International Journal of Manufacturing Economics and Management
ISSN 2784-1278 ISSN-L 2784-1278, Vol. II, No. 1 / 2022
https://doi.org/10.54684/ijmem.2022.2.1.16

SIMULATION MODEL OF PROCESS SPORADIC CONTROL

Ivan M. Gostev[1], Pavel E. Golosov[2]

[1]"Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Moscow, 127051, Russian Federation
[2]The Russian Presidential Academy of National Economy and Public Administration (RANEPA), Moscow, 119571, Russian Federation

Corresponding author: Ivan M. Gostev, igostev@gmail.com

Abstract: The control solutions for modern production with a large number of processes performed simultaneously require equal or more computing power than, for example, calculating hashes for bitcoin. Hereinafter the reference is made to the problem of maximizing the production of the quantity of some mechanical units, which include several types of details required to assemble the unit in a certain quantity. The details are assumed to be manufactured and delivered to the assembly shop in a certain set. It is required to ensure the production of details on a specified number of machines for the assembly of maximum units for a fixed time. Any detail is expected to be produced on any machine. The same details can be produced in parallel. The optimization parameters will be provided by number of details of the required nomenclature, as produced with limited machine support used for the time-constraint production of these details. The production of a unit results from the available details of required nomenclature and its assembly from such details. We consider the production simulation model focused on maximizing the output of assemblies and increasing the efficiency of the use of available equipment. To solve the problem, we use the sporadic control mechanism for the number of manufactured details.

Key words: parallel algorithms, simulation modeling, scheduling, effectiveness of execution, deadline, sporadic control, SimEvent, Simulink.

1. INTRODUCTION

Currently, the optimization of processes of mass production of details and assembly of their finished units becomes a multi-criteria optimization challenge. This is due to the necessity to ensure a uniform workload of the machine park and to maximize the output of finished units for a certain period of time. The assembly of such a unit requires to have appropriate details and in the relevant quantity.
Supposing there are two shops — a machine shop and an assembly shop. The details are made in the machine shop and moved to the assembly shop. There are some machines used for the potential production of a detail, with such production time to depend on the type of a detail. It is necessary to maximize the number of units assembled in the assembly shop. This problem belongs to NP-complete task and is classified as a "backpack" problem. Classical solution of this problem requires construction of an objective function that should regulate (optimize) the output of the necessary number of certain details, in order to maximize the number of assembled units for a period of time. However, the application of this technique leads to multi-criteria optimization with parameter constraints and requires large computational resources, as well as cannot be considered optimal.
In this work, this problem is addressed by the sporadic control method. This method belongs to D.I. Ageikin, who used it to control the sensors at a nuclear power plant [1] back 1959. This method is in the decision to be made with respect to the primary production of a specific detail and the required quantity thereof on the basis of some numerical characteristic — priority, which is attributed to this detail. The surplus in details of this type decreases the priority, as well as their increase is supported by shortage. Therefore, the system itself dynamically controls the production of the desired number of details of each type. This work uses two types of sporadic control, which will be outlined below.
Note. Since assembly always takes the same period of time, we will consider it zero in this model.

2. PROBLEM STATEMENT

Let us suppose that there is some unit consisting of n type of details, each of which is needed for its assembly in k quantity. The problem is to ensure the production of the required number of details in some shop to assemble the maximum units in some time. In this case:

- it is required to ensure the proper number of types and details within this unit is produced;
- to ensure parallel production of the same details (there is a finite set of machines for their production);
- to minimize the time of manufacturing the sets of details to assemble the unit;
- it is assumed any detail can be produced on any machine, but the production time of different details is different;
- defective details may occur during the production of details. In this case, the detail must be re-manufactured;
- if a manufacturing process has been started for a detail, it must be terminated. It is not possible to interrupt production.

To address the challenge, we used simulation methods of cloud systems [2 - 4]. For this purpose, a mass service-based architecture [5] was built, the process of modelling whereof used different laws of occurring inputs and sporadic control mechanisms. Matlab/Simulink [6] with SimEvent package [7] was used as a toolkit to solve the problem.

3. GENERAL DESCRIPTION OF THE MODEL

We imagine the process of the detail manufacturing as a task to be solved on the server. General architecture of such system is shown on Figure1. The tasks (applications) are generated in the Generators block and come to Switch1. Then, they run to PriorityServer, where the priority of tasks on servers is assigned. This server sends the tasks to Entity_Queue, where the tasks wait for execution time. This queue is organized to preferentially flow tasks in order of priority. Further, the tasks pass through Switch2 to the server Subsystem1-Subsystem5, with each subsystem to contain 10 server blocks. If an execution error occurs in any server block, the task returns to the execution queue through connectors T1-T5.

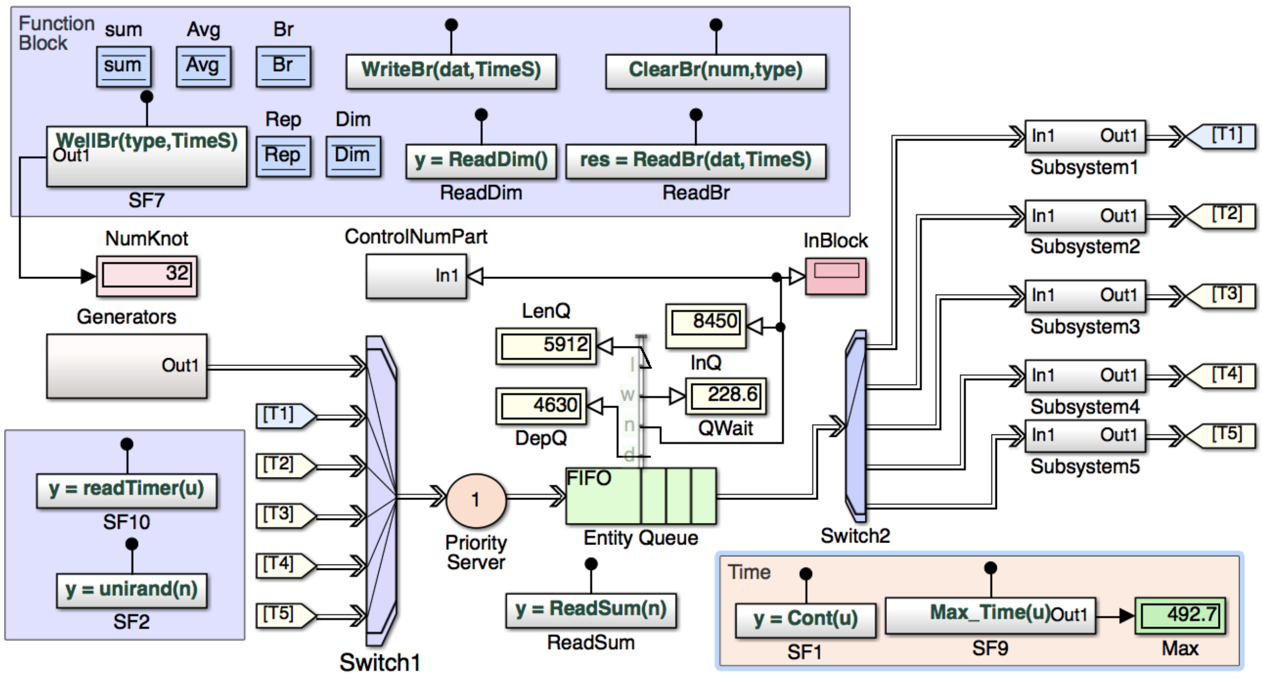

Fig. 1. General flow chart of the simulation model

ControlNumDetail block (Figure 2) is designed to control the number of requests entering the queue and consists of 3 comparators, which receive statistical data on the number of requests in the queue.
When the preset values (64, 128 and 256) are exceeded, the signals are fed to Switch1-Switch3 that generate a final signal to the input of DimPar() function. Depending on the value of this signal, the EntityGen1-EntityGen10 generators (Figure 3) assign a value of TypeNum parameter, which determines the number of requests for a detail of each type in the splitters Rep1-Rep10 (Figure 3).

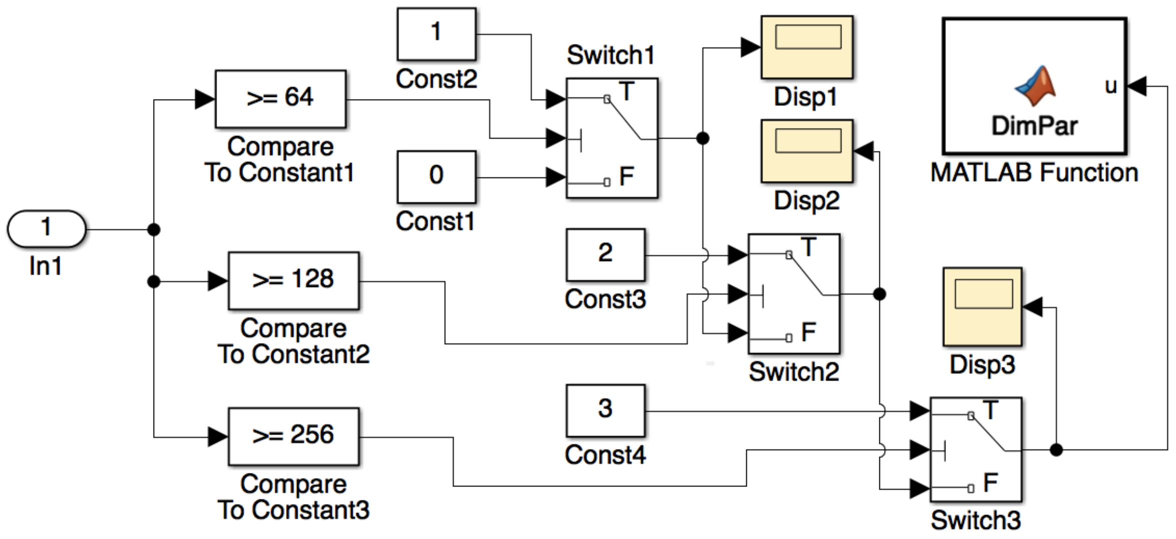

Fig. 2. Flow chart of ControlNumDetail module

4. BLOCK OF GENERATORS

The block is shown in Figure 3 and consists of 10 EntityGen1-EntityGen10 task generators, which corresponds to ten types of detail. Then, the tasks are transferred to Rep1-Rep10 splitters, in which the tasks are multiplicated under TypeNum control parameter. The tasks are further transferred to InputSwitch, from which they follow to Switch1 of the general chart (Figure 1).
Subsystem1-Subsystem10 blocks are used to set the time of task generation. In this case, 500 conventional units were used in the simulation time interval. Gen1-Gen10 display blocks show, how many tasks were generated during the simulation time interval. Function GenFun is used to set the task generation laws, and function StampEntity() — to determine the initial value of the starting task parameter in the system. The model used four types low of generation: with constant interval of each task; by exponential law; with equal probability law on the interval; and Poisson's law of distribution. The generation law is selected in function GenFun().

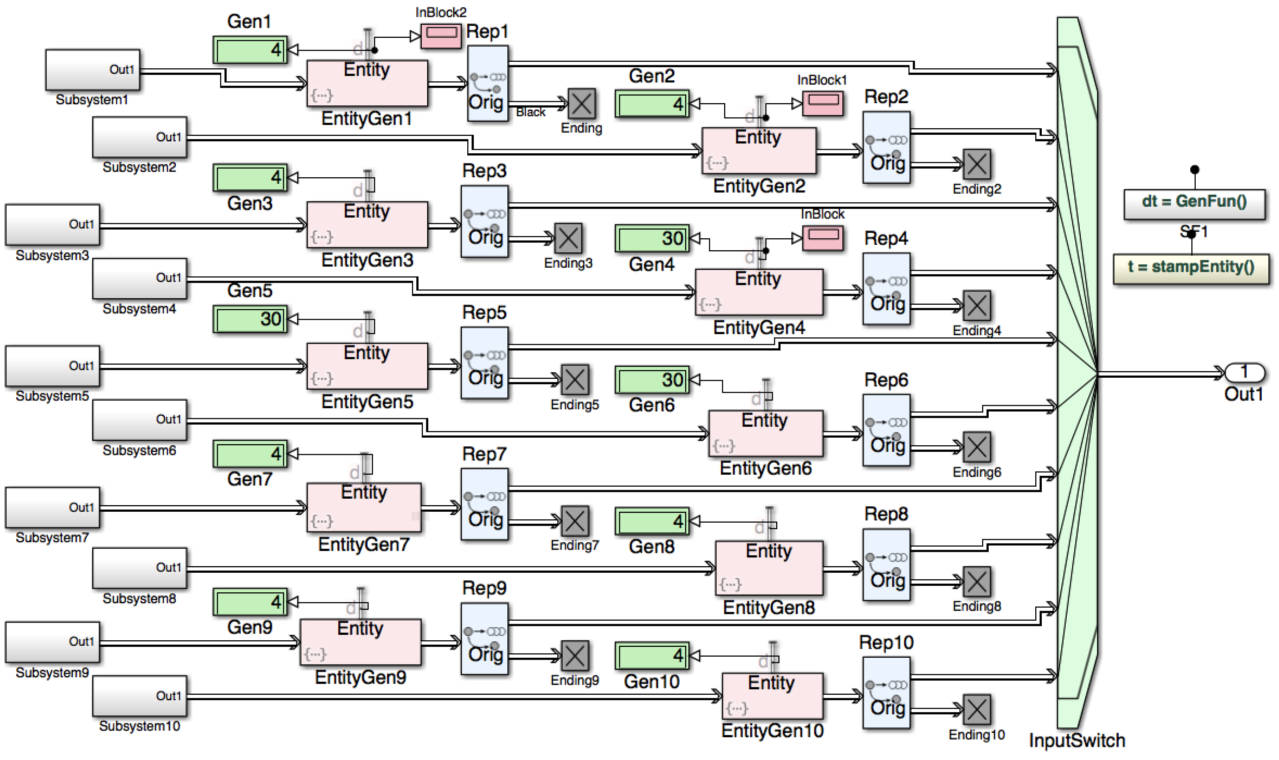

Fig. 3. Flow chart of Generators module

One of the Subsystem1-Subsystem10 blocks is shown in Figure 4. It consists of the following blocks. The SignalBuilder block is intended for arbitrary setting of the generation intervals of request. In this case, its lie in the range from 0 to 500 conventional time units.

The EntityGenerator block is used to define the law of entity generation, for example Poisson and define its parameters. The signal from this generator comes from the EntityGate switch, which is controlled by a signal from the MessageSend block. If this block receives a unit from the Product block, then the EntityGate switch passes the signal to the corresponding request generator EntityGen1-EntityGen10 in Figure 3. Otherwise, there is no output signal and this generator does not create requests for the manufacture of the detail. The use of such a mechanism makes it possible to generate request in any interval of the total simulation time and further optimize the efficiency of the entire system.

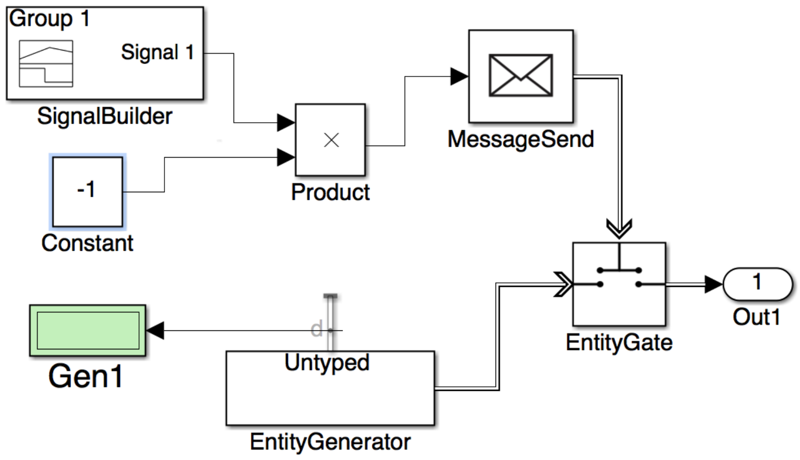

Fig. 4. Flow chart of one of Subsystem placed in Figure 3

5. SUBSYSTEM BLOCKS OF THE BASIC MODEL

The server nodes were combined into groups of 10 for the simplification of the overall architecture; therefore, in this project uses 50 servers united into 5 Subsystems (in the general chart of Figure 1). Each subsystem is shown in Figure 5. Adetail from servers, it has OutputSwitch input splitter distributing tasks among the servers and InputSwitch as a resulting switch.

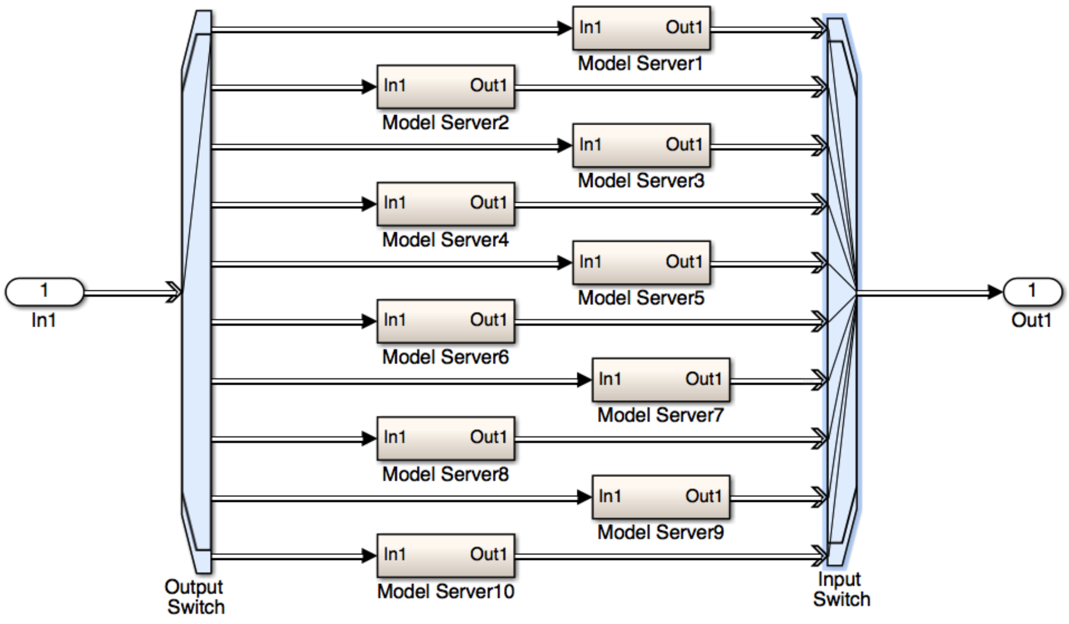

Fig. 5. Flow chart of Subsystem1-Subsystem5 modules

6. SERVER BLOCK

The server function blocks in Figure 6 containing the following components, are used to process the tasks flowing into the system. Two elements of Entity Server type. The first of which is the main server (Entity

Server). This server block uses a random number generator with a uniform distribution in the range [0,1] to simulate a defective detail. A value of 0.99 is chosen as the criterion for the rejected detail. When this value is exceeded, the Term task parameter is assigned with value 2 and it goes through Reject switch to the second server — Error Server, which is intended only for registration of server error condition and has zero processing time. During normal performance the task from the server is sent to the DelPass3 terminator and is destroyed, which means that the finished detail is sent to the assembly shop. DelPass3 terminator performs the following functions:

- terminates the existence of the task.
- calls WellBr function, which increases (in memory) the number of completed tasks (completed details) of this type by 1.
- enters parameters of the executed subtask into the global variable Br using WriteBr() function;

When the task arrives to the server block, function GateCtrl1(-1) is called, which closes the EntityGate input valve. When the task is finished, this function is called with parameter 1 to open this valve. All other elements of the server block — Display and Scope types — are auxiliary and intended to control the number of executed tasks and the course of computational processes.

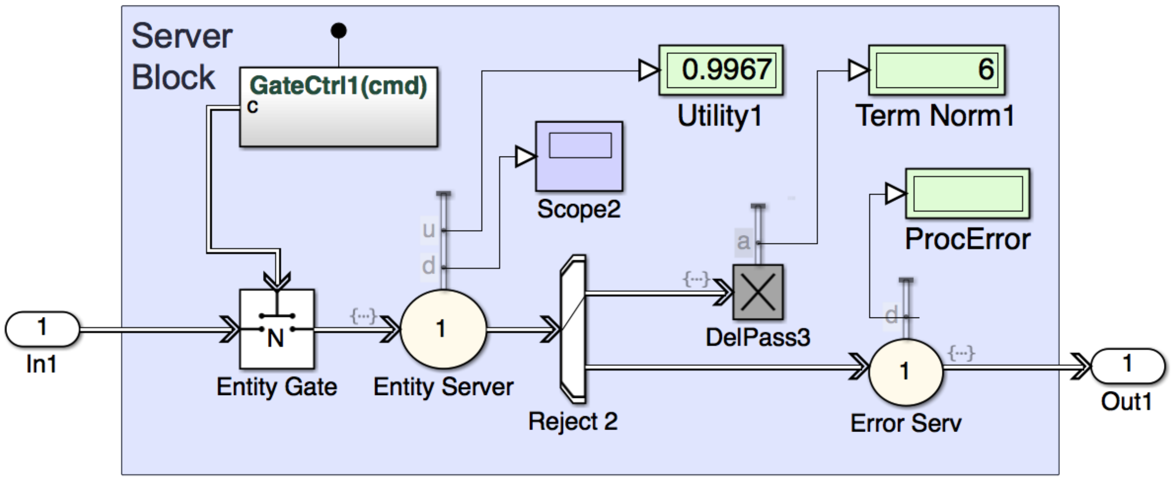

Fig. 6. Flow chart of the server module

7. TASK PERFORMANCE CONTROL MECHANISM

Several sporadic control mechanisms were used to solve the task. WellBr(type,TimeS) function was written to implement the first principle. This function has two parameters. The first one is the task type (detail), and the second one is the start time of the detail processing. For the purpose of system control, an array of Avg(10) global variables consisting of 10 elements is created. Each member of the array corresponds to the type of the detail, and its value corresponds to the current priority of the task that is queued in Entity_Queue. There is a BR array to account for finished details, which stores all the types of manufactured details (tasks) and their quantities. Since the queue has priority maintenance, the tasks with higher priority will go through the queue faster.

Since the main criterion for task performance is to assemble the maximum number of units, WellBr function has counters that record the number of manufactured detail of each type. As soon as the number of each detail type matches the number needed to assemble the unit, such number of details is subtracted from the corresponding counters, and the list of finished units is added with 1. The following algorithm is used to manage the priorities. Since the production time of each detail is different and the interval of the tasks to be processed can be determined by different statistical laws, the number of details produced will also be different. Some details will be produced in many quantities and some – less. When the finished set of details is subtracted from their counters, the value of the number of finished details of a type will be equal to zero at a timepoint. In this case, WellBr() program sets the priority value in the corresponding variable Avg to a few units less. (In Simulink, the maximum value of priorities corresponds to 0). Therefore, the more the priority value increases for a specific detail, the faster it moves in the queue and executes. However, to prevent negative priority values, the priority must be decreased in accordance with the increasing number of finished details. As soon as the number of some details becomes sufficient to assemble the unit, the priority of execution of those details decreases.

For the maintenance of proper system operation, each counter value change involves the removal of the number

of tasks (details) required to assemble the node from the BR array.
The second mechanism of sporadic control is based on controlling the number of detail requests flowing into the queue. In this case, the control element is the number of detail requests in the queue. ControlNumDetail block is used as the control element. The increase in the number of queue orders leads to the decremental decrease in the number of orders, which is regulated by TypeNum in the task generators. According to the value of this parameter, Rep1-Rep10 splitters generate the desired number of requests. As the number of tasks in the Entity_Queue increases, the number of tasks at the output of the splitters decreases, and vice versa, as the number of tasks in the queue decreases, the number of tasks at the outputs of Rep1-Rep10 splitters increases.

8. SIMULATION RESULTS

10 task generators (detail types) were used as an example. Assuming that some details are required in a single quantity to assemble the unit (e.g., the body of the unit) and some details — in large quantities (e.g., bolts and nuts). Accordingly, the labor intensity of manufacturing details will be different. Therefore, the application rates for more labor-intensive details (and therefore fewer details) should not exceed that of the fewer labor-intensive details. In this model, the following durations for the production of details have been assumed: 40,10, 8, 2, 2, 2, 20, 20,12, 12. The priority of all details was selected to be equal to some average value — 30. The results of modeling at two variants of the generation law at constant intervals of generating orders (curves 1) and with enabled sporadic control mechanism (curves 2) are presented in Figure 7.

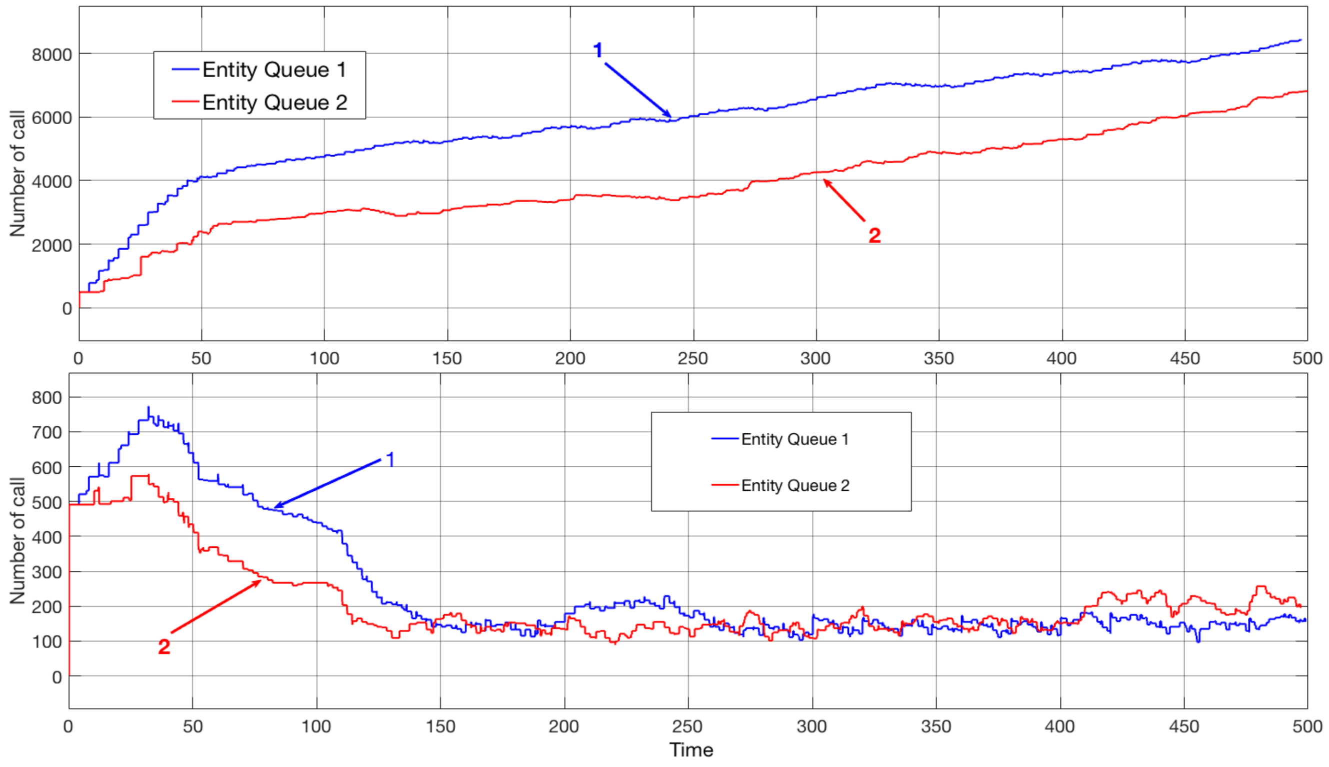

Fig. 7. Behavior of Entity_Queue (number of reqtests of detail all types). The above chart is without the sporadic control mechanism, the lower chart — with the control turned on. Curves 1 correspond to the fixed interval of requests generation. Curves 2 — the requests are flowing in exponentially low

As a result of the simulation, we have the following results. The number of assembled units at a fixed interval of applications and discontinued control in the time interval of 500 conventional units is equal to 32 pcs. And when the control is on, it is equal to 55. Accordingly, with the exponential law of succession of applications we have: when the control is off — 46, when the control is on — 62.
Thus, the use of sporadic control allows you to improve the production capacity by about 30%. From the behavioral analysis of Entity_Queue (Figure 6) we see it grows very fast in the absence of sporadic control. Consequently, long lead-time applications queue with short lead-time ones, and this significantly increases the queue. At the same time, some details are missing to complete the units and as a consequence, the number of assembled units is much lower than when the control is off.
With control on, it is easy to see how sporadic control adjusts the size of the queue. Initially, there is a large inflow of requests being further decreased through their adjustment and change in the queue priority and

maintained at some constant level regardless of the law of requests.

9. CONCLUSIONS

The use of sporadic control to regulate manufacturing processes can significantly improve the production efficiency. In this case, control mechanisms are greatly simplified due to no necessity to generate and, thereafter, optimize a target function [8]. Attribution of parameters to the tasks not only optimizes the output, but also regulates the number of finished details that were not used in the assembly of units.
In addition, this chart allows to easily rearranging production where it is necessary to change some parameters of details or assembled units.
Notwithstanding that the provided chart is somehow simplified, it can be modified and adjusted to the real production. Moreover, such a control chart was originally developed for task management in a cloud system and has proven its high efficiency [9-11]. In addition, such management system can be applied with minor modifications to logistics, mining and other management processes.

10. REFERENCES

1. Ageykin D.I., Kostina E.I., Kuznetsova N.N., (1959). *Sensors of automatic control and adjustment system*, MASHGIS, Moscow. 579 p. (in Russian).
2. Erl T., Puttini R., Mahmood Z. (2013). *Cloud Computing: Concepts, Technology & Architecture*. Pearson, 528 p.
3. Sudheer M.S., Reddy K.G., Sree R.K., Raju V.P. (2017). *An effective analysis on various scheduling algorithms in cloud computing*. Proc. of the International Conference on Inventive Computing and Informatics (ICICI), pp. 931–936.
4. Tomar A., Pant B., Tripathi V., Verma K.K., Mishra S. (2022). *Improving QoS of cloudlet scheduling via effective detailicle swarm model*. Lecture Notes in Electrical Engineering, vol. **768**, pp. 137–150.
5. Brandberg C., Di Natale M. (2018). *A SimEvents model for the analysis of scheduling and memory access delays in multicores*. Proc. of the 13th International Symposium on Industrial Embedded Systems (SIES), 2018, pp. 8442094.
6. MathWorks. (2016). *Simulink® User's Guide*. MathWorks®, Release R2016a, Natick, MA.
7. MathWorks. (2016). *SimEvents®, User's Guide*. MathWorks®, Release R2016a, Natick, MA.
8. Golosov P.E., Kozlov M.V., Malashenko Yu.E., Nazarova I.A., Ronzhin A.F. (2012). *Analysis of computer job control under uncertainty*. Journal of Computer and Systems Sciences International, vol. **51**(1), 49–64.
9. Golosov P.E., Gostev I.M. (2021). *About one cloud computing simulation model.* Proceedings of Systems of Signals Generating and Processing in the Field of on Board Communications, pp. 9416100.
10.Golosov P.E., Gostev I.M. (2021). *Simulation of servers with interrupts in large multiprocessor systems*. Journal of Instrument Engineering, **64**(11), 879–886. (in Russian).
11.Golosov P.E., Gostev I.M. (2022). *Cloud computing simulation model with a sporadic mechanism of parallel problem solving control*. Scientific and Technical Journal of Information Technologies, Mechanics and Optics **22**(2), 269–278. (in Russian).

Received: February 23, 2022 / Accepted: June 15, 2022 / Paper available online: June 20, 2022 © International Journal of Manufacturing Economics and Management