A Benchmark Simulator for Log-structured Data In Fault Prediction and Anomaly Detection in Flexible Assembly Systems - DRAFT

Tero Keski-Valkama

***Abstract*—**

***Index Terms*—**

I. Introduction

Flexible assembly system was described by Donath and Graves [1] as a system consisting of a set of products each with a specified volume assembled on a workshop consisting of a fixed number of cells. In practise, a flexible assembly system contains parts and materials stored in an intermediate storage, a conveyor or crane system to move the parts, materials, intermediate assemblies and finished products between the cells, and the cells with work machines and necessary tooling to assemble and process intermediate assemblies and products.

System downtime is a significant cost for flexible assembly systems. System downtime is reduced by preventive maintenance typically scheduled periodically. Recently Internet of Things and opening of the industrial system APIs have created new possibilities for predictive maintenance. Different sites with similar flexible automation systems have widely different environmental and workload conditions which has an impact on the wear and tear of the flexible automation system. Taking different site environments and workloads into account when planning preventive maintenance cycles is non-trivial.

Going from preventive maintenance towards predictive maintenance optimizes and targets the maintenance related costs towards the activities that have best impact on improving the availability and operation of deployed flexible manufacturing systems. Predictive maintenance in flexible assembly systems benefits from automatic indicators for potential future faults.

My own future research includes plans to implement novel anomaly detection methods for flexible manufacturing systems. Validating and quantifying these methods require public datasets to execute these methods against.

Currently there are no standard benchmarks to test different anomaly detection methods and compare their performance against each other. Such standard benchmarks exist in other fields of machine learning, such as the MNIST handwritten digits benchmark [2]. Realistic simulations are often used [3] for domains where real data is hard to collect or where real data would be subject to business confidentiality.

Fault-free simulations of flexible assembly systems are generally available [4] and have been studied for visualization and optimization purposes. These are not directly suitable for benchmarking fault detection methods.

II. Problem Formulation

A standard benchmark should reflect a realistic task and it should have enough data to be useful.

Real data regarding flexible assembly system faults is not generally available, because fault data is generally subject to business confidentiality in flexible assembly industry. The generally available simulation models do not typically contain failure modes and are not typically designed to emit log structured events in a realistic fashion, but they can be used as a starting point in creating a flexible assembly system model.

To enable relevant research into anomaly detection in flexible assembly systems a realistic simulator for log structured events is needed. All references to simulated faults and failure modes in the research described in this article are hypothetical and should not be considered to reflect actual characteristics of any specific existing systems. An effort is made to make the simulations contain the necessary features of generic and realistic failure modes to make the models useful for research, but for example the simulated fault frequencies and the configuration of the flexible assembly system along with workflows and end products are arbitrary and hypothetical.

A simulation for benchmarking anomaly and fault detection methods should not be fully deterministic to reflect the real world dynamics. The simulation should also include realistic faults with possible early indicators such as variations in delays in the steps of the process.

III. Modes of Failures in Flexible Assembly Systems

Flexible assembly system failures are typically component failures where a failure of a single module or a component will lead to the system becoming inoperational. Since the flexible assembly systems are dynamic systems with numerous moving parts, the wear and tear of the components and tools are a significant source of downtime. In addition to the component failures, certain systemic failures can cause downtime, such as power loss and network communication failures.

The systemic failures mentioned don't typically show any indication before they happen so predictive maintenance has little potential to be applied there. Tool wear and tear depends on the workload. Typically tools that wear out fast, for example multiple times for one unit of work in a cell, receive much attention and such tools are generally well maintained. Tools that have wear and tear but have longer life spans are more susceptible to being overlooked by operators and might have

TABLE I
TABLE OF RECOGNIZED FAULT TYPES IN FLEXIBLE ASSEMBLY SYSTEMS

Fault type	Unexpected faults	Potentially has an early indication	Useful target for anomaly detection
Auxilliary component wear and tear	X	X	X
Human error, misconfiguration	X	X	X
Human error, failed manual step	X	X	X
Incompatible parts, materials or replacement tools	X		
Systemic failures (power, network)	X		
Tool wear and tear for quickly degrading tools		X	
Tool wear and tear for slowly degrading tools	X	X	X

some indicators of degrading before a failure. Cranes, hatches and conveyors have relatively long life spans and are primarily maintained in periodic preventive maintenance. If we could get indicators for impeding failure for these components, the periodical maintenance could be scheduled earlier to prevent downtime. In addition to these physical faults the flexible assembly system can suffer from human errors. Human operator can inadvertently misconfigure the flexible assembly system so that its operating mode changes unexpectedly, for example by disabling a cell which can lead to the system stopping without a proper failure. In addition to configuration errors, human operators might fail in manual assembly steps for example by pressing the button to mark the step as completed too quickly. The flexible assembly system might also get in incompatible materials, parts or replacement tools and fail when trying to use them.

For useful anomaly detection systems we need to concentrate on faults that are unexpected, and that potentially have early indications. The table I summarizes the classes of faults.

IV. Flexible Assembly System Model

The simulation consists of a realistic model of a flexible assembly system with a class of faults randomly injected into the process. The simulation will not be fully deterministic and contains soft faults which do not affect operation in addition to hard faults which result in immediate downtime.

The simulator is implemented in Python and it generates a JSON file which models the logs from the system. The log message fields are defined in Table **??** and and example message is shown in **??**.

The simulation framework consists of a scheduler that takes the next event from the event queue and applies it to the respective module. The simulated modules execute when events are applied to them and generate new events and immediate log entries with timestamps and other metadata. The simulated modules have optional active failure modes that affect the simulated operation of the component through generated log messages or the subsequent generated events. Failure modes are set up to the simulated modules in a separate failure module. The overall framework sets up the simulation and events.

A. Normal Operation of the Simulated System

The simulated hypothetical system assembles and tools a car transmission block loosely inspired by a Youtube video of Chrysler transmission assembly [5]. The workflow consists of several manual assembly steps in separate cells and transporting the subassemblies between the cells by the means of cranes and conveyors. Multiple transmission blocks are being assembled simultaneously in separate cells.

The transmission component consists of four subassemblies. The main sequence starts from the frame of the transmission block and continues through several manual steps done in separate cells. The steps of the main sequence are given in the Tables II and III. Each step can contain a separate sequence of events and associated log messages. The mean durations for manual steps can vary in normal operation $\pm$ thirty percent. The mean durations for automatic steps can vary $\pm$ five percent.

B. Failure Modes

The previous subsection describes the normal operation of the system. The failure modes of the manual steps include human errors by pressing OK prematurely without properly executing the step. This leads to the next manual step failing and returning the block back to the previous step or removing it from the assembly line altogether.

V. Evaluation

The baseline benchmark is done by automatically inferring a deterministic finite automaton from the fault-free log events as was shown by Langer et al. [6]. This trivial model does not work very well, but can be used as a baseline for more advanced methods. The finite automaton is then used to evaluate different logs generated to contain certain faults.

VI. Results

VII. Conclusion

The flexible assembly model simulator is published in GitHub [7] as open source.

Additional research is needed to improve this benchmark simulation by using experiences from real flexible manufacturing systems and their modes of failure and respective future fault indicators. I will be continuing this work in the future in association to my future research conserning automated anomaly detection in flexible assembly systems.

TABLE II
TABLE OF THE STEPS 1-16 IN THE MAIN ASSEMBLY PROCESS

Step	Description	Duration	Log messages	Potential failures
1	Crane 1	30 s	Going forward, stopping, going back, stopping	Wear & tear
2	Manual inspection	37 s	OK pressed	Failed manual step
3	Conveyor	30 s	To cell, stop, to next cell	Wear & tear
4	Bowl feeder gives components	5 s	Given	Wear & tear
5	Add components	21 s	OK pressed	Failed manual step
6	Conveyor	30 s	To cell, stop, to next cell	Wear & tear
7	Bowl feeder gives components	10 s	Given	Wear & tear
8	Add components	34 s	OK pressed	Failed manual step
9	Conveyor	30 s	To cell, stop, to next cell	Wear & tear
10	Crane with subassembly A	10 s	Going forward, stopping, going back, stopping	Wear & tear
11	Combine with subassembly A	34 s	OK pressed	Failed manual step
12	Conveyor	30 s	To cell, stop, to next cell	Wear & tear
13	Conveyor with subassembly B	10 s	To cell, stop	Wear & tear
14	Combine with subassembly B	35 s	OK pressed	Failed manual step
15	Conveyor	30 s	To cell, stop, to next cell	Wear & tear
16	Bowl feeder gives components	5 s	Given	Wear & tear

TABLE III
TABLE OF THE STEPS 17-31 IN THE MAIN ASSEMBLY PROCESS

Step	Description	Duration	Log messages	Potential failures
17	Conveyor with cover	10 s	To cell, stop	Wear & tear
18	Add cover and screws	76 s	OK pressed	Failed manual step
19	Conveyor	30 s	To cell, stop, to next cell	Wear & tear
20	Tighten the screws	28 s	OK pressed	Failed manual step
21	Conveyor	30 s	To cell, stop, to next cell	Wear & tear
22	Conveyor with subassembly C	10 s	To cell, stop	Wear & tear
23	Combine with subassembly C	60 s	OK pressed	Failed manual step
24	Conveyor	21 s	To cell, stop, to next cell	Wear & tear
25	Tighten the screws	16 s	OK pressed	Failed manual step
26	Conveyor	21 s	To cell, stop, to next cell	Wear & tear
27	Bowl feeder gives components	5 s	Given	Wear & tear
28	Add components	11 s	OK pressed	Failed manual step
29	Conveyor	21 s	To cell, stop, to next cell	Wear & tear
30	Tighten the screws	32 s	OK pressed	Failed manual step
31	Conveyor	21 s	To output gate	Wear & tear

REFERENCES

[1] M. Donath and R. J. GRAVES, "Flexible assembly systems: an approach for near real-time scheduling and routeing of multiple products," *The International Journal Of Production Research*, vol. 26, no. 12, pp. 1903–1919, 1988.

[2] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[3] G. Jäger, S. Zug, T. Brade, A. Dietrich, C. Steup, C. Moewes, and A.-M. Cretu, "Assessing neural networks for sensor fault detection," in *Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 70–75.

[4] G. Rosati, M. Faccio, C. Finetto, and A. Carli, "Modelling and optimization of fully flexible assembly systems (f-fas)," *Assembly Automation*, vol. 33, no. 2, pp. 165–174, 2013. [Online]. Available: http://dx.doi.org/10.1108/01445151311306690

[5] Cars, "Chrysler transmission assembly line," 2014. [Online]. Available: https://www.youtube.com/watch?v=447sSED91v0

[6] F. Langer, K. Bertulies, and F. Hoffmann, "Self learning anomaly detection for embedded safety critical systems," *Schriftenreihe des Instituts für Angewandte Informatik, Automatisierungstechnik am Karlsruher Institut für Technologie. KIT Scientific Publishing*, pp. 31–45, 2011.

[7] T. Keski-Valkama, "Flexible assembly simulation: FAS-Simulator," https://github.com/keskival/FAS-Simulator.

Tero Keski-Valkama Tero Keski-Valkama is working as a software architect in Cybercom Finland Oy. He has been programming neural networks since high school.