Benchmarking of Manufacturing Control Systems in Simulation

Bart Saint Germain, Constantin Zamfirescu, Olaf Bochmann, Paul Valckenaers and Hendrik Van Brussel
Department of Mechanical Engineering, Katholieke Universiteit Leuven
Celestijnenlaan 300 B, B-3001 Leuven, Belgium
E-mail: Bart.SaintGermain@mech.kuleuven.ac.be

Abstract
This paper discusses a general framework in order to get accurate and as fast as possible simulations in the field of manufacturing control. The approach described in the paper gives the opportunity to have a benchmarking tool that gives the users a clear idea about the performance of their different strategies. The tool makes it possible to make decisions based on information on expected performance, also under non-nominal conditions. This simulation tool differs from conventional simulation approaches in two ways.

The first property is the distinction between the emulation and the control. Whereas the emulation is the reflection of the reality, a model of the entity behaviours in the factory. The control is the logic to control the factory or in case of a simulation, the emulation.

Secondly the system proposed will simulate as fast as possible without getting unrealistic results. Most of the current systems need to have a trade off between simulation speed and accuracy of the results.

Keywords:
Benchmarking, Manufacturing control, Emulation, Flexible production systems, Performance evaluation

1 Problem Statement

Today, manufacturing industries are aware that designing the right kind of flexibility into their forthcoming production systems is vital to their future competitiveness. However, industry is lacking the tools and the methodology to evaluate the performance of alternative flexible designs, which are being considered for the next production facility, regardless whether this is a green field (building a new factory from scratch) or a brown field (renovate/reuse a factory) development. Indeed, it no longer suffices to consider operations under nominal conditions to assess performance. The raison d'être for the flexibility in a factory design is to have robustness in the face of abnormal situations. As a consequence, the analysis of models that make significant simplifications of the envisaged manufacturing plants is no longer capable of predicting actual plant performance.

The alternative is to build simulation models of the alternatives for a forthcoming flexible production system. These simulation models have to be accurate. In particular, they must include the advanced manufacturing control, which exploits the flexibility in the underlying system and which is instrumental to achieving the desired robustness in the face of change and disturbances. To realize this approach, a technology is required with the following properties:

1. It must be possible to simulate faster than reality such that a typical performance analysis campaign can be executed in weeks. Ideally, it must be possible to simulate months of production in a couple of hours computer time.

2. Validation of the simulation must be fast and easy.

3. It must be possible to develop the simulation model in a few weeks. In particular, the simulation of any configuration of production equipment must be generated automatically, given simulation code for the equipment itself. The development of code for production equipment probably will take
some (incompressible) time, but should be fast if the equipment is some variant of equipment for which the code already exists.

This paper addresses the first and second requirement. The third requirement is outside the scope of this paper. The second paragraph addresses current simulation systems and approaches. Paragraph 3.1 discusses a solution to the second requirement. Paragraph 3.2 presents a solution to the first requirement. Finally, conclusions are given.

2 Current Simulation Systems

To address the first requirement above, conventional discrete event simulation systems execute as fast as possible. To achieve this, they maintain an internal event calendar from which they process the events, which have been inscribed in that calendar, in chronological order. The processing of an event typically causes new events to be inserted in the calendar at some future moment. For instance, processing an event corresponding to the arrival of a work piece at an idling machine will change the machine status to busy and insert the end-of-processing event in the event calendar. To simulate as fast as possible, the simulation program jumps from the current event to the next without delay.

In the above, the implicit assumption is made that decision-taking happens instantaneously. This causes problems with the simulation of manufacturing system that are equipped with an advanced manufacturing control system, which requires some time to “think” and communicate. A few simulation systems provide a brute-force solution: switch to real-time mode in which the processing of the next event is delayed until the specified time has passed. This is not really acceptable because simulation speed is reduced to impractical levels and some technical issues still remain in those existing simulation systems. In other words, the technology to measure the performance of advanced manufacturing control systems in simulation is missing.

Furthermore, the conventional simulation approach utilizes models that simplify the targeted production system and, as a consequence, often blend the model of the manufacturing system with its control (typically equipment and control is aggregated). This creates a tricky validation problem: the simulation developer has to check that the behaviour of his simulation corresponds to the behaviour of the manufacturing system including its control. The simplifications and aggregations destroy the one-to-one connections between the envisaged reality and the simulation model. This makes validation difficult, but more importantly, it prevents re-validation after some model modifications from benefiting from the validation of the original model.

The next paragraph addresses the above problems. Section 3.1 addresses the validation issue with an approach that has additional benefits when the chosen alternative needs to be implemented. Section 3.2 addresses the reconciliation of fast simulation with advanced control.

3 Novel Simulation Approach

A simulation approach, and corresponding software, has been developed to address the mentioned problems.

This paragraph describes the different parts of the simulation framework together with a solution in order to get smooth and accurate simulations. Section 3.1 discusses the different parts in the simulation tool, emulation and control. The advantages from such a system comparing to conventional simulation approaches are discussed in detail. A second paragraph of section 3.1 specifies how this simulation approach helps the users to create benchmarks. Section 3.2 goes deeper into the solutions in order to run the simulation as fast as possible without creating inaccurate results.
3.1 Simulation = Emulation + Control

3.1.1 Simulation approach

Simulation refers to a broad collection of methods and applications to mimic the behaviour of real systems. Simulation as described in this paper comprises two separated parts: emulation and a control subsystem.

The emulation reflects the emulated world as it is and provides the state of the emulated world to the control. All parts in the emulation have a close (one-to-one) correspondence to parts in the physical world. The emulation parts aim to mimic the behaviour of their physical counterpart. These parts form a set of building blocks suitable for building different compositions, e.g. different factories. The control on the other hand encapsulates all decision logic of the simulated system. It sends the appropriate commands to the emulation.

In the system described in this paper, modelling of a complete factory can be decomposed into modelling of smaller entities. This decomposition is not a functional decomposition but is a reflection of the reality. In this way the modelling is reduced to a one to one mapping with the reality. Because the world fits together, no real conflicts will arise while modelling, only some small syntactic problems can arise.

In the world of factory simulation, two types of emulation building blocks exist: “Temporary Entities” and “Processing Units”. Temporary entities differ from processing units in their life cycle. Temporary entities will typically have a short lifecycle and they need to go through the factory in order to undergo some operations. Operations are performed via processing units, which have a much longer lifecycle.

The processing units have one or more “buffers”, input or output buffers. These buffers are connected among each other and form in this way the topology of the manufacturing facility. Connections have of course their capabilities and constraints; some products can be moved via a connection whereas others are blocked. To address this connection functionality, the connection is mapped in a software component: “connections”.

![Diagram of general building blocks of the emulation](image)

Figure 1, general building blocks of the emulation

The question why the separation of the emulation and control is of such importance is not specific to this domain. Many research domains shifted from an approach where the world of interest was aggregated with the control toward a clear separation between the components. In the domain of developing hardware for example, we see a shift towards an architecture controlled by a changeable (software/firmware) component.

The decomposition into a one to one mapping with reality has several benefits:

- The mapping is easier. The only concern is to look at the behaviour of individual manufacturing entities and to reflect this in software components without being exposed to the overall complexity.
- The validation can be done for every unit and is not exposed to the overall complexity. By matching each single component, the global behaviour will also match the reality as it is.
- The concepts of the Object Oriented approach like specification, generalization, and aggregation… can be used among the manufacturing facility units. In this way the approach does not prevent you to make a simple model first, and extend it later on.
- Mapped units can be reused in new topologies or other manufacturing facilities.

In conventional simulations the border between emulation and control is not enforced. All one to one relationships may be lost because the control is mixed up into the emulation. In this way the modelling of factory becomes a very complicated and precise work. The modelling and the validation of the case have to happen on a global view. It is very difficult to predict how the control influences the behaviour on individual parts. Reusing of components is also a much harder task whenever you have to pull out the needed emulation piece out of the scrambled simulation part.

### 3.1.2 Re-usability

Research in the domain of object-oriented programming has pointed out that the reusability of different software components has a significant impact on the development speed and the accuracy of the developments. This property is also valid in the domain of simulations. To enable a fast development and validation, the simulation has to be as reusable as possible.

- **Emulation divided in different reusable components.**
  
  As mentioned, the emulation is divided in several emulation entities. These entities reflect the behavior of a physical (not artificial) part in the factory. Emulation entities can be reused in other topologies. The emulation entities are the building blocks to model new factories or to adapt the topology of the current factory.

- **The emulation can be reused for many different controllers**

  Manufacturing control benchmarking requires that different controllers can be tested on similar situations. For this reason, the simulation requires the emulation to behave in the same way for every new control benchmark. Due to the separation of the emulation and control, the emulation can be transferred, reused for several control implementations. To minimize the emulation subsystem redesign, the emulation is made in such a way that it is not exposed to specific control aspects.

- **The control can be reused for many different emulation types or topologies**

  Simulation is not developed for only one factory topology or factory type. The control subsystem can be tested on several topologies.

- **The control is not only made for simulation purpose. The control can be plugged into the real factory.**

  The purpose of the control system must be seen broader than only for simulations. To the control subsystem, ‘being connected to a specific emulation’ should be indistinguishable from ‘being connected to the real world’. Consequently, it is easy to connect the control system to the real factory instead of its emulation.

The reusability of the different software components faces to some limitations. These limitations are minor difficulties in the emulation-control issue. All difficulties are of the same nature of a one to one translation.

- **The emulation has a set of pre-made emulation entities.** The user of the emulation is capable of making different emulation models, as a composition of several emulation entities. The only limitation a designer has to take into account is that the composition reflects a feasible, realistic situation: The emulation model is a blueprint of an imaginable real factory.

- **Different emulation and control subparts can be combined to construct a simulation run.** Every subpart, emulation or control, is made within its own scope. It is imaginable that an emulation subpart, built to be used in a certain domain, cannot serve for a control that requires other
functionalities. For example: a control system built to handle only job shop factories cannot be used with flow shop emulations. In terms of software, this means that control and emulation subparts need a well-defined interface. Only subparts with matching interfaces can be used for a simulation run.

- To switch from simulation to a real control of a factory there are no conceptual limitations whenever the emulation and control are separated, the control is not exposed to emulation specific properties. This does not mean that a control developed for simulation purposes can be directly transferred to the real factory. There are some differences to take into account:
  - Transactions, backup, …
  - Observe-ability of the underlying manufacturing system
  - Command-ability of the underlying manufacturing system
  - Connections to other systems
  - …

3.1.3 Benchmarking approach
In this section a benchmarking approach is described to select or measure how good or bad a control system works on several factory types or different topologies. This section emphasizes the need of a simulation tool with the properties described in this paper. The benchmarking algorithm specified in this section will start from several control and emulation implementations to select and implement the “best” control system in the shop floor.

As explained in the previous section, a control or emulation subsystem can be connected to a subset of all available emulations or control subsystems respectively. Moreover, because all logic of the two subsystems is encapsulated, the approach will not force you to re-implement the control whenever the real factory has to be managed.

In general, the benchmarking procedure comprises the following steps

- Select alternative plants and model them into the emulation (E). In green field situations, the topologies of the emulated plants can be chosen freely. Brown field developments have to account for the constraints imposed by an existing situation. In order to build such a model, the users can use the already predefined emulation building blocks. New types of building blocks have to be implemented and verified.

- Select alternative control implementations. (C)

- Simulate and evaluate the different possible emulation-control combinations(Evaluate(Ei,Cj)). For evaluation, appropriate statistical models are essential to avoid making serious errors leading to fallacious conclusions and, ultimately, poor decisions. The description of the different statistical models is out of scope for this paper, more information can be found in the literature. (M. Law, Averill, Kelton, W. David, 2000). Eventually there is a combination of a controller and emulation that performs the best (Ex, Cy)

- As last step, the control should be disconnected from the emulation and connected to the real plant. This requires extra attention to robustness, integration,… (C’y)

Formally the benchmarking algorithm looks as follows:

\[
E = \{E_i : i = 0.. n\} \\
C = \{C_j : j = 0..m\} \\
\{Evaluate(E_i, C_j) : E_i \in E, C_j \in C, match (E_i,C_j) = true\} \rightarrow Ex,Cy \\
Cy \rightarrow (factory,C’y)
\]
Simulations and especially those where control and emulation are separated can help to use this general algorithm. The main problem with the normal simulations is that to obtain a good idea of the improvement of the chosen control or emulation subsystem, any irrelevant subpart must not influence the result. Due to the decomposition, the emulation approach will give you a better idea of what really would happen.

3.2 Real-Time Simulation + As-Fast-As-Possible
In the solution, discussed in this paper, the need to account for deliberation and communication time from the control system is reconciled with the need to speed up the simulation as much as possible at every opportunity. This paragraph presents how this has been achieved.

3.2.1 Discrete event emulation
The emulation part of the overall simulation uses discrete event simulation (DES) to execute the events as fast as possible (M. Law, Averill, Kelton, W. David, 2000). Discrete event emulation is closely related to discrete event simulation. Although we refer not directly to DES because it would be confusing for the reason stated in the previous sections (emulation + control = simulation).

Discrete event simulation concerns the modelling of a system as it evolves over time by a representation in which the state variables change instantaneously at separated points in time. These points in time are the ones at which an event occurs, where an event is defined as an instantaneous occurrence that changes the state of the system.

As in the object-oriented approach, there is a distinction between the state of a class/emulation-entity and the behaviour.

- The state of a system/manufacturing facility is defined as a collection of variables necessary to describe a system at a particular point. In the case of a conveyor belt this includes the number of products currently transported and the respective positions of these products.
- The behaviour is the logic to change the state triggered by an event. It contains all knowledge to evaluate the situation and respond to the respective event. Behaviours can trigger new events, on the times where the state has to be re-evaluated. E.g. the “new order” event will trigger a next “new order” event according to a certain distribution. In this case the breakdown will happen at time 60.

In a first example, figure 2 shows the situation where a breakdown happens while an order moves on this conveyor belt. After some time the conveyor belt is repaired.

- The conveyor belt will start on time 20. The “PowerOn” event placed at the beginning of the simulation is put at the respective time on the event calendar.
- As consequence of starting the conveyor belt there is a breakdown triggered according to a certain distribution. In this case the breakdown will happen at time 60.
- From the control there is a command to start moving an order on the conveyor belt. This operation should start on time 40, defined by the control.
- Arrived at time 40 the “Start on Conveyor Belt” behaviour will insert a theoretical, if everything would go as foreseen, end event. In this case the move operation would take 80 time units so, the “End On Conveyor Belt event” will be inserted in time 120.
• A breakdown event was inserted in the event calendar at time 60. This means that the orders currently moving on the conveyor are blocked. To repair the conveyor belt it takes 40 (time 100) emulation units.

• Once the Conveyor belt is repaired, at time 100, the orders currently blocked can go on. This repair event will insert a new adapted “EndOnConveyorBelt” event at time 160; 120 + 40.

• The emulation calendar is now on the point where the first end event was placed, this is time 120. Due to the fact that the orders were blocked, no action will be triggered at this time.

• The last event in this example will change the state of the order, the order will be marked to be at the end of the conveyor belt.

![Figure 2: Discrete event model of conveyor belt with breakdown](image)

In the second illustration (Figure 3) a speedup event is handled. In this case, the adjusted end event should come before the original one. This adjusted end event will update all states, the state of the conveyor belt and the state of all product of the respective processing unit. The second end updater will notice again that the state was already changed.

![Figure 3: Discrete event model of conveyor belt with speedup](image)
3.2.2 Accurate emulation

The crucial point for the discrete event simulation/emulation is that every relevant event in the system is modelled. Obvious events as start, stop, breakdown, … events are necessary but not sufficient to get an idea of what is going on in the manufacturing facility. Additionally, decision events are needed in the emulation to accurately reflect the factory operations. A typical example of a decision event is when a resource performed an operation on an order. This will invoke decision strategies in the control part to decide where and what operation should come next.

Like the physical operations, the decision-making activities can be decomposed into discrete events. With a very simple control system this still is feasible. A static control system for example, like FCFS\(^1\), can be implemented in such a way. This is not the case for an advanced control system where there are for instance agents or holons, each responsible for a subpart of the system, have a certain level of autonomy. In such a Holonic architecture (Valckenaers, P, Van Brussel H. Kollingbaum M. Bochmann O., 2001), the global behaviour is formed by the aggregation of the decisions and actions of the different holons. It would be a hard task to create a discrete event model of this behaviour. Even if the mapping would be possible, the separation between the control and the emulation would be too weak.

To overcome this difficulty, a real-time component is added to the emulation. This real-time component will make sure that the control gets his time to “think”, as it would be in reality. In order to achieve this, the emulation needs to trace all events towards the emulation. Accurate emulation can be achieved by going into real-time whenever there are control decisions pending and jump as fast as possible in the other cases. The control on his turn must inform the emulation whether there is still a chance that there is a decision commitment towards the emulation.

To give a better idea of how such a system works, two examples are given. The examples are a chunk of a simulation run where the next event in the virtual world is the creation of a new order. This creation happens on time 10210. The emulation will inform the control and the control will on its turn notify the emulation what the next operation should be. This control decision takes some time.

In the figures Figure 4 and Figure 5 we see both the emulation and the control represented as timelines. The execution time line represents the control and both the event calendar and the emulation execution time line are part of the emulation. The arrows between the emulation and the control represent messages. Typical messages are: “NewOrder”, “ProcessOperation”,… The arrows from the emulation time line to the event calendar indicate that there is a new event placed on the event calendar.

In the first example visualised in Figure 4 and Table 1, the control is relatively fast. The emulation factor is defined as, one emulation unit corresponds to one millisecond in the reality.

The second example shows how the emulation reacts on a slow controller; the decision to know to next operation takes about 510 milliseconds. For simplicity, the emulation factor in this example is still the same, one emulation unit for one millisecond.

Note that the scale of the emulation time and the scale of the computer time in Figure 4 is not the same. Although a message or a new event arrow goes visually backwards, the new event is placed into the future, e.g. When placing the “StartOperation”, event 5, the arrow goes visually backwards but still the event is placed into the future.

\(^1\) First Come First Served
a. Fast controller, emulation speed factor = 1/1

![Diagram showing synchronization between emulation and fast controller]

**Figure 4: Synchronization between the emulation and a fast controller**

<table>
<thead>
<tr>
<th></th>
<th>Computer Time (msec)</th>
<th>Emulation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process “New Order” event</td>
<td>110</td>
</tr>
<tr>
<td>2</td>
<td>insert “New Order” event (after 400msec)</td>
<td>111</td>
</tr>
<tr>
<td>3</td>
<td>Notify control</td>
<td>112</td>
</tr>
<tr>
<td>3</td>
<td>Control receives new order message</td>
<td>113</td>
</tr>
<tr>
<td>4</td>
<td>Control reports to emulation (do operation1 within 60msec)</td>
<td>121</td>
</tr>
<tr>
<td>5</td>
<td>Emulation receives “New Order” response (do operation1 within 60 msec)</td>
<td>122</td>
</tr>
<tr>
<td>6</td>
<td>Process “StartOperation” event</td>
<td>123</td>
</tr>
<tr>
<td>7</td>
<td>insert “EndOperation” event (after 600msec)</td>
<td>124</td>
</tr>
<tr>
<td>8</td>
<td>Process “New Order” event</td>
<td>125</td>
</tr>
<tr>
<td>9</td>
<td>insert “New Order” event (after 407msec)</td>
<td>126</td>
</tr>
<tr>
<td>10</td>
<td>Notify control</td>
<td>127</td>
</tr>
</tbody>
</table>

Table 1: *Synchronization events data between the emulation and a fast controller*
b. Slow controller, emulation speed factor = 1/1

![Diagram showing synchronization between emulation and a slow controller](image)

**Figure 5: Synchronization between the emulation and a slow controller**

<table>
<thead>
<tr>
<th>Event</th>
<th>Computer Time (msec)</th>
<th>Emulation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process “New Order” event</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Current time</td>
</tr>
<tr>
<td>2</td>
<td>insert “New Order” event (after 290msec)</td>
<td>111</td>
</tr>
<tr>
<td>3</td>
<td>Notify control</td>
<td>112</td>
</tr>
<tr>
<td>4</td>
<td>Process “New Order” event</td>
<td>250</td>
</tr>
<tr>
<td>5</td>
<td>insert “New Order” event (after 410msec)</td>
<td>251</td>
</tr>
<tr>
<td>6</td>
<td>Notify control</td>
<td>252</td>
</tr>
<tr>
<td>7</td>
<td>Control reports to emulation (do operation1 within 60msec)</td>
<td>500</td>
</tr>
<tr>
<td>8</td>
<td>Emulation receives “New Order” response (do operation1 within 60 msec)</td>
<td>501</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2: Synchronization events data between the emulation and a slow controller**
3.2.3 Offline visualisation

In order to visualize a simulation run there are two options: online visualization and offline visualisation. (Saint Germain, B., Valckenaers, P., Van Brussel, H., Hadeli, Bochmann, O., Zamfirescu, C., Verstraete, P., 2003)

Online visualisation means that while running the simulation there is some visualization. At the end of the simulation run you get typically an overview of the selected performance parameters. This type of visualization is available in most of the current simulation packages.

Offline visualization works in a totally different way. The simulation happens as fast as possible; this means that there is no waiting time between two different discrete events. While simulating as fast as possible the state of the factory is serialized into a file, a movie is made out of the states. This file enables the user of the simulation tool to see and evaluate the states.

Both systems have advantages and disadvantages; the following table gives a good overview.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Offline visualisation</th>
<th>Online visualisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding or changing performance</td>
<td>supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>measurements (same simulation run)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rewinding</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Jump to specific time</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Select level of detail</td>
<td>supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Direct interaction</td>
<td>Not supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

3.2.4 Future Developments

In order to add more flexibility into the discrete event calendar, the calendar can have some levels of detail. A first level would be typical the conceptual level, starting and stopping of the operations. A second level could emulate the operators needed for that operation. In this way you create an environment where you can specify the level of detail and speed up the simulation.

A special discrete event level is the visualization level. This level will update the state of the emulation frequently in order to get a smooth visualization.

4 Conclusion

This paper gives an overview how to make a simulation in a flexible way. This simulation is easy to validate. This flexibility is the result of two system properties:

- The separation from the emulation and control. Both subsystems are not exposed to specific knowledge of the other subsystem.
- The one to one mapping in the emulation with the reality. This decomposes the validation problem of the total emulation into smaller entities.

The second part of the paper describes a procedure in order to get accurate simulation results. The simulation runs as fast as possible without creating artificial situations. The controller get enough time to decide for the next actions. Secondly, the controller has no more time than it would be the case in reality. This is the result of the real time component that replaces the discrete event execution.

Within this simulation framework, a visualisation approach is proposed. This approach is a combination of offline and online visualisation.
5 References


