. As mentioned in the section of flexibility, the team carried out a demonstration to build a part (puzzle box) whose components were produced in three different labs with three different processes using OSCM networking services.

Metric	Baseline	Goal	Results	Validation Method
Domain: Range of Machines that the System can Accommodate	Typically a single type of machine, e.g. machining centers; forming machines or molding machines.	Any type of manufacturing machines	With the configuration facilities of OSCM, a user can configure any machine to the network	Users were invited to configure any machine that they liked into the network. The network has diverse set of of machines. See description above.
Domain: Ability to accommodate different transactions	Most systems support a single transaction form.	Allow users to completely configure transaction information	Was able to support queues, auto scheduled transactions and manually scheduled transactions.	Users in the network used the transaction engine within OSCM.
Flexibility	Most systems are built with a single application in mind	OSCM is built like an operating system and meant to be application agnostic	OSCM was easily able to support at least 6 different use cases	Prototyping different use cases
Scalability	-	OSCM should be able to scale with number of resources and number of users (albeit with scaling hardware infrastructure)	OSCM was able to scale up to 40 resources without any problems and continues to grow	Functioning open cyberphysical network made available to users.
Ease of Use	-	Users should be able to deploy and use resources/machines on OSCM within 1 or 2 hours.	Around 50 undergraduate students were able to stand up functioning resources within a 2-hour assignment.	Laboratory assignments for formal university classes
Integrating distributed capacity to complete job	Currently most system focus on a single facility	Users should be able to deploy transactions on any resource they have access to in the network	Deployed a product with 3 components in three remote labs and successfully created product	Demonstration of capability

V. TECHNOLOGY OUTCOMES

TECHNOLOGY DELIVERABLES

#	Deliverable Name	Description	Deliverable Type
1	A Reference Model for Cyberphysical Manufacturing Networks	This is a specification of the different software objects needed for creating a cyberphysical manufacturing network	Word document with an html document for the API.
2	OSCM Software Kernel	This is delivered as a software package on GitLab. Based on the architecture developed, it is integrated with deliverables 3, 4 and 5 into a single executable platform. Additional software code is provided for OSCM clients	JavaScript Software
3	OSCM Software Environment for Network Operations, Administration and Monitoring	Software integrated with deliverable 2	JavaScript Software
4	Scheduling App	Software integrated with deliverable 2	JavaScript Software
5	Manufacturing Capability Search App	Software integrated with deliverable 2	JavaScript Software
6	An Expandable Cyberphysical Manufacturing Network	Accessible demonstration of working software running a cyber-physical manufacturing network available at: https://oscm-il.mechse.illinois.edu	Live Demonstration accessible through given link
7	Syllabus and Short Course Materials	User Manuals are developed in the form of tutorials. A laboratory exercise is also included.	Word/PDF document
8	Business Canvas	Strategies for developing a business using OSCM's capabilities	Word/PDF document

SYSTEM OVERVIEW

The Operating System for Cyber-physical Manufacturing (OSCM) is a full stack, operating system to manage manufacturing hardware (machines), manufacturing data (databases), and manufacturing software (applications), in networks of cloud manufacturing. OSCM aims to make manufacturing transactions automated, safe and verifiable across frictionless and scalable networks. It has all the software components for machine owners and customers to register, administer, and access manufacturing capacity in the network through several end-user applications. Because OSCM is conceived on idea of managing and servicing networks of manufacturing resources, it implements services to facilitate:

i. a comprehensive object representation for manufacturing machines that captures their mechanical, controls and operational information,

ii. a communications engine with the sockets and safe protocols to provide virtualized resources with the support of dynamic streams of data between the hardware layer and the end-user applications.

iii. a manufacturing transaction engine to match the logistic and process requirements of jobs to the resource availability and manufacturing capabilities, and

iv. an application user interface (REST API) with a service architecture for owners, customers, and in general, end-user applications in the network to create advanced workflows intertwining manufacturing resources and transactions. These form the technological basis of the operating system.

The implemented system, with the exceptions of minor architectural considerations, matches all the services that were proposed.

System Requirements

OSCM is implemented as a MEAN (Mongo, Express, Angular, Node) full-stack application. It does not require any proprietary software except the free, open-source stack of libraries and frameworks mentioned above, along with a few other frameworks such as Passport (for authorizations) and Winston (for logging) All the libraries needed for OSCM are automatically installed when you clone the project from Git and perform an "npm install". The team made a decision to use no proprietary software libraries because it anticipated the need for DMDII to distribute/use this software. The MEAN stack is a one of the most one of the best, most easily containerized, basic software stack used for cloud application development. More importantly, uses the same programming language (i.e. JavaScript) for the front and back ends of the stack. Given the small development team, this was an important consideration.

System Architecture

The goal of making manufacturing transactions automated, safe, and verifiable requires the operating system, on the one hand, to manage the connections to the resources and capture their hardware characteristics, and, on the other, a means by which end-user applications can access this information to formulate transactions given the constraints of different resources. For example, users in OSCM use CAM-registered tools (cloud or desktop) in the ecosystem to generate the trajectories for a given geometry and manufacturing process. From the CAM application, they set job deadlines and use manufacturing setup information, to match the operational and logistic requirements of the job against a database of resources and select one to execute the manufacturing transaction. To make this transaction verifiable, the provider sets end-user monitoring tools -available at job execution- to create digital twins and display relevant data form the controller and potentially external sensors attached to the system. This workflow requires the operating system to share specific information on the resource to end-user applications to generate manufacturing paths with allowable process parameters, post jobs to it, and route a stream of dynamic data to feed several real-time monitoring applications in parallel.

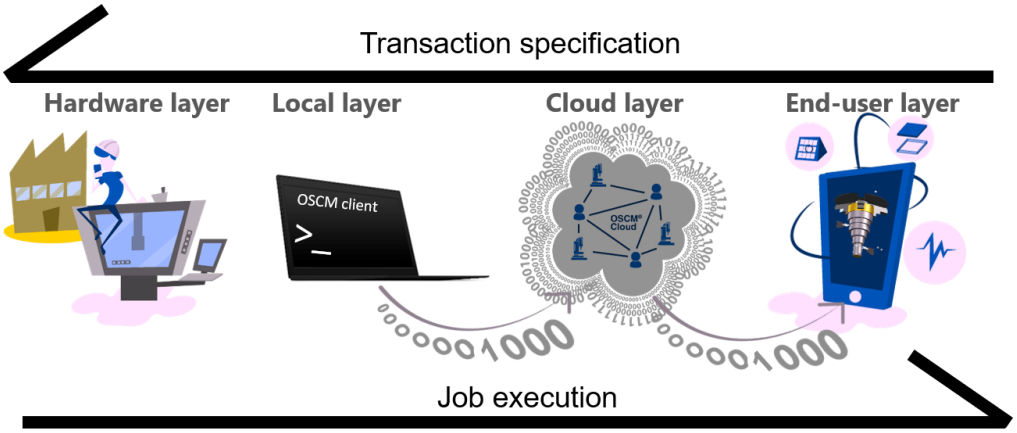

Figure 1 Layers of OSCM. Manufacturing machines at the hardware layer are connected to the software components at the local layer necessary to stream data from the machine controllers to the cloud. End-user applications in the ecosystem use this data to aid the generation and monitoring of manufacturing transactions

At the simplest level, the operating system provides a transition of manufacturing resources from hardware, to software, to services, accessible via user-applications. This is achieved through a cross-platform architecture starting from the physical system (the hardware layer), followed by native applications (local/edge layer) to gather and transmit data from the resources to a cloud server (cloud layer), and ending at a set of end-user applications (end-user layer). The cloud server can be thought of as a hub of virtual resources and streams while the end-user applications let users in the system administer the virtual resources and engage in manufacturing transactions. The general schematic of this architecture is illustrated in figure 1.

Figure 2. shows a schematic picture of the abstract framework of the operating system with the principal model abstractions and their interaction throughout the layers, starting from the hardware all the way to the end-users. In what follows, this section describes the underlying concepts and strategies used in the design of the software components and abstraction models of the operating system.

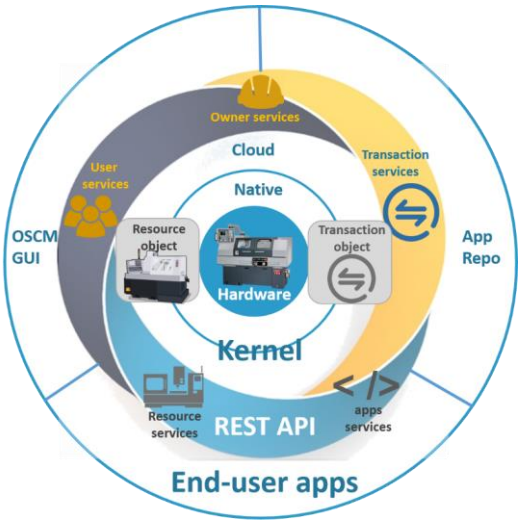

Figure 2. Detail architecture of OSCM. The operating system is centered on the manufacturing resource. It consists of the native, cloud and application layers, interacting direclty with the underlying hardware through the resource and transaction abstract models. The kernel offers the abstractuion to hide low level hardware details from the end-user applications by means of the aplication user interface.

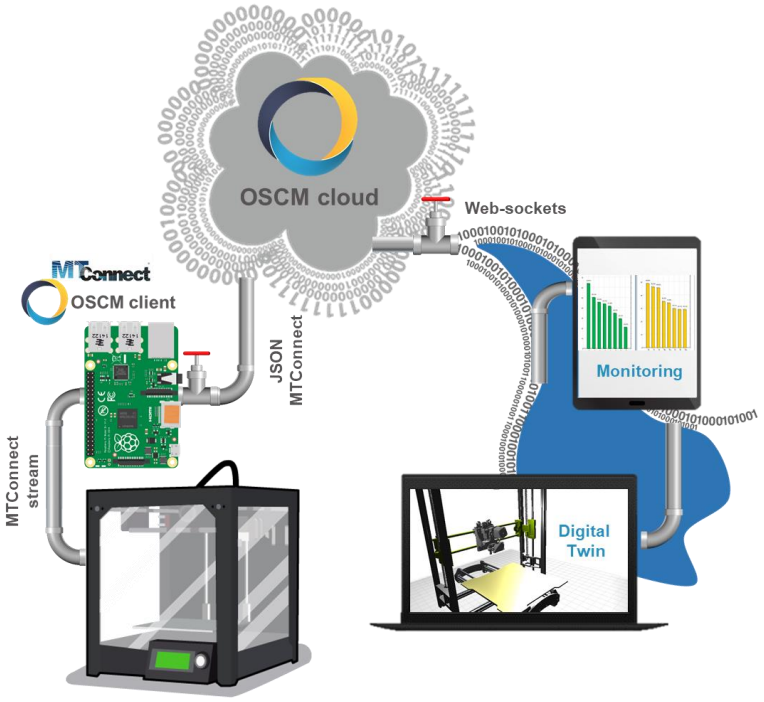

Figure 3. A generic 3D printer connected to OSCM. The MTConnect adapter developed inside Octoprint request position and state data from the printer's firmware. The adapter pushes the data to OSCM client in the raspberry Pi which authenticates the user with OSCM cloud. OSCM cloud pulls data form OSCM client, stores it locally and creates a web sockect connection for third party applications

Native layer

The native layer consists of the software components necessary to connect the controller of a physical machine and any services available sensors, network controller, and programmable logic controller, to the cloud layer. This is implemented with a dual combination of MTconnect[1] adapter technology and OSCM client software. Figure 3 illustrates the connection of a generic 3D printer (compatible with Octoprint[2]) to the operating system. In this example, a Raspberry Pi computer connects to the firmware of the printer via Serial-USB communications, through Octoprint. Octoprint was extended with an MTconnect-adapter plugin that queries the state of the controller, periodically, and serves a connection to the OSCM client.

[1] The MTConnect standard offers a semantic vocabulary for manufacturing equipment to provide structured, contextualized data with no proprietary format
[2] Open source server with a web interface for your 3D printer.

MTconnect adapters are hardware dependent. They collect machine data from the NC controller and sensor data-acquisition software, time-stamps it, and format it as streams, parsed by OSCM client. Adapters are provided by vendors or developed (with native development tools) to interface with the APIs in the hardware layer and open a TCP-socket connection with an agent, in this case the OSCM client. They drive the communications by pushing new events of the stream to OSCM client periodically.

The OSCM client collects data from the adapters and maintains resource state information. This program is hardware agnostic and responds to periodic state request from OSCM cloud. Its implementation over conventional agent technology is justified by the need of program, at the native layer, with an architecture services for authentication, adapters, and resource communications, capable of producing streams in a format familiar to the OSCM cloud. OSCM client is configured to different adapters via a Device.xml file (provided by the adapters) and to different resources via client configuration files (downloadable from OSCM cloud). It authenticates users connecting resources to network by exchanging credentials and JWT-tokens[3] via HTTPS[4] protocol with the cloud layer and retrieving authentication tokens. Authenticated user-resource pairs result in two TCP-socket client connections, one to the adapters and the other to OSCM cloud.

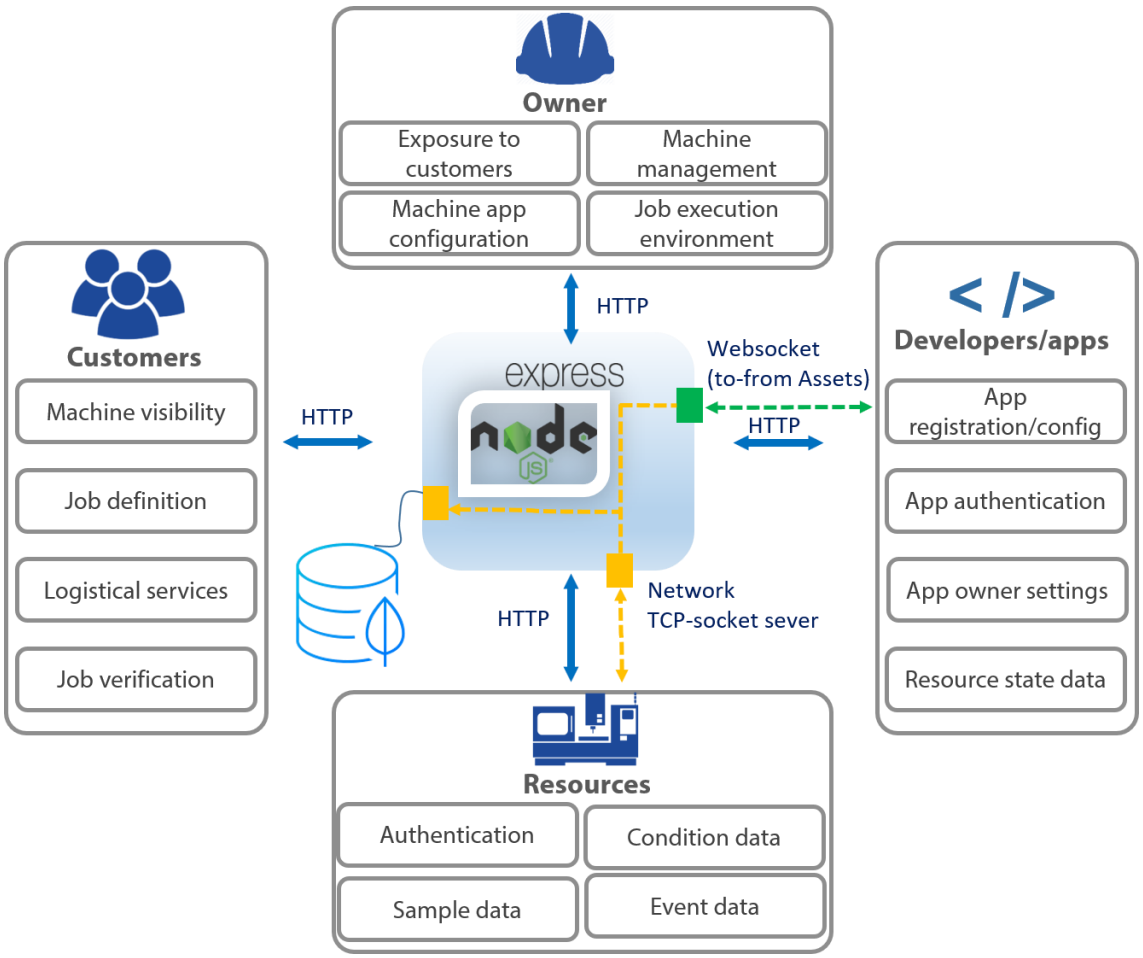

Figure 4. OSCM Cloud interfaces.OSCM cloud intefaces with the resources via HTTP REST API and network TCP-socket connection; with custmers and faciliy owners via HTTP REST API; and to developer applicatiosn via web-sockets and HTTP REST API.

[3] Open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

[4] Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP, the protocol over which data is sent between your browser and the website that you are connected to

Cloud Layer

OSCM cloud is at kernel of the operating system. Implemented as a Node.js[5] and Express.js[6] back-end server with a MongoDB[7] database, its architecture is based on the resource object model and the transaction operation (see figure 2), both accessed through different server interfaces. The interfaces, described in figure 4, are a TCP-socket server connection facing the physical resources (OSCM client); and an HTTPS Restful API with web-socket server facing end-user applications. The abstract models of resources and transaction are a bridge between end-user applications and the actual resource data obtained at the hardware level. In what follows, we described the model abstractions and interfaces at the cloud layer.

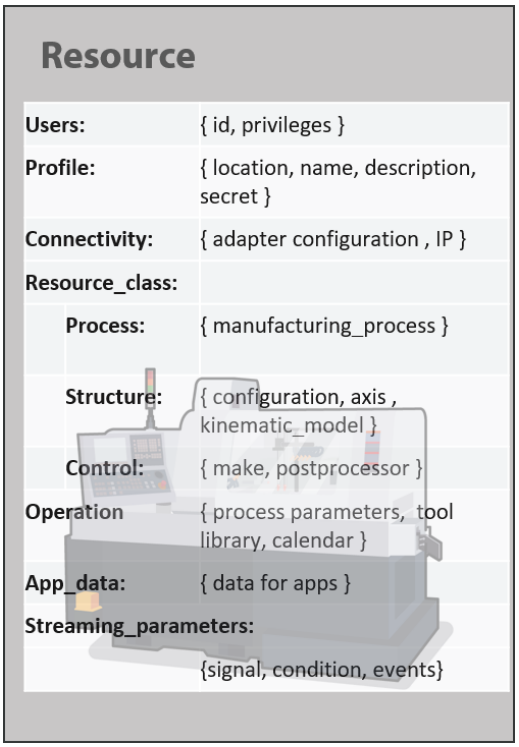

Figure 5. Simplified object model of a resource. ***Users*** *in the resource can be owner, administrator, operators, or guests.* ***Connectivity*** *has information to generate the connections between OSCM client and OSCM cloud, and OSCM client and the MTConnect Adapter. Every resource is an instance of a* ***resource class*** *in OSCM cloud. Classes have information about the manufacturing process, configuration of the mechanical system and characteristics of the controller.* ***Operation*** *has the allowable manufacturing parameters (constraints) for users of the resource.* ***Application data*** *are data used by third-party applications to render a service over the resource, for example a CAD model or calibration table. Finally, streaming has the type of signals (sample, conditions and events) and frequencies streamed from the resource.*

Resource model

A strong resource model, at the cloud layer, is key to OSCM. The resource model is the fundamental object of the kernel. It is used to instantiate software copies of physical systems, transforming them from hardware capabilities to manufacturing services. This broadens the impact/extends the reach of a resource by sharing its copy across a variety of end-user applications, aiding the specification, execution, and monitoring of automated manufacturing transactions. *A comprehensive resource model is key to making resources participate as nodes across networks of cloud manufacturing rather than limit them to be hardware objects at the end of the manufacturing transactions*.

Resources in OSCM cloud are the object representation of virtualized physical systems. Figure 5 shows the simplified object model of a resource. They are instantiated from the system-resource classes by authenticated users in OSCM and stored in the application database. Classes correspond to different manufacturing processes and have specific information on the resource like dimensions, process, and controller parameters. For example, a user can register a three-axis milling machine, specify its axes and workspace dimensions, maximum cutting feed and depth of cut, controller make and postprocessor parameters. This information is useful to end-user applications to instantiate transactions and render services over the resources.

All resources hold information about their users and privileges. Users in the operating system can register resources as owners or be invited to existing ones as administrators, operators or guests. Guests are customers in the network with limited access to some of the software or hardware tools on the resource, necessary to complete a manufacturing transaction. Operators can access NC programs and load them on to the machine controller for execution. They can change the status jobs (waiting, executing or complete) and attach notes pertaining to their execution. Resource administrators have access to all the resource operator tasks. They can manage the configuration of the resource, select available tooling and standard operating protocols (SOPs), and register assets. Additionally, they have privileges to pricing, scheduling and job queue management. Finally,

[5] Environment to build server-side applications. Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

[6] Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

[7] a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemata.

resource owners have all resource administrator privileges. The can manage all personal in the node and network configuration parameters for registration and deregistration, and credential management.

Resources aggregate static information with profile, connectivity, and operational parameters. This information is set by the owners/administrators and includes pictures, encrypted credentials for authentication of OSCM client, calendar object, MTConnect adapter configurations, process parameters, and tooling among others. Finally, resources have an entry for dynamic information to configure signals and frequencies for streaming data like sample, events, and conditions, from the hardware layer.

Transaction object

A second important abstraction in the operating system is the transaction operation. Transactions are objects created from the manufacturing agreements between customers and providers. They can be simple objects detailing information of a job in a single resource, or complex ones with the array of jobs in advanced manufacturing supply chains. The transaction object must have information on the times and status of the jobs as well as on the end-user applications made available to the customers for the specification and verification of the jobs. The development of a strong transaction engine is fundamental to the ability of the operating system to support complex workflows. The resource model and the transaction operation form the fundamental building blocks of this operating system.

Transactions are instantiated by authenticated users, in end-user applications and OSCM Graphical User Interface (OSCM front end GUI), after matching the logistic and process requirements of a job to a resource or a facility. The current model of transactions has the basic information that ties together customers and resource owners, on manufacturing jobs. Figure 6 shows the simplified object model of a transaction. When a customer requests a transaction on a resource or facility, (s)he proposes the process, setup instructions, critical delivery schedule times for the job, and attaches the manufacturing files to it, stored at the server and tagged with the transaction id. At any point of the agreement, the resource or facility owner can deem the transaction requested, accepted, completed, processing or cancelled.

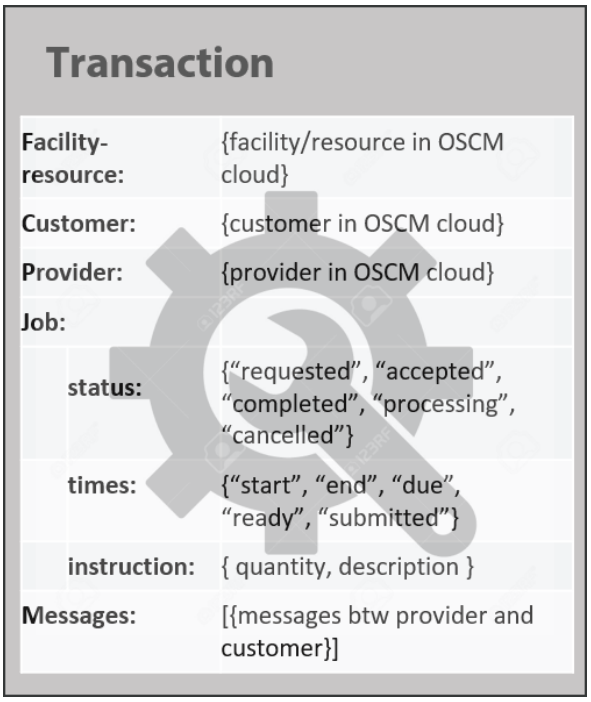

*Figure 6. Simplified object model of a transaction. The **Customer** initiates the transaction with the **Provider.** A transaction is defined by a **job** with a **status** assigned by the provider. Status correspond to the state of the transaction at any time of the agreement. A job has different **times** breaking down the transaction into its processing and setup times. Finally, the transaction stores a possible exchange of messages between customer and providers*

Communications engine

The challenge of virtualizing a machine goes beyond the problem of structuring a resource model. There is a need for a proper communications engine – bridging the local and cloud layers -- for the management of data-streams, as various applications compete for the attention of the physical system. The engine must tie the emanating data of a resource to its virtual representation in the cloud, provide the right semaphores for resource conditions and events, and make sure that end-user applications gets quality sample data in near real-time. It solves the problem of time aligning the data at all stages of the communications pipeline to allow end-user applications to create operational contexts that correlate data from multiple sensors the controller in a resource. This is the technology that makes it possible for the operating system to spawn digital twins, attach virtual controllers for simulation applications, and enable new schemes of cloud-centralized control in computer numerical control.

The communications engine is implemented with the net[8] framework to create a server for the TCP-socket with OSCM client, and the socket.IO.[9] a framework to enable real time, bidirectional web-socket communications between the applications and OSCM cloud. The TCP-socket interface handles real-time data-streams from physical resources including controller and sensor data. After a client has authenticated a user-resource pair (exchanging credentials and JWT-tokens via HTTPS), the communications engine matches the new connection with its corresponding virtual resource and serves a web-socket stream for it. The stream is maintained by periodically pulling and pushing states (a JSON object) from OSCM-client to the web-socket. End-user applications can authenticate with the resources in the cloud (also by exchanging credentials and JWT-tokens via HTTPS) and tap into their streams by leveraging the client side of the web-socket interface served by OSCM cloud. Web sockets support high-speed data transfer for browser applications with low communications overhead. They allow end-user applications to access streams without having to poll the server for a reply or going through the database, avoiding the overhead of HTTPS. The architecture or the communications engine for stream data management allows a single resource to feed multiple end-user apps through its virtual representation in the cloud.

HTTPS restful API sever

In addition to managing the interfaces to the physical resource, the operating system must provide a stable, consistent way for applications to exchange information with the kernel. This is achieved through an application user interface (API), the software hooks to interface resources and transactions with end-user applications in the ecosystem. This is the first cloud server API architected around the manufacturing resource. The RESTful API of OSCM cloud is centered on the resource object and assembled in software calls or services tailored to the different players in the operating system, namely, services for customers, owners, resources, and end-user applications. Figure 3 shows the basic services of OSCM cloud API. These are HTTPS calls like post or get that allow end-users apps to create, maintain and administer virtual resources in the network.

Customer services are the API calls for authenticated users in OSCM to send jobs and book time in available facilities and resources in the network. These calls allow customers create jobs and transactions and communicate with providers. On the other side, owner services are the API calls for authenticated users in OSCM to manage the facilities and resources. These calls allow owners to configure their machine information (like technical specifications, tooling, adapter configuration, etc.) and manage the transaction in their resources. Developer-apps services are the API calls to register and authenticate third-party applications in OSCM cloud. These services handle the application sessions with OSCM cloud as well as the resource-data-streaming services. Authenticated third-party applications in the ecosystem use developer-apps services to create application-dependent resource data like tables, cad models, etc.

OSCM API has the software hooks to create transactions, linking users to resources in the network. Like a system call interface in a computer operating system, end-applications communicate with OSCM API to access virtual resource properties like calendars or configuration information, allowing them to render different services. The security layer of the operating system relies heavily on this interface as OCSM clients and end-user applications are required to authenticate the with virtual resources before pushing or pulling resource data to the cloud. OSCM API is how software developers can extend the ecosystem of application in our operating system.

End-user layer

The end-user layer of OSCM is key to a revolutionary model of machine as a service (MaaS). Given the current trend of automation and data exchange in manufacturing technologies, OSCM has a huge potential as a platform for applications in cyber and cloud-manufacturing and, virtually any CAM application, to enable software solutions that link their output directly to a resource/machine. This model is already been applied to link CAD

[8] The ***net*** module provides an asynchronous network API for creating stream-based TCP or IPC servers
[9] Socket.IO is a JavaScript library for realtime web applications.

tools with 3D models of hardware. OSCM enables a service model where the information that gets to the provider from the manufacturing process is much more specific than conventional STL files or technical drawings.

OSCM front end GUI

The end-user layer is the set of applications that provide and interface between the user and the kernel (OSCM-cloud). The current implementation of OSCM has a graphical user interface (GUI) that allows users to configure and manage their resources as well as to create and verify manufacturing transactions in a marketplace-like browser application. OSCM-Cloud GUI is available at https://oscm-il.mechse.illinois.edu and implemented using the Typescript-based open-source front-end application platform Angular[10]. It is programmed around the manufacturing resource with tools for setting calendars, blocking times, accepting transaction-requests, managing users, and configuring end-user applications in the ecosystem. Additionally, it implements a queuing system to handle jobs in facilities as well as an accounts engine to link the transaction to specific customer accounts in the facility.

OSCM end-user applications

Programmers in OSCM can develop native and client-side applications in any programming language that support the HTTPS protocol. The applications are paired with individual resources/machines at the resource-application repository. These applications are micro-services, hosted independently from OSCM cloud, that authenticate with resources and access their services through OSCM RESTful API. Owners configure them to the specifics of resources, allowing their users to participate in workflows for the generation, execution and validation of manufacturing transactions. The micro service architecture allows the operating system to be extended without growing the complexity of its kernel.

Presently, OSCM has the prototypes of a digital twin application (https://digitaltwin.mechse.illinois.edu), and a monitoring application (https://monitoring.mechse.illinois.edu), and a trajectory generation application. The monitoring application assembles configuration and state data from OSCM-client and serves it through a dynamic html interface. The trajectory generation application creates micro-forming machine-tool paths for a research setup. The Digital Twin application takes controller information from OSCM client and drives a 3D model of the machine. The 3D model files of the digital twin are created with the help of an Autodesk Fusion 360[11] add-in, which converts a machine assembly into a package of .stl files and a .json file. The .json file has the kinematic structure and coordinate transformations of the mechanism. Owners upload the 3D model to the digital twin application and pair it with a resource in OSCM GUI. Authorized users can then launch the digital twin, from the resource-application repository, to monitor jobs. They are redirected to a browser window where the digital twin application authenticates with the resource, pulls the 3D model from the application server, and loads it to a webGL[12] interface. This application connects to a web-socket stream from OSCM cloud that feeds the position of the joints to a kinematics engine which computes and displays the position of links in real time. The Digital Twin is particularly interesting as it assembles properties from resources in OSCM cloud like 3D models, kinematic models, and dynamic streams to render a manufacturing service in real time.

FEATURES & ATTRIBUTES

OSCM, as implemented has the following features and capabilities.

a) OSCM is conceived as first 'operating system' for cyber-physical manufacturing.
b) It enables the exposing of exposing of capacity and capability of a manufacturing resource and provides user, job, resource and software asset management services.

[10] AngularJS is a JavaScript-based open-source front-end web framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications.
[11] Fusion 360 is a cloud-based CAD/CAM tool for collaborative product development that combines industrial design, mechanical engineering, and machine tool programming into one software solution.
[12] WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 3D and 2D graphics within any compatible web browser without the use of plug-ins.

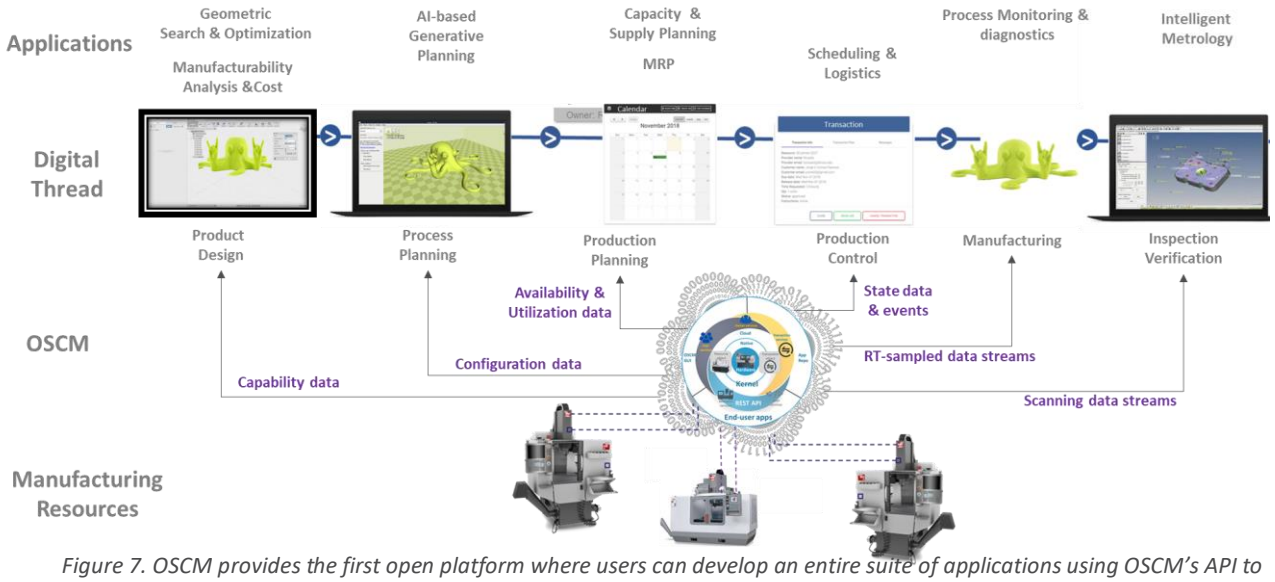

Figure 7. OSCM provides the first open platform where users can develop an entire suite of applications using OSCM's API to support a product realization digital thread

c) It creates a platform not a point solution for Digital Manufacturing or Industry 4.0.
d) It has enabled the standing up of the first operational Cyberphysical Manufacturing Network (CyMaN)
e) It has been demonstrated to enable interaction between multiple classes of users (machine owners, customers, software apps, machines)
f) It manages multiple classes and types of information ranging from near real-time data streams and video to static information services
g) It securely manages and routes data streams from a machine to different endpoints
h) It is extendible through the addition of new network nodes, users and services
i) It provides close interaction between resources and 3rd party cloud-based software apps.
j) It integrates manufacturing operations with IoT services.
k) OSCM is the first open deployment platform for cloud-based manufacturing software app developers we envision that OSCM can be used as the engine to support the entire product realization digital thread as shown in Figure 7.

TARGET USERS & MODES OF OPERATION

OSCM target end-users are enumerated in section *III Project Review: Use Cases and Problem Statement*. The reader is referred to this section for a detailed description of target users and different modes of operation of the OSCM platform. In addition, from a deployment point of view the following is possible.

a) Only the backend of the platform can be deployed and used through the API. The deployer can construct his/her own front-end to interact with the OSCM kernel. An html document providing a complete documentation of the API is given in *Deliverable 1*. This can be used as a reference for such a project.
b) The entire OSCM platform is deployed to replicate the functionality of the live system at Illinois. Here deployer copies clones the software directory (*Deliverable 2-5*) and follows the installation instructions given with the software. Since we use the node package manager, installation is a single command. Installation is essentially a single. The tutorials and labs given in (*Deliverable 7*) can be used to train users, depending on their roles in the system.
c) A prototype system is maintained at Illinois. A user can log in and create their own private sub-network of resources and users.

VI. INDUSTRY IMPACT

OSCM is the only cloud platform that is open, expandable (through its API), resource/centric that allows facility and machine owners to bring their manufacturing resources/machines on-line and interact with their customers through a cloud-based transaction engine. OSCM is the only platform that merges operations with IoT in a cohesive and seemless manner. As such it is relevant to both OEMs and SMEs in the transportation, aerospace, semiconductor and heavy engineering industrial sectors.The current implementation of OSCM can be configured to use as the following:

a) Enterprise visibility platform.
b) IoT platform for manufacturing
c) Service (e.g., specialized machining or prototyping) platform for an enterprise
d) Manufacturing application repository
e) Kernel for a condition-based maintenance system
f) Supplier network management platform
g) Manufacturing customer management platform
h) Manufacturing resource management platform

Having being designed as an operating system, this platform can be configured and/or extended very easily to support most manufacturing applications.

VII. TRANSITION PLAN

TRANSITION CHART

The transition chart provides a catalog of all of the project deliverables and their respective transition route. Deliverables can transition through deployment at an industry member's site, as an educational reference or through a commercialization effort. Each of these transition routes are detailed in the Transition Summary section below.

#	Deliverable File Name	Technology Integration	Education	Commercialize
1	Deliverable 1: (a) Reference Model for Cyberphysical Manufacturing (b) OSCM API	X X	X	X X
2	Deliverable 2-5: (a) OSCM Software	X		X
3	Deliverable 6: (a) Cyberphysical Manuf. Network (CyMaN)	X	X	
4	Deliverable 7: (a) Tutorials (b) Labs		X X	
5	Deliverable 8: (a) Business Canvass			X

TRANSITION SUMMARY

Deliverable 1 *– Reference Model for Cyberphysical Manufacturing*

- Technology Integration: MxD has a working facility consisting of two 3-D printers hooked up to the OSCM Cloud at Illinois to demonstrate the implementation of the reference model for cyberphysical manufacturing.
- Education: The document describing the reference model is used by Illinois for its advanced undergraduate classes in manufacturing (specifically, ME 451: CAD/CAM and ME 452: Numerical Control) to introduce students to digital manufacturing and cloud platforms for manufacturing.
- Commercialization: The reference model developed here forms the blueprint for development by Cohesive Manufacturing, a startup formed by the PI, Placid Ferreira and the chief architect of the platform, Jorge Correa

Deliverable 2-5 *– Software Implementation of OSCM*

- Technology Integration: MxD has a working facility consisting of two 3-D printers hooked up to the OSCM Cloud at Illinois to demonstrate the functioning of the software platform.
- Commercialization: The software implementation provides an example for Cohesive Manufacturing, a startup formed by the PI, Placid Ferreira and the chief architect of the platform, Jorge Correa, for implementing cloud services around a manufacturing resource.

Deliverable 6 *– Cyber-physical Manufacturing Network (CyMaN) @ https://oscm-il.mechse.illinois.edu*

- Technology Integration: MxD has a working facility consisting of two 3-D printers hooked up to the OSCM Cloud at Illinois to demonstrate CyMaN's capabilities and functioning. University of Illinois will continue to host this platform as several laboratories are currently using it.
- Education: The platform is used for advanced undergraduate classes in manufacturing (specifically, ME 451: CAD/CAM and ME 452: Numerical Control) laboratory exercises to introduce students to digital manufacturing and cloud platforms for manufacturing. Additionally, the platform is used in reseach projects and hosts experimental facilities at least 4 labs at Illinois.

Deliverable 7 *– Tutorials and Labs*

- Education: These documents are used by Illinois for its advanced undergraduate classes in manufacturing (specifically, ME 451: CAD/CAM and ME 452: Numerical Control) to introduce students to digital manufacturing and cloud platforms for manufacturing.

Deliverable 8 *– Business Canvas*

- Commercialization: This business canvas is used by Cohesive Manufacturing, a startup formed by the PI, Placid Ferreira and the chief architect of the platform, Jorge Correa, as a strategic tool for implementing cloud services around a manufacturing resource.

RECOMMENDED SEQUENCE OF USE

For a user wanting to get familiar with the scope and implementation of OSCM, we recommend the following sequence of use of the project deliverables.

1. Access and Read the document titled *A Reference model for Cyberphysical Manufacturing Networks* in **Deliverable 1** to understand the motivation and framework of OSCM.
2. Access the live Cyberphysical Manufacturing Network (**Deliverable 6**) in the cloud (https://oscm-il.mechse.illinois.edu) and use the tutorials (**Deliverable 7**) to get familiar with OSCM's services. You can use this to create your own private sub-network of manufacturing resources and users to prototype a desired application.
3. After you have got yourself familiar the OSCM services and implementation, you can deploy your own cyberphysical manufacturing network by obtaining the software for MxD (**Deliverable 2-5**) and following the installation instructions.
4. To develop new applications or to extend OSCM services, you will require the API specifications given in the document OSCM_API.html given in **Deliverable 1**.

NEXT STEPS & CHALLENGES

The following are our next steps and challenges.

a) We hope that MxD will consider using OSCM as the framework for integrating the results of different projects. This will provide a cohesive and meaningful way of distributing project results just like OSCM itself is distributed live though the platform running at https://oscm-il.mechse.illinois.edu. The platform can also be used for integrating results from different projects.
b) The University of Illinois continues to run a live platform (url given above) so that any users can access and use its services
c) Other research projects at the University of Illinois and Northwestern are beginning to use the platform as infrastructure for new research proposals. One NSF project on nanoManufacturing has already integrated the platform into its data infrastructure.
d) The PIs on this project will continue to publish research articles on the platform and capabilities to more researchers familiar with OSCM.
e) Cohesive Manufacturing Inc., a startup by PI Placid Ferreira and Jorge Correa (lead architect of OSCM) hope to commercialize OSCM-like capabilities for supplier network management and customer management.

The major challenges faced by a platform such as OSCM is the fact that most manufacturing decision-makers are unfamiliar with cloud computing infrastructure and services. As a result, they have difficulties understanding the strategic importance of a platform like OSCM that enables cohesive development of Industry 4.0/Digital Manufacturing capabilities. Our strategy here is to educate and inform manufacturing decision-makers, while developing a suite of applications that exemplify the strategic importance of adopting OSCM as the platform for future development.

VIII. WORKFORCE DEVELOPMENT

Deliverables 6 and 7 encapsulate the workforce development efforts of this project. Based on this project, two classes, ME451 CAD/CAM and ME 452 Numerical Control have exposed at least 50 undergraduate students per year to digital/cloud manufacturing and Industry 4.0 concepts by using the platform and tutorials as the basis of laboratory assignments and class projects.

IX. CONCLUSIONS & RECOMMENDATIONS

OSCM was conceived as first 'operating system' for cyber-physical manufacturing to create an extendible, non-proprietary platform for developing Digital Manufacturing or Industry 4.0 capabilities. OSCM's services are configured around two primary interacting objects: manufacturing resources and manufacturing jobs/transactions and support multiple operational scenarios associated with job and resource management. It is the first platform that integrates manufacturing operations with IoT services, thus providing an operational context to any data extracted from a resource. It is also the first platform to provide repository services to facilitate the development and distribution of cloud-based apps for manufacturing.

In this project, OSCM was successfully implemented, deployed and tested and currently runs (at the following URL: https://oscm-il.mechse.illinois.edu/) supporting a cyberphysical manufacturing network (CyMaN) **integrates around 40 manufacturing resources ranging from 3-D printers to CNC machines, ultrasonic welders and CVD reactors in around 10 laboratories located at 5 different sites**. Two user facilities continue to use its services to make their facilities available to users. Several labs are currently developing applications specific to their needs over OSCM. For example, researchers at Illinois are developing a sub-net of CVD reactors inside OSCM for large-scale community-based experimentation to support and optimize the growth of 2-D materials. OSCM has been proposed to NSF by a multi-university team as the platform over which to support AI

in Manufacturing Research. Finally, two of the researchers who worked on this project have started up a company, Cohesive Manufacturing Inc., hoping to commercialize cloud-based manufacturing services shown possible OSCM.

Our recommendations to MxD and member companies for additional development include the following:

a. MxD member companies can have individuals register and log into the live OSCM platform (at https://oscm-il.mechse.illinois.edu/). Our team would be happy to help you navigate through the platform and answer your questions.
b. Prototype one of your Industry 4.0/Digital Manufacturing applications as an OSCM application. Our development team would be happy walk you through the process. The most difficult infrastructure for Industry 4.0/Digital Manufacturing has already been developed in the platform and these prototype projects can be completed in a few weeks and cost very little money.
c. MxD may consider requesting software development projects to use the OSCM platform. This would allow live versions of these application be made available to your membership for testing and evaluation.
d. MxD can consider making unused machine capacity on its shop floor available to its membership and partners using OSCM.
e. From a modeling point of view, the reference model for OSCM can be extended
    a. Compound Transaction. The current transaction engine only considers unit (i.e., indivisible) transactions. The model needs to be extended to compound transactions that might span multiple facilities.
    b. Extension of Resources. The resource model in OSCM only considers processing resources. Extending the resource model to logistical resources (e.g., transportation, warehousing/storage) and verification/certification resources would greatly extend the applicability of the platform.
f. As a cloud manufacturing platform, the following technical improvements would be important next steps:
    a. The data-streams in OSCM are implemented using TCP sockets rather than web sockets. This was done because of speed and security considerations. Now, if speed is not a major consideration, one should consider using web sockets for scalability and containerization.
    b. Containerization of OSCM using Docker is important, especially for large-scale applications.
    c. Implementation of pub/sub services will make OSCM more flexible with respect to its data services.
    d. Any commercial implementation would have to add financial processing services to the platform.
    e. 3-legged authentication will make the platform friendlier for users, while 2-factor authentication will make it more secure.
    f. For full-scale operation, the hardware-software platform needs to be optimized. In our live deployment, OSCM is sharing a server with a couple of its assets. This makes the service quite slow. If the live version of the platform is to be maintained, we suggest hosting it on a scalable infrastructure platform like AWS, Azure or Google.

X. LESSONS LEARNED

We believe that this project successfully implemented an important infrastructure platform for digital manufacturing or Industry 4.0. While the teams worked very well together and were able to achieve all the promised deliverables. Among the problems encountered were the following.

a) Our industrial partner could not help us with an on-site implementation of the platform because of cyber security concerns that could not have been foreseen at the beginning of the project. In hindsight, we should have started with a larger group of industry partners and continued to recruit partners as the project progressed

b) Software infrastructure development is complex and one should be more cautious and conservatives with schedules.
c) We should have placed more emphasis on developing some applications to highlight the capabilities of the platform
d) Open-source software is a rich and important enabler when it comes to web/cloud software development. While majority of these libraries and frameworks are robust and stable, the one or two take problematic libraries can account for about 95% of your effort. The lesson one must learn here is to only use stable and well-tested open-source libraries and do not be afraid to develop your own source when you detect problematic libraries.

XI. ACCESSING THE TECHNOLOGY

This project has been implemented so that no background intellectual property is needed. The technology can be accessed at multiple levels of sophistication with increasing demands on infrastructure and personnel.

a) For the purposes of getting familiar with OSCM and prototyping applications, the OSCM cloud at the University of Illinois (https://oscm-il.mechse.illinois.edu) can be freely accessed. We plan on keeping this running and maintained at least until the end of 2021. All that is needed for this type of access is a computer with Google's Chrome web browser.
b) To run OSCM on-premises, one require a server running Windows or Linux operating system, connected to the internet, and having an SSL certificate. In most cases one would need an IT engineer familiar with deployment of web/cloud software.
c) To extend OSCM services and to write software applications over OSCM's API, in addition to the above, one would require a full-stack developer (familiar with Node.js and Express).

XII. DEFINITIONS

What follows are a set of definitions, terms, and acronyms used in this document. These definitions were gathered from various source including the internet, reference papers, standards organizations, and the authors of these documents.

- **OSCM** : Operating System for Cyber-physical Manufacturing
- **NOAM**: Network Operations, Administration and Monitoring
- **CyMaN**: Cyber-physical Manufacturing Network
- **Resource**: In OSCM, a resource is a manufacturing machine, for example, a CNC machine or a 3-D printer.
- **Facility**: An entity that contains multiple OSCM resources and queues.
- **Queue**: A list of transactions associated to a particular facility or resource, ordered by their arrival times.
- **Transaction**: An object containing all necessary information regarding a manufacturing job. It contains information related to the customer, provider, job instructions, files, dates, ready time, due date, and other information configured by a resource/facility owner.

- **Asset**: Web applications, hosted independently from OSCM cloud, that authenticate with resources and access their services through OSCM API.
- **Customer**: Entity or person that consumes manufacturing resource services.
- **Provider**: Entity or person that manages and/or provides manufacturing resource services.
- **Owner**: Person that registers a resource or facility in OSCM server.
- **Adapter**: Machine dependent software that interacts with the machine/controller API and an agent, here the OSCM client. Its goal is to fetch resource data from a physical resource and format it accordingly, so the rest of the OSCM stack can securely consume it.
- **OSCM Client**: Unique piece of software that connects to the adapters and the OSCM cloud. It contains all communication configurations to pipe a data stream from the adapter to the OSCM cloud.
- **JSON**: JavaScript Object Notation
- **STL**: Standard Triangle Language
- **MTConnect**: The MTConnect standard (ANSI/MTC1.4-2018) offers a semantic vocabulary for manufacturing equipment to provide structured, contextualized data with no proprietary format. (https://www.mtconnect.org).
- **TCP**: Transmission Control Protocol is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. It works with the Internet Protocol (IP). (https://searchnetworking.techtarget.com/definition/TCP)
- **Websockets**: WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. (https://en.wikipedia.org/wiki/WebSocket)
- **IP**: Internet Protocol
- **GUI**: Graphical User Interface
- **HTTP**: HyperText Transfer Protocol
- **HTTPS**: HyperText Transfer Protocol Secure
- **WebGL**: WebGL is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins. (https://en.wikipedia.org/wiki/WebGL)
- **SOP:** Standard Operating Procedure
- **JWT-token**: Open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. (https://jwt.io/)
- **Docker**: Docker is a set of platform-as-a-service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. (https://en.wikipedia.org/wiki/Docker_(software))

- **Git**: Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. (https://git-scm.com/)
- **GitLab**: GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and CI/CD pipeline features, using an open-source license, developed by GitLab Inc. (https://en.wikipedia.org/wiki/GitLab). (https://gitlab.com/)
- **npm**: npm is a package manager for the JavaScript programming language. (https://www.npmjs.com/)
- **pub/sub**: pub/sub messaging, is a form of asynchronous service-to-service communication used in serverless and microservices architectures. (https://aws.amazon.com/pub-sub-messaging/)
- **Three-legged OAuth**: Three-legged OAuth processing involves four parties: resource owner, OAuth client, authorization server, and resource server. In other words, three-legged OAuth is a traditional pattern with resource owner interaction. In this case, a resource owner wants to give a client access to a server without sharing credentials. (https://www.ibm.com/)
- **NIST**: National Institute of Standards and Technology.
- **REST API**: Representational State Transfer defines a set of rules that developers follow when they create their API. (https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/)
- **API**: Application Programming Interface.
- **MEAN Stack**: MEAN an open-source JavaScript software stack for building web applications. The MEAN stack is MongoDB, Express.js, AngularJS, and Node.js.
    - https://en.wikipedia.org/wiki/MEAN_(software_bundle))
    - https://www.mongodb.com/
    - https://expressjs.com/
    - https://angular.io/
    - https://nodejs.org/
- **NC**: Numerical Control.
- **CNC**: Computer Numerical Control.
- **CAD**: Computer-Aided Design.
- **CAM**: Computer-Aided Manufacturing.
- **OEM**: Original Equipment Manufacturer.
- **COTS**: Commercial Off-The-Shell.
- **MaaS:** Machine as a Service
- **MES**: Manufacturing Execution Systems.

- **IIoT**: Industrial Internet of Things.
- **CVD**: Chemical Vapor Deposition.
- **AWS**: Amazon Web Services offers reliable, scalable, and inexpensive cloud computing services. (https://aws.amazon.com/)
- **Azure**: Azure is a cloud computing service for building, testing, deploying, and managing applications and services. (https://azure.microsoft.com/)
- **Octoprint**: Open source server with a web interface for your 3D printer. (https://octoprint.org/)
- **SSL**: Secure Socket Layer.

XIII. APPENDICES

None