State machines simulator for distributed decision system

T Savu$^{1,*}$ and B A Jugravu$^{1}$

$^{1}$ Manufacturing Engineering Dept. (TCM), University POLITEHNICA of Bucharest, 313 Spl. Independentei, 060042 sector 6, Bucharest, Romania

E-mail: savu@ctanm.pub.ro

Abstract. The amount of computing power assigned to the intelligent components of a production system must be justified by the benefits brought following the decisions these components are taking. The paper proposes a state machines and messages exchange based model for developing simulators used to estimate the system’s behavior and performance in different configurations. A simple system was modeled and simulations were performed for different numbers of electric transporters in the system, different charging times and for different degrees of independence concerning the decisions taken by the manufactured products. It was possible to determine the most effective number of transporters, a value which is increasing, as expected, with the charging time. A second conclusion was that allowing the manufactured products to decide themselves concerning the transporter to use at a certain moment does not necessarily increase the overall system’s performance.

1. Introduction

When defining what a “next generation manufacturing system” means today, more and more opinions are related not only to the modern manufacturing methods, like nano-technologies or additive technologies, but also to the means and methods for organising the systems. Descriptions like “production systems that can think for themselves” or “self-organizing manufacturing processes for highly customizable products” [1] are referring not only to the decrease of human-performed tasks but mainly to the fact that the knowledge which is driving the system is now placed in the system’s hardware and software itself, most of the times together with the inference engines, sometimes called artificial intelligence, which are updating the existing knowledge assets.

Placing the knowledge, including the decision power, in the non-human assets of the production system means nowadays not only using centralised computing systems like ERP software. The cyber-physical systems part of the fourth industrial revolution, represented by the intelligent machine-tools, autonomous ground industrial vehicles and more often by the manufactured smart products themselves, are now able to perform part of the knowledge processing and act like intelligent agents.

A balance has still to be found, since the computing capabilities of the cyber-physical systems must be justified by the benefits obtained when distributing the decision power to the machines, vehicles and even products.

Estimating the system’s behavior and effectiveness in different configurations can be performed by modeling and simulation. The paper proposes an original model where system’s components (products, machines and transporters) are acting like state machines [2] which are exchanging messages and have different decision capability levels assigned to them. Decision algorithms can be used during simulation and the effects on the overall system’s effectiveness, in terms of total
processing time, can be estimated also for different values of the system’s parameters (number of products, processing times, number of electrical transporters and charging time).

2. Model description
A UML state machine diagram of the three components types is presented in figure 1. When in the Idle state, if a batch didn’t complete its process yet, it must launch a transport order to be moved to the next working place, if that is different from the current one. After launching the transport order, the batch is placed in the Order sent state and is waiting for an acknowledgement of order’s receipt.

When a transporter is in its Idle state and transport orders exist, the transporter is choosing one of the existing transport orders. If the closest waiting batch is found in the transporter’s current position, then the transporter is issuing a Loaded message to the batch, is updating its destination and path data according to the transport order and is putting itself in the Moving state. If the closest waiting batch is in another position, then the transporter is sending a Trans order received message to the batch which issued the order, is updating its data about the path to be followed and is putting itself in the Moving state to the ordering batch.

When receiving a Trans order received message, a batch is moving itself from the Order sent state to the Wait for transport state and it’s updating its data about the transporter which must arrive.

When arriving at a place from which an order was issued, a transporter is sending a Loaded message to the batch which issued the order, is updating its data about the path to be followed and is moving toward the destination specified in the transport order. When receiving the Loaded message, a batch is putting itself in the Moving state and is updating its data about the transporter which loaded it.

After completing a transport order and arriving at its destination, a transporter is sending an Unloaded message to the carried batch and is putting itself in the Idle state. When unloaded, a batch is putting itself in the Idle state, if the unloading place is a warehouse, or in the Wait for process state, if the unloading place is a working one.

When in its Idle state, a working place is checking for batches in its input place, in the Wait for process state. If such a batch is found, the working place is sending it a Process started message, is extracting the operation time from the batch’s process data and is switching in the Busy state. When receiving the Process started message, a batch switches to the In process state. When the operation finishes, the working place is sending a Process ended message to the batch and switches in the Idle state. When receiving the Process ended message, a batch switches to the Idle state.

![Figure 1. Model’s UML state machine diagram.](image-url)
If a batch was just released from a working place and the next operation in the process must be performed in the same place, then the batch is not issuing a transport order, but is placing itself in the **Wait for process** state.

3. Simulator’s description

### 3.1. Simulated system’s structure

For testing the previously described model by simulating a specific case, it was chosen a system (figure 2) consisting in 12 working places (PL 001 to PL 012) and a warehouse (PL 013) where the batches arrive in the system and from where they are leaving the system. Loading and unloading from the warehouse can be performed using one of the places P13 to P18. Loading and unloading the batches in or from the working places is performed using places P1 to P12. When moving from one place to another, a transporter is using a straight line, if possible, or is using intermediate places P19 to P24 for changing the direction. For example, a transporter can move directly from P6 to P14 or from P1 to P15, but for moving from P2 to P18 it must use a path like P2 – P20 – P18 or P2 – P23 – P24 – P22 – P18. Place P25 serves as a recharging station for the electrical transporters’ batteries.

In the first simulator version, the transport orders are launched by the batches without mentioning a specific transporter. When a transporter is in its **Idle** state and transport orders exist, a transporter is choosing the order which was issued from the closest point from the transporters current position.

If a batch finished the last operation of its process, it will have to be moved back to the warehouse, so the batch will issue a transport order having as destination one of the P13 to P18 places. After completing its process and returning to the PL 013 warehouse, a batch is deleted from the system.

### 3.2. Simulation parameters and implementation

A batch of parts is defined initially by an ID and a description of the process it must complete. The process description consists in an array of records, one record for each operation in the process. Each record contains an ID of the corresponding working place and a processing time. Initially, all batches are placed in the facility’s receiving warehouse, each one in an **Idle** state.

A transport order record contains the ID of the batch which launched it, the source and destination places’ IDs and an earliest time when the batch will be available at the source point. The distances from the transporter’s current position to the issuing points are computed using an implementation of the Dijkstra algorithm [3].

![Simulated system’s structure](image)
A Capacity value is assigned to every transporter, representing the remaining capacity of its electric battery, initially 100%. While moving, the remaining capacity decreases with time, both with a fixed quantity on each time unit and with a variable quantity, proportional to a Mass value of the transported batch. When a task is finished and the transporter enters the Idle state, if the remaining capacity is less than 20%, the transporter goes to the charging station P25, where is spending a constant amount of time. When charging finished, the transporter enters the Idle state and its capacity is set to 100%.

The simulator was implemented using LabVIEW graphical programming environment [4, 5]. An annotated user interface is shown in figure 3, where real time data from batches and transporters are displayed, together with a simple graphical representation of the components’ positions and with the transporters’ remaining capacities graph.

The simulation software was configured to generate sets of 50 batches, the number of operations in one batch’s process being uniformly distributed between 5 and 10. The operations from all processes were assigned to working places PL 001 to PL 012 following also a uniform distribution. Operations’ durations were randomly generated with a uniform distribution between 60 and 240 time units / batch. Batch masses were assigned uniformly distributed values between 10 and 200 units. The batches are initially randomly created in places P13 to P18.

Simulations were performed for different numbers of transporters in the system, from 1 to 7, and for different charging times, 100 and 500 time units. Each simulation consisted in 30 separate runs.

![Figure 3. LabVIEW simulator’s user interface.](image)

3.3. Simulation results
When using only one transporter, many batches are leaving the warehouse very late. This is because, after the first batches are completing their first operations, they are closer to the transporter than the batches still in the warehouse, so their transport orders are accepted with higher priority. On the 30 runs set, one transporter completed the job in an average of 12773 time units when needed 100 time units for each recharge and in an average of 17350 time units when one recharge took 500 time units.

Not depending on the recharging time, in both cases 11 – 12 rechargings were needed.

The average total time is represented vs. the number of transporters, both for the 100 and 500 units charging time, in figure 4. The main conclusion is that, for more than two transporters, there is no significant reduction of the average total time, nor for the 100 units or the 500 units recharging times.
Also, for three or more transporters, increasing the charging time does not lead to an increase of the average total time.

Reasons for the above-mentioned conclusions can be found when comparing the remaining capacity vs. time graphs for different number of transporters (for three transporters in figure 5 and for seven transporters in figure 5). Smaller decreasing slope of the remaining capacity shows a less busy transporter, needing to recharge at larger time intervals. In the beginning of the simulation, the graphs for the different transporters are keeping together, as all the transporters are busy to move the batches from the warehouse to the working places. This “common graphs” interval is longer for smaller number of transporters, because it takes a longer time to move all the batches from the warehouse with less transporters. All the batches are launching their initial transport orders at the same time, so the beginning of the simulation is a busy time for the transporters fleet. After all the batches left the warehouse, transport orders are launched only when batches are finishing their operations. The needed work intensity of the transporters depends now only on the technological processes, so ne transporter is less busy when there are more transporters in the fleet.

Figure 4. Average total time vs. number of transporters, for two charging times (CT).

Figure 5. Remaining capacity vs. time for three transporters.
3.4. Changing batch behavior

In the second version of the simulator, when a batch is launching a transport order, the closest idle transporter is selected by the batch and its ID is mentioned in the order data. If there are no transporters in the Idle state at that moment, then the order is launched without mentioning a specific transporter.

A transporter in the Idle state is first searching the orders which were assigned to it and is choosing the one which has the closest starting point. If no orders were specifically assigned to it, then the transporter is looking to choose from the orders which have no transporter ID in their data structure.

Using the same values for the number of batches, number of operations for one batch and processing times, it can be concluded that there are no significant differences for the average total time (figures 7 and 8) when the batch behavior is changed.

4. Conclusions

It was found that using the state machines and messages exchange based proposed model provides an adequate solution for easily developing a robust and scalable simulator which allows changing the system components’ behavior and customising the data sets to be computed.

Although the simulation parameters were limited to a very specific case, the nature of the conclusions can be easily replicated for more complex real situations. Batch behavior will be further modified to be able to choose one from a group of similar working places and, when choosing a transporter, it will be possible to account for busy transporters which will become idle soon. A
transporter will not move to the recharging station when below a fixed limit of its remaining capacity but will try to accomplish fewer demanding tasks by estimating the necessary energy consumption. A more complex description of the recharging stations will be implemented, including one in which discharged batteries are just swapped with newly charged ones, thus decreasing the charging time of a transporter.

References
[1] Meshel R and Feuer B Z 2015 Next Generation Manufacturing: Production Systems that Think Siemens PLM Corporate Blog https://tinyurl.com/yxhxjl9v, accessed in February 2019
[2] Wagner F, Schmuki R, Wagner T and Wolstenholme P 2006 Modeling Software with Finite State Machines: A Practical Approach (Boca Raton: CRC Press – Taylor & Francis Group)
[3] Blokdyk G 2018 Dijkstra’s Algorithm: End-to-end Data Analysis (Create Space Independent Publishing Platform)
[4] Bitter R, Mohiuddin T and Nawrocki M 2017 LabVIEW: Advanced Programming Techniques, Second Edition (Boca Raton: CRC Press – Taylor & Francis Group)
[5] Bress T 2013 Effective LabVIEW Programming (Austin: National Technology & Science Press)