A Simulation-Based Approach to Assess Condition Monitoring-Enabled Maintenance in Manufacturing

Mehdi Dadfarnia*
Communications Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
mehdi.dadfarnia@nist.gov
*Corresponding author

Michael E. Sharp
Communications Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
michael.sharp@nist.gov

Serghei Drozdov
Communications Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
serghei.drozdov@nist.gov

Jeffrey W. Herrmann
Department of Mechanical Engineering
Catholic University of America
Washington, DC, USA
herrmannj@cua.edu

2023 7th International Conference on System Reliability and Safety (ICSRS) | 979-8-3503-0605-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSRS59833.2023.10381326

***Abstract*—Industrial Condition Monitoring Systems (CMSs) collect and evaluate system and equipment operations to support control and decision-making. Despite their usefulness, understanding the full implications of their impact presents unique and unexpected challenges, especially on new manufacturing systems. System configuration and operational procedures have significant impact on CMS capability and performance. Thus, evaluating a CMS in context of the application is essential both prior to and during deployment. We developed a discrete-event simulator to analyze scenarios with arbitrary manufacturing system configurations and typical operation policies to address this need. This paper describes the results of a study that uses our simulation model to assess the benefits and risks of a CMS selection. We evaluated CMSs with various manufacturing system configurations and maintenance policies to observe how the CMS affected manufacturing system performance. We also discuss the selection of metrics, focusing on the alignment between CMS performance metrics and manufacturing performance indicators.**

***Index Terms*—condition monitoring, manufacturing systems, maintenance, performance metrics**

I. Introduction

Modern manufacturers rely on both digital and physical devices collectively known as Condition Monitoring Systems (CMSs) to inform operations, planning, maintenance, and decision-support across a facility. However, many manufacturing firms are reluctant to incorporate these technologies into their maintenance practices despite their potential usefulness [1]. This reluctance follows from the significant resource needs, as well as difficulties assessing their return on investment [2]. In cases where companies have invested in CMS technology, some are faced with buyer's remorse when the system does not perform as expected, or fails to show clear benefits in relevant time scales. Although many factors can contribute to this, many enterprises simply do not have the resources or knowledge base to address these issues.

Previous work demonstrated assessing a CMS's impact in terms relatable to manufacturing firm decision-makers [3]. For instance, avoiding or reducing risks with increased reliability (such as from using a CMS) has monetary value translatable into financial impact [4], [5]. Additional work provided a framework for identifying the key elements within a manufacturing system's configuration and maintenance policy that determine how a CMS impacts an asset's performance [6].

This work explores risks and benefits of a CMS-driven maintenance policy in a manufacturing setting, both new in development and previously established. We showcase methods for testing, and matching specific types of CMSs to specific systems and goals of a manufacturer. Additionally, this paper focuses on interactivity between a CMS and maintenance policies across different manufacturing configurations.

Broadly, factors preventing the adoption of otherwise useful technologies are either a lack of trust or limitations of economic and personnel resources. To address this, we aim to develop procedures to help users and decision-makers assess CMS technologies' impacts in both economic and engineering capacities. Engineering and system impact assessments support proper levels of trust and confidence, whether to justify the continued use of existing systems or investing in new monitoring technologies.

Our study focuses on impacts of a CMS with respect to overall system performance instead of the more prevalent low-level metrics like accuracy or missed/false alarm rates. System-level manufacturing performance indicators like the part production and quality, or machine availability, provide more directly relevant and actionable information to decision makers. Manufacturing firms use interrelated performance indicators like these to gain understanding of their manufacturing performance [7], [8]. They are explicitly picked to reflect the goals and values of the enterprise and aim to capture

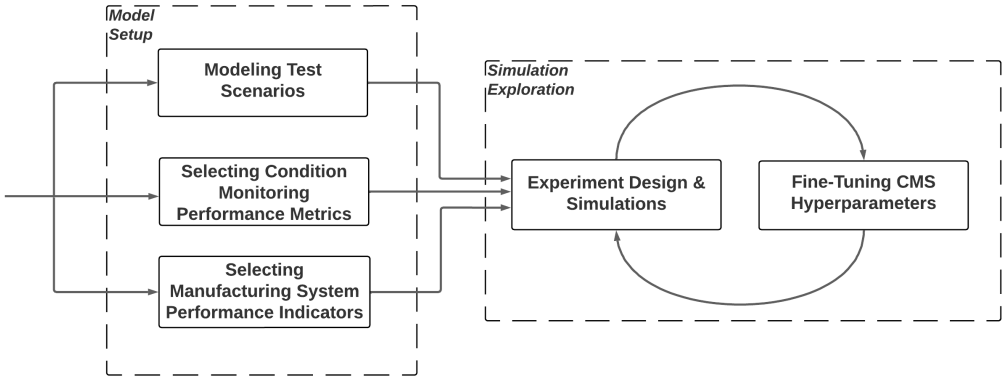

Fig. 1. Flow diagram of steps used in this paper's simulation-based CMS performance assessments.

factors impacting or reflecting system performance. Obvious factors affecting system performance include maintenance policy, machinery configuration, production scheduling, inventory capacity, and regulatory constraints. These same factors can affect the capabilities and effectiveness of a CMS deployed across these systems.

This paper uses a simulation-based approach to evaluate CMS impact on manufacturing performance. Specifically, we consider the interactions of maintenance policy and the production flow path between manufacturing equipment (a.k.a. manufacturing system configuration) with a CMS's ability to impact manufacturing performance. This research considers a set of manufacturing configurations and maintenance policies with significant interplay on CMS performance. In many scenarios, we can manage the effects of CMS performance and corresponding system performance by selecting particular types of CMSs and fine-tuning associated parameters.

Section II gives a description of the methods used to setup simulations and assess CMS performance impact. This section also details the selection of manufacturing system performance indicators and CMS performance metrics to showcase the relationship between CMS and manufacturing performance. Section III presents the results of a case study that explores CMS effectiveness and impacts across five different 6-machine multi-stage manufacturing processes and five different maintenance policies. Finally, Section IV concludes with a summary of our presented work and the prospects for future work.

II. Experimental Setup for Simulations

To help better understand a CMS's impact on discrete-part manufacturing performance, we implement a set of steps using discrete-event simulations divided into two phases. In the first phase, *model setup*, we model a set of manufacturing configurations and maintenance policies and select system-level manufacturing performance indicators and algorithm-level condition monitoring performance metrics. In the next phase, *simulation exploration*, we observe which test scenarios see manufacturing performance improvement with a CMS, and experiment with fine-tuning the CMS to observe the extent to which the CMS can improve manufacturing performance.

Figure 1 sketches the main steps that comprise the two phases of our simulation-based experiments:

- **Model Setup**
  - *Modeling Test Scenarios,* to identify the space of manufacturing machinery configurations and maintenance policies under consideration. This step also involves identifying the subset of maintenance policies that makes use of CMSs, the data collection points from the manufacturing configuration that goes into the CMS as inputs, and the CMS algorithms under consideration.
  - *Selecting Manufacturing System Performance Indicators (PIs),* to identify the set of PIs the manufacturing firm cares about most.
  - *Selecting Condition Monitoring Performance Metrics,* to select algorithm-level performance metrics that can provide a comparative assessment between CMS algorithms applied to the same configuration and maintenance policy.
- **Simulation Exploration**
  - *Experiment Design & Simulations,* to describe the experimental factors and effects in simulating the test scenarios, as well as to observe and analyze performance estimations in the simulations themselves.
  - *Fine-Tuning CMS Hyperparameters,* to optimize CMS models and obtain better performance metrics.

A. Modeling Testing Scenarios with Sim-PROCESD

We model test scenarios for a manufacturing system using an open-source, discrete-event manufacturing simulator called Sim-PROCESD (Simulated-Production Resource for Operations & Conditions Evaluations to Support Decision-making) [9]. Researchers and analysts commonly use discrete-event simulation to rapidly evaluate alternatives to complex manufacturing system configurations and operating policies that support decision-making in regard to manufacturing operations planning and, more recently, maintenance operations planning [10]. Decision makers can quickly run and interpret large numbers of simulation trials to obtain estimations of

system responses (such as manufacturing system performance indicators) and their associated measures of variability [11].

Sim-PROCESD specifically is designed to allow modeling of:

- Any configuration of machines and buffers in a multi-stage manufacturing facility that produces discrete parts (Section II-A1).
- Various degradation mechanisms for machine health and corresponding impacts on part quality (Section II-A2).
- Maintenance policies and repair models that manage machine health, including CMS-integrated condition-based maintenance policies (Section II-A3).
- Characterization of the manufacturing processes and maintenance policies through performance indicators and tracking managed resources (the subjects of Sections II-B & II-C).

1) Configuration of Machines, Buffers, & Maintenance Personnel in the Manufacturing Facility: Each test scenario models a configuration (C_j) of networked machines and buffers. Those machines manufacture parts, with each part progressing along directed or allowable flow paths within the multistage manufacturing configuration. Each part's path is a subset of machines and buffers that takes the part from the facility's source to its sink (see Figure 2 for an example of different possible part paths). The part enters the facility at the source, and each machine along its path performs manufacturing processes on the part until it reaches its final product state at the manufacturing facility's sink.

The six key objects in Sim-PROCESD for modeling production behavior are machines, buffers, sinks, sources, parts, and maintainers. The "upstream" and "downstream" conventions refer to the relative arrangement of assets within a specified part production path - an object receives a part from another upstream object, or an object places a part in another downstream object. Descriptions of the six key objects are as follows, using conventions from a predecessor software to Sim-PROCESD (see [12]):

- Machine: Receives parts from upstream objects and performs manufacturing processes on those parts. A machines receives and processes one part at a time. Once processing is done, the part is available for downstream objects. The *cycle time* τ_m is the time required to process a part on machine m; this time can be constant or random.
- Buffer: Stores parts in a production line that a machine is not currently processing. Each buffer b is characterized by *buffer capacity* (β_b) to specify the maximum number of parts that it can store.
- Source: Introduces parts to the manufacturing system at a rate of γ per unit of time, similar to an inventory of unprocessed parts or raw materials that are up for production.
- Sink: Received the parts at the end of the manufacturing line, similar to an inventory of finished products. Can also be thought of as a buffer with an infinite capacity.
- Part: Objects sent downstream from a source, routed through a flow-path of machines and buffers, to be collected at a sink as final products. Multiple types of parts can be simulated through a model, but each part is considered a self contained complete unit.
- Maintainer: Maintainers perform maintenance tasks. For the purposes of this study, only one maintainer (worker) is needed to complete a task on one machine; the number of maintainers equals the maximum number of machines that can be maintained simultaneously. When all maintainers are busy, any machines that need maintenance will wait in a queue until a maintainer becomes available. There are two types of maintenance tasks: (1) repair a faulted machine to full health, and (2) inspect a machine and determine its health state.

In each configuration model, we instantiate the desired objects with appropriate parameters (such as buffer capacity or machine cycle time) and set the upstream part routes for the modeled machines, buffers, and sink. Using these key objects, we can develop many different manufacturing system configurations. The total set of configurations a manufacturing firm wishes to consider during testing, we denote as $\{C_1, ..., C_q\}$.

2) Machine Degradation & Part Quality Models: We model each machine with a health degradation scheme that is tied to its use. The machines represent a set of tools and equipment to implement a manufacturing process such as joining, cutting, or milling [13]. The degradation model of each machine reflects deteriorating tool wear and other faulty mechanical failures that deviate from the machine's planned manufacturing process tolerances. The health of the machine is an indicator for the quality of its manufacturing process, which directly influences finished product quality [14]. In this work, we lump all machine faults and deterioration paths into a single indicator of effective health, and relate that health to both the performance of the machine and the corresponding quality of the products it produces. Degradation that does not show symptomatic influence on the production rate or product quality can be implemented, but is not currently showcased in this work. Although this simplification may not be valid in every case, it is used here for illustration purposes, and as a means to provide quick but informative approximations prior to performing more in-depth simulations.

Based on the health-related capabilities of each machine, a similarly simplified and lumped product quality parameter associated with each part is updated as they pass through the machines, signifying work has been done. The quality of a finished product is the degree to which a set of characteristics inherent to the product fulfills product design requirements [15]. This quality indicator includes any and all tolerances which the finished product's characteristics must meet based on the design and application requirements. The quality of the finished product is assumed in these simulations to be how well the set of machines that processed the product were able to meet the design requirements. As parts move downstream a manufacturing configuration C_j through a part flow-path of machines, each successive machining process transforms input parts into outputs that are closer to the finished product

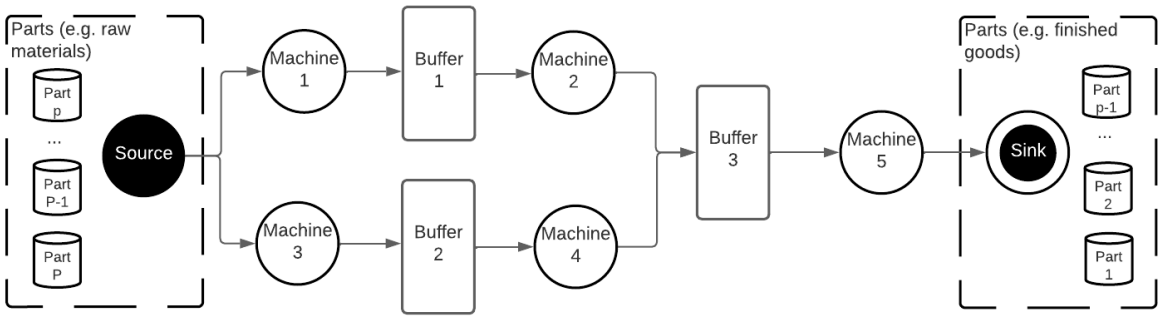

Fig. 2. Example configuration of machines and buffers with 2 possible part production paths.

state, ideally contributing to suitable levels of product quality. Each machine's manufacturing process in the part's flow-path contributes to the finished product's end phase quality, from raw material or simple part to finished product. The progressive degradation of a machine represents the erosion of that machine's manufacturing quality capabilities, and ultimately limits any positive contributions to part quality. This results in output parts with a decreased finished product quality indicator.

In this study, we modeled machine health with an indicator from 0 to 1 corresponding to a nonoperational machine that requires maintenance (Health = 0) to a machine that implements perfect manufacturing processing quality(Health = 1). Likewise we modeled the part quality with an indicator from representing the deviation from nominal tolerances where 0 is a unprocessed part or one that completely fails all quality tests, and 1 indicating a finished product with perfect scores on all quality requirements.The relative scales and permissible thresholds for these metric can be set by user preferences.

For the scenarios presented in this paper, parts start at the simulation source with a quality indicator of 0, and each machine along a part's flow-path is meant to contribute equally to produce a finished product with a quality indicator of 1. For example, in a production path of 4 machines, each machine contributes up to 0.25 to the part's quality indicator, bringing it closer to the nominal tolerances for quality assurance. As a machine's health degrades, its contribution decreases proportionally. In the example, if a machine is at 0.5 health, it would contribute about 0.125 to the part's quality indicator. We also model some degree of variation in machine health and its contribution to part quality. This represents deviations due to process "noise" that may come from environmental factors, thermal effects, operator error, etc. [16].

We characterize each test scenario to have a manufacturing configuration C_j with M number of machines and B number of buffers. Depending on the configuration C_j, the maintenance policy Π_k, buffer capacities β_b for each buffer b, machine cycle times τ_m for each machine m, and the duration of the simulation T_{sim}, each simulation of $S_{j,k}$ produces a total of P parts. The number P varies between simulation runs of the same test scenario $S_{j,k}$ due to modeled stochastic parameters that represent the variations and randomness that appear in manufacturing systems. We note that simulation termination T_{sim} scraps all parts in the production process.

The value contribution to the quality indicator of a part p by a machine m at time t is captured by:

$$q_m(t) = (v_m + \epsilon_m)(H_m(t) + \delta_m), \text{ where } t \geq 0, \quad (1)$$

where v_m is the expected quality value added by machine m onto a part; ϵ_m is a small amount of randomly generated noise in the added value v_m; $H_m(t)$ is the health indicator of machine m at time t; finally, δ_m is the minuscule randomly generated variation in the machine's health.

Eq (1) shows the quality contribution that machine m provides to a part processed at time t, thus the resulting total quality indicator will depend on the quality contributions for machine m and all prior machines that processed that product in its part flow-path $w \in W$. A part path w is the set of all machines that processes a part from source to sink. A configuration C_j will have a maximum of W part paths, derived from its arrangement of machines and buffers. For simplicity, we ensure that the length L_j (or number of machines) of each part path w within each configuration C_j is the same, i.e., all parts have the same number of machining stages.

The final product quality indicator Q_p for any part p that went through the part flow-path w is defined as:

$$Q_p = \sum_{m \in w} (v_m + \epsilon_m)(H_m(t_{p,m}) + \delta_m), \text{ where } t_{p,m} \geq 0, \quad (2)$$

The timestamp when machine m processed part p is indicated by $t_{p,m}$ and is progressive off a global clock of the simulation. In this study $v_m = 1/L_j$ for all $m \in M$ in C_j. This follows from our choice to model all part paths w in C_j as having the same path length L_j, and nominally require each machine to contribute equally to achieve a total part quality indicator Q_p value of 1. The machine health indicator is bound by $H_m(t_{p,m}) \in (0, 1)$ and represents the ratio by which the machine is capable of contributing quality value to the part

it processes. This means that, when taking into account the variations that come from ϵ_m and δ_m, the part quality indicator is bound by
$Q_p \in \big((1 + \min(\epsilon_m))(\min_{H_m \neq 0}(H_m) + \min(\delta_m)), (1 + \max(\epsilon_m))(1 + \max(\delta_m))\big)$.

In each simulation the machines are all reset to start with a perfect health condition ($H_m(t = 0) = 1, \forall m$) for each test scenario $S_{j,k}$. Each machine's degradation model instantiates at the beginning of each simulation run, along with its production processes. The machines degrade independently from each other, though we set them to reference the same underlying degradation mechanisms.

At each degradation event, we model a 0.125 reduction in a machine's health indicator $H_m(t)$. This forms a finite and discrete state space for a machine's health indicator, $H_m = \{1, 0.875, 0.75, 0.625, 0.5, 0.375, 0.25, 0.125, 0\}$, whose state transition function is deterministic. If the health indicator reaches 0, a maintenance work order is created. Other mechanisms like inspection or alerts from a CMS can trigger a maintenance work order before the machine's health indicator reaches 0. Section II-A3 will discuss various maintenance policies in more detail. Regardless of the triggering event, completion of the maintenance work order results in perfect maintenance, reverting the machine's health indicator back to 1. This restoration to perfect health is also a simplification that may not always be valid, but is used here for convenience and best case illustrative purposes.

The distribution of the time between degradation events is a two-parameter Weibull distribution with probability density function:

$$f(t) = \frac{\lambda}{\alpha^\lambda} t^{\lambda-1} e^{-(t/\alpha)^\lambda}, \text{ where } (t, \alpha, \lambda) > 0, \qquad (3)$$

where λ is the shape parameter and α is the scale parameter.

Each degradation event follows a stochastic clock structure as shown in Equation (3). Since the events themselves are finite and discrete, the machine health indicator states are also finite and discrete, and the state transitions are deterministic. The degradation model for $H_m(t)$ is a stochastic-timed state automata that generates a generalized semi-Markov process, a well-studied behavior of many stochastic, discrete-event systems [17].

3) Maintenance Policies : The goal of maintenance is to keep equipment healthy, safe, available to fulfill all related system requirements. In production manufacturing, often the additional goals of maintaining flexibility, consistent operations, high throughput, and quality production output are also added. Throughout this work, we describe, model, and implement several types of maintenance policies to form a set of policies $\{\Pi_1, ..., \Pi_r\}$. Each test scenario evaluated $S_{j,k}$ is characterized by the pair $\{C_j, \Pi_k\}$ of manufacturing configuration C_j and a specific maintenance policy Π_k. This means given q manufacturing configurations and r maintenance policies, there exists $q \cdot r$ possible number of test scenarios from $\{S_{1,1}, \ldots, S_{q,r}\}$.

In this study, we focus on three major classes of maintenance strategies: corrective run-to-failure policies, time-based inspection policies, and CMS-enabled condition-based policies. Here, we consider the time-based inspection and CMS-enabled policies forms of proactive preventive maintenance, as they are nominally designed to trigger before a machine fails. A reactive, corrective maintenance policy, also known as run-to-failure, represents the worst-case scenario where there is no chance for action or discovery prior to machine failure. Time-based inspections represent commonly performed practical maintenance practices that operate and trigger with no direct need of continual system monitoring. The CMS based policies represent the potential results of enacting autonomous or semi-autonomous active evaluation of the system during operations. The three additional CMS-enabled policies we selected represent various methods between a best-case scenario where perfect information about each machine's health is known and a severely limited assessment of the system health states.

We model a maintainer object within the simulator to execute these policies. The maintainer has two possible actions: performing maintenance work that moves machines back to full health, or conducting inspections to gain knowledge of current machine health. Each maintenance policy describes the mechanisms and events involved in which the maintainer object performs these tasks.

**** Run-To-Failure Policy***

In our corrective, run-to-fail policy model, the machines produce parts until their degradation state indicates failure ($H_m(t) = 0$). At failure, the machine shuts down, no longer receives incoming parts, and scraps (i.e., discards) any parts currently being operated on. This failure then triggers the machine to submit a 'repair' maintenance work order that will be recognized and queued with the maintainer. Maintenance only occurs when maintenance resources are available. Maintenance resource availability for repairing a machine depends on the maintenance capacity parameter MT_{cap} of maintenance personnel that can perform the repair task. This parameter models the number of repair maintenance work orders that can be addressed simultaneously. Once maintenance is available for the failed machine, a maintainer responds and performs the task that restores the machine to full health ($H_m(t) = 1$).

**** Periodic Inspection Policy***

In our preventive, time-based inspection policy models, an inspector (a special sub-class of maintainer) periodically checks the health indicators of the machines in the configuration. Each inspection pauses the machine for a duration of TTI_m to call a maintenance work order if the health of the machine is under a threshold TH_{insp_m}. The first inspection of each machine is requested at a time $\tau_{first\_insp\_m}$ after the simulation start. This same interval is nominally the time until the first inspection of a machine after it has undergone a repair action. All subsequent machine inspections are requested every τ_{insp_m} units of time until the machine again goes under repair or the simulation ends. We model $\tau_{first\_insp\_m} < \tau_{insp\_m}$ to represent earlier inspections that check for early machine failures that may result from high failure rates early in a typical

machine reliability bathtub curve.

The occurrence of a machine's inspection depends on the scheduled request time and the availability of maintenance personnel that can perform inspections (which we will refer to as inspectors). Inspector availability is dependant on the user-specified inspection capacity parameter IN_{cap}, i.e., the number of inspections that can be performed simultaneously, and the priority log of queued inspection tasks from other machines. If all the inspectors are busy, any additional scheduled inspections will be queued for the next inspection availability. Failures and requests for machine maintenance supersede inspection requests: if a machine fails with inspections queued, those inspections will be canceled and restarted as normal after repairs occur.

**** Condition Monitoring System(CMS) Enabled Policies***

We have modeled three different CMSs for the condition-based maintenance policies in our case study (Section III). The simulation-based approach of this paper can apply to other CMS types. We do not model any time-based inspections in conjunction with our CMS-enabled policy models. The three CMS-enabled policies are:

- Complete Perfect Sensing - the CMS exactly knows and reports the entire state of the system
  - 1) Perfect CMS
- Limited Scope Sensing with AI Diagnostics - The CMS uses AI algorithms and a limited set of inspection capabilities to infer where problems with machines occur
  - 2) Part Quality Contribution Indicator-based (PQCI-based)
  - 3) Prediction Error Minimization-based (PEM-based)

Complete Perfect Sensing

The 'perfect CMS,' provides complete and accurate information about each machine's health to the maintainer. The CMS records the machine health indicator data from each machine's sensors every τ_{sense} units of time. If, at any sensing event, the CMS detects that any of the machine health indicators has fallen at or below a machine health threshold TH_{cms}, it creates a maintenance work order for the corresponding machine.

Limited Scope Sensing with AI Diagnostics

The other two CMS-enabled maintenance policies deploy a sensing apparatus only at the end of the production line (at the sink object) and use algorithms that can predict machine health. We refer to these two as the limited scope 'AI-Inference' CMSs since their alerts about machine degradation or failure are not based on complete information. These two AI-Inference CMSs represent monitoring systems with low requirements for setup.Instead of directly monitoring the machines, they instead utilize final part quality information from the end of the production line. This lowered investment is appealing to many enterprises and allows a much more cost-effective CMS-enabled maintenance policy than the perfect CMS case. However, the trade-off is that they produce higher rates of false or missed alerts for machine health.

The AI-Inference CMS inputs come from each finished product p that arrives at the sink. Given each finished product p, the CMS records that part's quality indicator Q_p and its part flow-path w (i.e. list of machines that processed it). The CMS uses these inputs from each product in an algorithm that identifies the most likely machine to be producing lower quality parts. Each product completed allows the CMS to update health indicator predictions for all the relevant machines. If a CMS predicts that a particular machine is low on its health indicator or contribution to part quality, the CMS creates a maintenance work order for that machine, same as in the perfect CMS scenario.

The difference between the two AI-Inference CMSs lies in the algorithm used to evaluate the likelihood of each machine requiring maintenance. We will refer to the first AI-Inference CMS algorithm as the Part Quality Contribution Indicator-based (or PQCI-based) CMS, and the second as Prediction Error Minimization-based (or PEM-based) CMS. In the style of temporal-difference reinforcement learning, the PQCI algorithm creates logical evaluations of the relative probability that any machine in the configuration can induce lower part quality contributions on the observed finished product. The PEM algorithm uses open-source, off-the-shelf local search algorithms (from the optimization module of Python's SciPy library) to minimize the prediction error for each machine's contributions towards finished part quality. We discuss the mechanisms behind these algorithms in greater detail in our prior work [18].

4) Repair Models: There are two major types of repair addressed in this experiment, reactive corrective work and proactive preventative work. As mentioned earlier, corrective work is defined as repairs that occur after a machine has reached a fail state, and preventative work occurs before the machine reaches the catastrophic failure state ($H_m(t) = 0$). Nominally the Periodic Inspection and CMS-Enabled maintenance policies would primarily or exclusively trigger preventative work, although corrective actions are a possibility. Conversely, the Run-to-Failure policy ONLY provides corrective work.

A standard assumption that we employ here is that corrective work is more resource intensive than preventive work, both in time/labor and parts/cost. To reflect this, we employ two different repair time models based on the state of the machine when the maintenance request was placed.

Reactive Corrective Repair

If the machine reaches a fail state($H_m(t) = 0$) triggering a request or before a request can be fulfilled, the machine halts production, scraping and in process parts it holds, then waits for the next available maintainer to come repair it.

The repair action takes $TTR_{failed\_m} + \epsilon_{TTR}$ to complete, where TTR_{failed_m} is the mean time to repair and ϵ_{TTR} represents some variance to the completion time. A single maintainer is reserved during this time and can not work on other maintenance work orders until the time to repair duration is over. Once completed the maintainer is released and the machine returns to producing parts at full health.

Proactive Preventative Maintenance

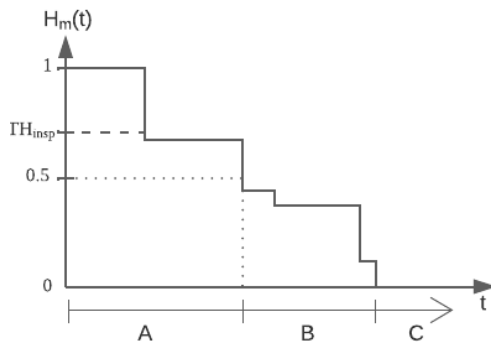

Fig. 3. Repair duration if machine health is in 'Zone A': $TTR_{mild\_det\_m}$; repair duration if in 'Zone B': $TTR_{deteriorated\_m}$; repair duration if in 'Zone C': TTR_{failed_m}.

If a machine does NOT reach a failed state($H_m(t) = 0$) before the maintenance request is fulfilled, then as the maintainer arrives to perform repairs the machine production is paused and repairs can commence. The amount of time needed for this repair action is proportional to the health state of the machine at the time of repair. The maintainer will repair the machine back to full health with either a time-to-repair duration of $TTR_{mild\_det\_m} + \epsilon_{TTR}$ or $TTR_{deteriorated\_m} + \epsilon_{TTR}$, depending of whether $H_m(t) \geq 0.5$ or $0.5 \geq H_m(t) > 0$, respectively (see Figure 3). No parts are scraped during this type of repair action.

Regardless of the type of repair action, after completion the machine is restored to complete and perfect operations ($H_m(t) = 1$). As previously mentioned, this is not always the case, but is a simplification made here to help highlight the interplay of system configurations and maintenance policies, without over complicating the results.

B. Selecting Manufacturing System Performance Indicators

Industry practitioners use a variety of performance indicators (PIs) to track the performance of their manufacturing firm and ensure that manufacturing performance reflects the company's strategy [7]. Balancing the cost and value of PIs is a challenge: "The trick is to measure as little as possible, but to ensure that you are measuring the things that matter" [19].

In this study, we used the following PIs to measure the effects of using different maintenance policies Π_k and manufacturing configurations C_j:

- *Machine Availability (MA):* The proportion of time in a simulation run that a machine is not in a failed state, not undergoing repairs, or (if applicable) not undergoing inspections.
- *Machine Mean Time Between Failure (MTBF):* The expected time between two machines failures in a simulation run.
- *Machine Repair Count (MRC):* Number of completed repair work orders for a machine in a simulation run. Includes corrective repairs as well as preventative repairs triggered by inpection or condition monitoring.
- *Work-In-Process (WIP):* The average number of parts waiting for further processing in buffers during a simulation run.
- *Produced Parts (PP):* The average number of parts produced in a simulation run.
- *Mean Product Quality (MPQ):* The average part quality from the produced parts.

The first three PIs (*MA*, *MTBF*, and *MRC*) describe manufacturing reliability and maintainability. The next two PIs (*WIP* and *PP*) give indicators of productivity and insight on machine bottlenecks. The PI *MPQ* is an indicator for manufacturing process quality.

C. Selecting Condition Monitoring Performance Metrics

Given the results of a simulation run, we can summarize information about predictions for machine maintenance and actual machine health conditions to construct a confusion matrix of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) [20]. We used the following confusion metric-based metrics to evaluate CMS performance:

- *True Positive Rate (TPR):* Also known as sensitivity or recall, *TPR* measures the proportion of TPs with respect to all actual machine health positives in a simulation run.
- *False Positive Rate (FPR):* Also known as fall-out, *FPR* measures the proportion of FPs with respect to all the actual negatives in a simulation run.
- *Positive Predicted Value (PPV):* Also known as precision, *PPV* measures the proportion of TPs with respect to all the CMS alerts for maintenance (predicted positives) in a simulation run.
- *F1 Score (F1):* Measures the weighted harmonic mean between PPV and TPR in a simulation run.

D. Experimental Design & Running Simulation Experiments

We conducted a systematic simulation study to compare different test scenarios on their manufacturing performance indicators and observe any impacts from CMS use. We experimented with different manufacturing configurations and maintenance policies for a manufacturing system that has M machines. The manufacturing configuration and the maintenance policy form our two experimental factors. We select q manufacturing configurations and the r maintenance policies to form the experimental levels for each factor. We experiment over all $q \cdot r$ combinations of the factors, each referred to as a test scenario $S_{j,k}$. We measure the effects of these factors on the experimental responses, the manufacturing system performance indicators (Section II-B) and condition monitoring performance metrics (Section II-C). All simulations for all test scenarios ran for the same duration of simulated time, T_{sim}, to reflect operation of the actual manufacturing system.

For each test scenario, the simulation results provide the manufacturing performance indicators and any CMS performance metrics. We then use the performances measured from each test scenario's simulation set to characterize each of the test scenario's manufacturing performance indicators and

CMS performance metrics with a sample average and sample standard deviation.

E. Fine-Tuning Condition Monitoring Hyperparameters

After conducting N simulation replications and obtaining performance estimates with each test scenario $S_{j,k}$, we focused on conducting further experiments with the subset of test scenarios that make use of a CMS-enabled, condition-based maintenance policy.

We experiment with hyperparameters for each of the three CMSs discussed in Section II-A3 (the Perfect CMS, the PQCI-based CMS, and the PEM-based CMS). The goal was to fine-tune the hyperparameters as a way to optimize the CMS models for obtaining better manufacturing performance indicator estimates and observing the corresponding CMS performance metrics. We focused on the following hyperparameters for each of our CMS models:

- Perfect CMS
  - TH_{CMS}: CMS alert detection threshold for direct machine health sensor measurements.
- PQCI-based CMS
  - h_{PQCI}: Number of last final product quality values processed by each machine (in its part path w) to consider for CMS alert prediction.
  - TH_{PQCI}: CMS alert detection threshold for predictions of each machine's probability to induce lower final product quality.
- PEM-based CMS
  - h_{PEM}: Number of latest final product quality values to consider for CMS alert prediction.
  - TH_{PEM}: CMS alert detection threshold for predictions of each machine's contribution to part quality.

For each hyperparameter, we selected Z values within a practical range. For each test scenario that made use of any of the three CMSs, we performed a parameter sweep for each of its hyperparameters. In each parameter sweep, we replicated N simulations for each of the Z hyperparameter values.

III. Case Study

A. Model Parameters

This section describes the fictional manufacturing configurations that were used to evaluate CMS performance. These are labeled $\{C_1, C_2, C_3, C_4, C_5\}$ (see Figure 4). Each configuration has six machines that represent 3-axis milling machines to produce a single type of part. Each machine in configuration C_1 performs a specialized, short operation. At the other extreme, each machine in configuration C_2 performs a more time-intensive series of operations on each part.

To allow for direct comparison between configuration performance, we ensure that the sum of cycle times τ_m of all machines in each part flow-path w, regardless of the configuration, is 6 hours. We also ensure that the cycle times are the same for every machine within a configuration. All of the buffers have a capacity β_b=10 units. The source introduces parts at a rate of γ=1 part per hour. We model the degradation

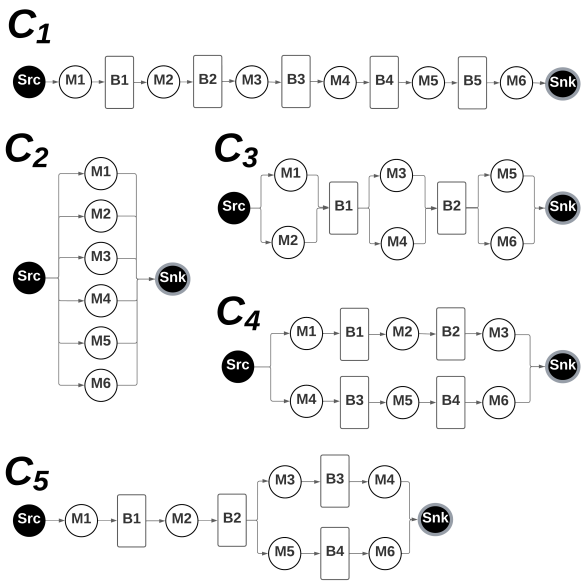

Fig. 4. Five manufacturing configurations $\{C_1, ..., C_5\}$, each consisting of 6 machines (M), 1 source (Src), 1 sink (Snk), and buffers (B).

events of each machine's health indicator $H_m(t)$ with Equation (3), where $(\lambda, \alpha) = (1.5, 12)$ for all machines except Machine 3. Instead, we modeled Machine 3's degradation with the parameters $(\lambda, \alpha) = (0.9, 3)$, which causes Machine 3 to degrade much more rapidly than other machines. This model choice allows us to test the sensitivity of our CMS-enabled maintenance policies to machines with varying reliability.

We evaluated CMS performance with the five maintenance policies described in Section II-A3: Π_1 (Run-to-Failure), Π_2 (Periodic Inspection), Π_3 (PQCI-based Condition Monitoring), Π_4 (PEM-based Condition Monitoring), and Π_5 (Perfect CMS). Three maintenance personnel ($MT_{cap} = 3$) are available for maintenance work, regardless of configuration or policy. This allows us to compare the performance of different test scenarios without worrying about performance bottlenecks caused by a lack of personnel. We present the remaining maintenance & repair model parameters, excluding CMS hyperparameters from Section II-E, in Table II.

B. Results & General Discussion

The choice of five different configurations and five different maintenance policy options illustrate the scale of analysis and decision-making for complex manufacturing systems. We run simulations on each of the 25 possible test scenarios. The total duration for simulating each test scenario is set to T_{sim}=10,080 hours. Initially, we run N=50 simulation replications for each of the 25 test scenarios. If the test scenario involved a CMS-enabled condition monitoring policy, we conducted a manual parameter sweep over the relevant CMS hyperparameters discussed in Section II-E ($Z = 9$ values for the Perfect CMS and $Z = 17$ for PQCI-based and PEM-based condition monitoring). Table I shows a subset of manufacturing performance indicators and condition monitoring

TABLE I
PERFORMANCE MEASURE MEANS
& STANDARD DEVIATIONS FOR TEST SCENARIOS

Test Scenario	*MA* (%)	*MRC* (Num. of Repairs)	*PP* (Num. of Parts)	*MPQ* (Index Val.)	CMS *TPR* (Ratio)	CMS *PPV* (Ratio)
$\{C_1, \Pi_1\}$	76.70	115.12	4730.16	0.60		
	±9.484	±47.427	±53.638	±0.123		
$\{C_2, \Pi_1\}$	76.66	115.22	7218.26	0.54		
	±9.496	±47.499	±22.831	±0.277		
$\{C_3, \Pi_1\}$	76.66	115.32	6351.08	0.57		
	±9.526	±47.638	±49.626	±0.165		
$\{C_4, \Pi_1\}$	76.66	115.24	6422.24	0.57		
	±0.095	±47.702	±41.333	±0.167		
$\{C_5, \Pi_1\}$	76.69	115.13	4812.82	0.58		
	±9.431	±47.250	±29.976	±0.149		
$\{C_1, \Pi_2\}$	74.26	216.29	4840.52	0.68		
	±8.891	±40.684	±55.565	±0.108		
$\{C_2, \Pi_2\}$	74.19	216.18	6888.28	0.64		
	±8.932	±41.218	±15.357	±0.253		
$\{C_3, \Pi_2\}$	74.26	216.35	6304.08	0.65		
	±8.838	±40.784	± 38.959	±0.148		
$\{C_4, \Pi_2\}$	74.32	214.94	6270.60	0.65		
	±8.880	±41.585	±38.913	±0.149		
$\{C_5, \Pi_2\}$	74.24	216.08	4734.92	0.67		
	±8.872	±40.677	±21.991	±0.131		
$\{C_1, \Pi_3\}$	76.71	115.06	4702.84	0.60	0.0	0.0
	±9.507	±47.498	±49.852	±0.123	±0.0	±0.0
$\{C_2, \Pi_3\}$	59.66	625.79	5403.82	0.72	0.113	0.339
	±7.880	±8.546	±15.630	±0.248	± 0.0047	±0.0114
$\{C_3, \Pi_3\}$	60.71	585.70	5051.36	0.74	0.093	0.377
	±7.793	±23.173	± 35.379	±0.143	±0.0051	±0.0168
$\{C_4, \Pi_3\}$	61.17	570.49	5080.58	0.74	0.101	0.375
	±7.477	±16.371	±39.420	±0.140	± 0.0050	±0.0156
$\{C_5, \Pi_3\}$	66.86	399.45	4792.70	0.66	0.012	0.411
	±12.258	±217.082	±38.467	±0.140	±0.0013	±0.0348
$\{C_1, \Pi_4\}$	76.48	121.89	4704.98	0.61	0.002	0.489
	±9.407	±46.209	±65.504	±0.123	±0.0008	±0.0737
$\{C_2, \Pi_4\}$	62.31	448.72	5592.98	0.63	0.041	0.182
	±11.459	±126.198	±26.540	±0.281	±0.0013	±0.0064
$\{C_3, \Pi_4\}$	69.00	306.00	5623.02	0.64	0.028	0.246
	±10.222	± 72.535	± 35.563	± 0.162	±0.0012	±0.0097
$\{C_4, \Pi_4\}$	72.90	215.46	5949.02	0.63	0.016	0.316
	±10.737	± 68.527	±44.396	± 0.167	±0.0012	±0.0195
$\{C_5, \Pi_4\}$	73.34	217.37	4800.18	0.61	0.019	0.303
	±11.113	±111.049	±36.637	±0.146	±0.0013	±0.01693
$\{C_1, \Pi_5\}$	66.09	669.00	3971.20	0.79	1.0	1.0
	±9.729	± 133.812	±50.869	± 0.075	±0.0	±0.0
$\{C_2, \Pi_5\}$	66.09	667.16	6151.54	0.77	1.0	1.0
	±9.788	±135.454	±30.758	±0.182	±0.0	±0.0
$\{C_3, \Pi_5\}$	66.10	668.82	5456.80	0.78	1.0	1.0
	±9.737	±133.034	±55.383	±0.106	±0.0	±0.0
$\{C_4, \Pi_5\}$	66.06	667.82	5368.60	0.78	1.0	1.0
	±9.707	±134.060	±55.919	±0.104	±0.0	±0.0
$\{C_5, \Pi_5\}$	66.15	665.88	4282.40	0.79	1.0	1.0
	±9.768	±134.042	±63.189	±0.092	±0.0	±0.0

*Note: Each CMS-enabled test scenario had their CMS fine-tuned across a sweep of hyperparameters; results presented for hyperparameters that produce the highest mean *MPQ*.

performance metrics that we generated from simulating these test scenarios.

The Table I shows that while manufacturing configurations have a major impact on part production *PP*, the use of CMSs in a maintenance policy also has major impacts on manufacturing performance. We observe the full impact that a CMS can have from the results of test scenarios that include Π_5, the Perfect CMS, and comparing them against the results of test scenarios that operate maintenance with Π_1, Run-to-Failure. A configuration with Π_1 has higher part production *PP*, lower maintenance repair counts *MRC*, and higher machine availability *MA* than the same same respective configuration that operates by policy Π_5. However, configurations with Π_5 indicate a much higher part quality *MPQ*. The higher maintenance work orders that result from using the Perfect CMS direct result in lower productivity and higher part quality. A manufacturer with higher emphasis on part quality over quantity would prefer the complete perfect sensing of Π_5.

Test scenarios that operate on the Π_2 Periodic Inspection maintenance policy also provide a useful reference point.

TABLE II
MAINTENANCE & REPAIR MODEL PARAMETERS

Periodic Inspection Policy Π_2	
IN_{cap}	1
TTI_m	1 hour
TH_{insp_m}	0.5
$\tau_{first\_insp\_m}$	20 hours
τ_{insp_m}	40 hours
Perfect CMS-based Policy Π_5	
τ_{sense}	1 hour
Time-To-Repair Values	
TTR_{failed_m}	20 hours
$TTR_{deteriorated\_m}$	5 hours
$TTR_{mild\_det\_m}$	2.5 hours

Manufacturing performance indicators for configurations with the Π_2 policy result in higher productivity and lower part quality than respective configurations that operate with Π_5, and lower productivity and higher part quality than respective configurations that operate with Π_1. Ideally, a cheaper, non-perfect sensing, CMS-based maintenance policy should result in manufacturing performance that is comparable with Π_2. That is not necessarily the case with policies Π_3 and Π_4.

We observe that policy Π_3 does show improved part quality *MPQ* for configurations $\{C_2, C_3, C_4\}$ where no part flow-paths w share a machine. This is due to the PQCI algorithm used in Π_3, which compares part qualities coming from different paths. In contrast, Π_3 was not able to produce a single maintenance work order in the test scenario $\{C_1, \Pi_3\}$, as indicated by the resulting *TPR* metric, because all the machines in C_1 are in the same, single part flow-path. Another impact that part flow-paths have on the CMSs used in Π_3 and Π_4 can be seen in the variability in maintenance work order counts for machines in configuration C_5. Because machines M1 and M2 are in the all of the flow-paths in C_5, the CMSs alert for maintenance less frequently than for other machines. Overall, insights about which configurations a CMS-enabled policy such as Π_3 can impact are invaluable for creating and improving upon similar CMSs. The goal is to use CMSs that achieve condition monitoring performance metrics (*TPR, PPV*) and resulting manufacturing performance impacts (*MPQ*) closer to that of the Perfect CMS case.

IV. CONCLUSIONS

In this paper, we evaluate the use of CMSs in a discrete-part manufacturing setting with manufacturing and condition monitoring performance measures across various manufacturing configurations and maintenance policies. We describe a simulation-based approach toward CMS evaluation and demonstrate our approach with a case study of twenty-five test scenarios from five configurations and five different maintenance policies (three being CMS-enabled condition-based maintenance policies). The case study results show that using CMSs can positively impact product quality but may hinder part production and machine availability. Future works will expand on the performance indicators and metrics, the types of CMSs and maintenance policies used, more refined CMS hyperparameter tuning, and sensitivity analysis for model parameters such as time-to-repair.

DISCLAIMER

The use of any products described in this paper does not imply recommendation or endorsement by the National Institute of Standards & Technology, nor does it imply that products are necessarily the best available for the purpose.

REFERENCES

[1] J. Bokrantz, A. Skoogh, C. Berlin, T. Wuest, and J. Stahre, "Smart maintenance: a research agenda for industrial maintenance management," *International journal of production economics*, vol. 224, p. 107547, 2020.

[2] E. Zio, "Some challenges and opportunities in reliability engineering," *IEEE Transactions on Reliability*, vol. 65, no. 4, pp. 1769–1782, 2016.

[3] M. Sharp, M. Dadfarnia, T. Sprock, and D. Thomas, "Procedural guide for system-level impact evaluation of industrial artificial intelligence-driven technologies: Application to risk-based investment analysis for condition monitoring systems in manufacturing," *Journal of Manufacturing Science and Engineering*, vol. 144, p. 7, 2022.

[4] C. Chen, G. Reniers, N. Khakzad, and M. Yang, "Operational safety economics: Foundations, current approaches and paths for future research," *Safety science*, vol. 141, 2021.

[5] K. B. Marais and J. H. Saleh, "Beyond its cost, the value of maintenance: an analytical framework for capturing its net present value," *Reliability Engineering & System Safety*, vol. 94, no. 2, pp. 644–657, 2009.

[6] M. Dadfarnia and M. Sharp, "Key elements to contextualize ai-driven condition monitoring systems towards their risk-based evaluation," *In 5th IEEE International Conference on Artificial Intelligence for Industries (AI4I)*, vol. 5, pp. 1–4, 2022.

[7] C. Lundgren, J. Bokrantz, and A. Skoogh, "Performance indicators for measuring the effects of smart maintenance," *International Journal of Productivity and Performance Management*, vol. 70, no. 6, pp. 1291–1316, 2021.

[8] Y. Koren, S. J. Hu, and T. W. Weber, "Impact of manufacturing system configuration on performance," *CIRP annals*, vol. 47, no. 1, pp. 369–372, 1998.

[9] S. Drozdov, M. Sharp, and M. Dadfarnia, "Simulated-production resource for operations and conditions evaluation to support decision-making," 2022. [Online]. Available: https://doi.org/10.18434/mds2-2733

[10] A. Negahban and J. S. Smith, "Simulation for manufacturing system design and operation: Literature review and analysis," *Journal of manufacturing systems*, vol. 33, no. 2, pp. 241–261, 2014.

[11] E. Zio, "The future of risk assessment," *Reliability Engineering & System Safety*, vol. 177, pp. 176–190, 2018.

[12] M. L. Hoffman, *Online Maintenance Prioritization Via Monte Carlo Tree Search and Case-Based Reasoning in Complex Manufacturing Systems*. The Pennsylvania State University, 2021.

[13] R. H. Todd, D. K. Allen, and L. Alting, *Manufacturing processes reference guide*. Industrial Press Inc., 1994.

[14] T. Wuest, C. Irgens, and K.-D. Thoben, "An approach to monitoring quality in manufacturing using supervised machine learning on product state data," *Journal of Intelligent Manufacturing*, vol. 25, pp. 1167–1180, 2014.

[15] "Quality management systems — Fundamentals and vocabulary," International Organization for Standardization, Geneva, CH, Standard, Sep. 2015.

[16] M. J. Kaiser, "Generalized zone separation functionals for convex perfect forms and incomplete data sets," *International Journal of Machine Tools and Manufacture*, vol. 38, no. 4, pp. 375–404, 1998.

[17] F. Grabski, *Semi-Markov processes: applications in system reliability and maintenance*. Elsevier, 2014.

[18] M. Dadfarnia, M. Sharp, and T. Sprock, "Understanding and evaluating naive diagnostics algorithms applicable in multistage manufacturing from a risk management perspective," *In International Manufacturing Science and Engineering Conference*, vol. 84263, p. p. V002T07A042, 2020.

[19] A. Neely and M. Bourne, "Why measurement initiatives fail," *Measuring business excellence*, vol. 4, no. 4, pp. 3–7, 2000.

[20] N. Japkowicz and M. Shah, *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.