Jurnal Teknik Industri, Vol. 20, No. 2, December 2018: 113-126
ISSN 1411-2485 print / ISSN 2087-7439 online
DOI: 10.9744/jti.20.2.113-126

Scheduling Flexible Manufacturing System with Stacker Crane Using Coloured Petri Nets

Ari Setiawan[1]*, Teguh Ersada Natail Sitepu[1]

Abstract: Flexible Manufacturing System (FMS) is a system that can increase production speed and accuracy significantly. FMS can process the various product at the same workstation. However, FMS needs an efficient allocation of resources as inputs (e.g. job schedule and material handling allocation). This paper presented a modelling of FMS production scheduling problem. The model consisted of fifteen workpieces, four machines, and a stacker crane. Coloured Petri Nets (CPN) was the programming language which used to simulate the model. The model had three modules; they were the machine, loading/unloading, and delivery module. Each module had a set of submodules. Machining process, pick-up request, and picking mechanism submodules were in the machine module, while job selection, job picking, and machine selection submodules were in the loading/unloading module. Additionally, delivery to pallet stacker, and proceed to stage two submodules were in the delivery module. The simulation executed 436 steps with 1.467 second computational time. The makespan was 1.647 minutes, and all machines had high utilization level, higher than 80%. However, the stacker crane utilization level was low.

Keywords: Scheduling, Flexible Manufacturing System, Coloured Petri Nets, stacker crane, simulation.

Introduction

The customer has very diverse demands in Industry 4.0 era and manufacturer is expected to have high production flexibility to satisfy those demands. Flexible Manufacturing System (FMS) is a manufacturing system with high production flexibility (Shivanand *et al.* [1]; Majija *et al.* [2]; Bohn and Jaikumar [3]). FMS can produce various products at the same workstation. FMS also can adjust production volume based on demand. FMS needs adequate allocated resources as inputs to achieve these abilities. Those resources consist of CNC machine, fixture, tool magazines, tool, automated material handling, and buffer. Based on Setiawan *et al.* [4], all those resources need maintenance schedule, to minimize any interruption in the production processes.

FMS also needs production scheduling to make production goes well (Sule [5]; Zhan *et al.* [6]; Sahraeian [7]). There are various methods in production scheduling, categorized by the exact method and approximation method. Different production scheduling methods can produce different makespan (the time difference between the start and finish of a sequence of jobs or tasks) and flow time (the time taken for completion of a flow of material) Setiawan *et al.* [8]. Setiawan *et al.* [9] have developed a mathematical model for FMS production scheduling considering cutting tools.

[1] Faculty of Industrial Technology, Department of Industrial Engineering, Institut Teknologi Harapan Bangsa, Jl. Dipati Ukur 80-84, Lebakgede, Coblong, Bandung, Jawa Barat, Indonesia 40132
Email: teguh_sitepu@ithb.ac.id

* Corresponding author

Since the model is hard to solve using an exact method, many researchers have been applied various approximation method to this model.

Pakpahan *et al.* [10] have developed an algorithm to solve the Setiawan *et al.* [9] model based on the ant colony optimization method. However, this algorithm used a static scheduling approach. This condition leads to the possibility that there will be unfinished jobs because the cutting tools are unavailable. Sitepu *et al.* [11] suggested calculating the cutting tools before a manufacturer decided to start any production processes. Setiawan *et al.* [12] developed dynamic scheduling to anticipate broken cutting tools during unscrewed operation. Furthermore, Setiawan *et al.* [13] developed a job rescheduling model for FMS which minimize makespan and minimize starting time difference between initial and new schedule. The model expected to give better scheduling and performance.

FMS production scheduling and performance were simulated using Pharo 3.0 programming language by Setiawan *et al.* [14] while the subject considers various cutting tools. Pharo needs pre-defined classes as input to create an FMS model. These inputs make the FMS model difficult to configure. Therefore, Petri Nets (PNs) is used as mathematical modelling language in the modelling, analysing, simulating, and controlling the manufacturing system. PNs is also useful to model systems whose behaviour can be described as interferences between asynchronous and concurrent processes (Gradisar and Music [15]; Pan [16]; Yasuda [17]). Gradisar and Music [15] modelled a multiproduct batch plant

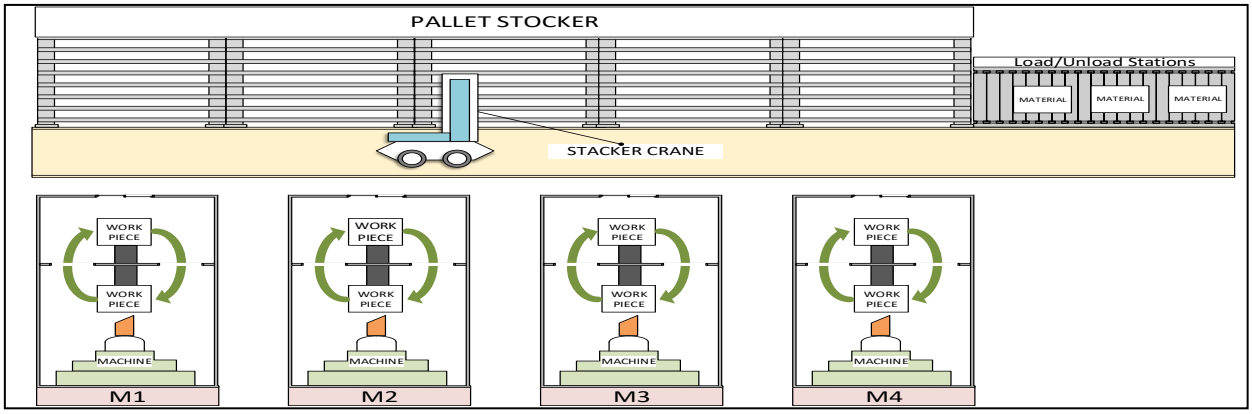

Figure 1. FMS construction

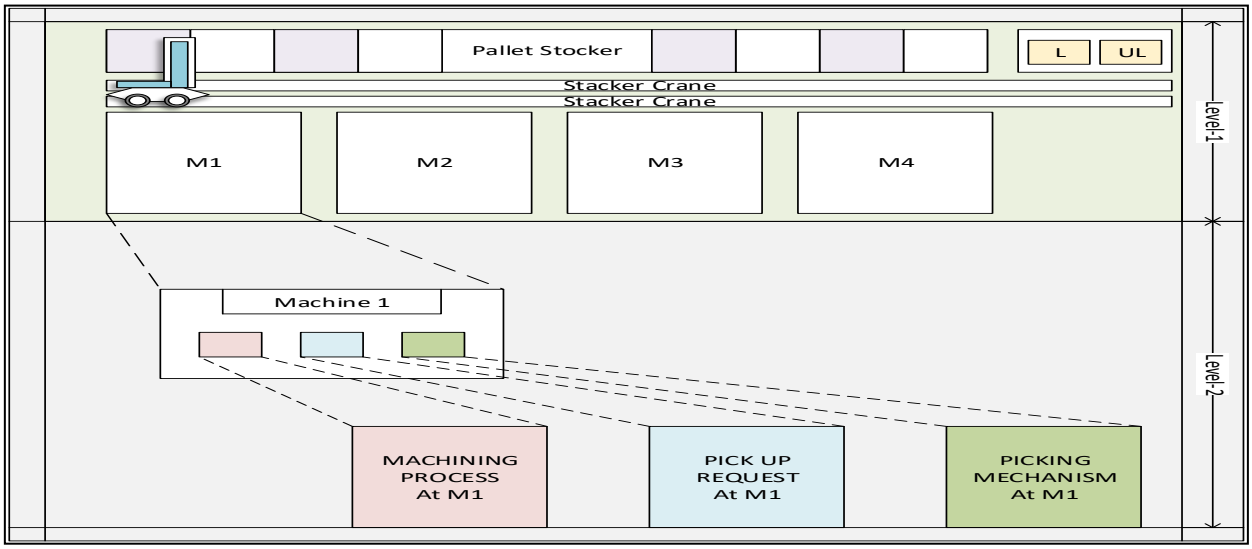

Figure 2. The CPN hierarchy for a *machine* module

using PNs. Pan [16] developed a computationally more efficient optimal deadlock control policy of FMS using PNs. The experimental results indicated that it is the most efficient policy among all known ones that can design optimal controllers. Yasuda [17] extended PNs for hierarchical and distributed control of large and complex robotic manufacturing system.

However, there is a backward compatible extension of PN called Coloured Petri Nets (CPN). CPN is a discrete-event modelling language that combines the capabilities of PNs with the capabilities of a functional programming language (Jensen *et al.* [18]; Igei *et al.* [19]). The main difference between PNs and CPN is that the CPN is used as a simulation tool without the necessity of a new extension definition (Rocha de Carvalho and Porto [20]). Jensen *et al.* [18] simulated a multi-product production system to verify the use of CPN. The system consists of twenty types of products with their operational sequences. The simulation result shows that the takt-time value converges stably from 7 to 8 seconds per product unit. Long *et al.* [21] simulated a production system in Industry 4.0 using several non-PNs methods (e.g. MM and UML) and various high-level modelling methods (e.g. CPN). The result shows that CPN still have huge gaps in flexibility and adaptability of a production system.

Therefore, in this paper FMS production scheduling is simulated using CPN. The proposes system considers the stacker crane which was not considered in the Setiawan *et al.* [14]. The objective of this paper is to measure the system performance, which includes the machine and stacker crane utilization level.

Methods

Problem Description

An Indonesian aircraft industry had been using FMS since 1992 (Setiawan *et al.* [14]). The construction of FMS in this company consisted of four

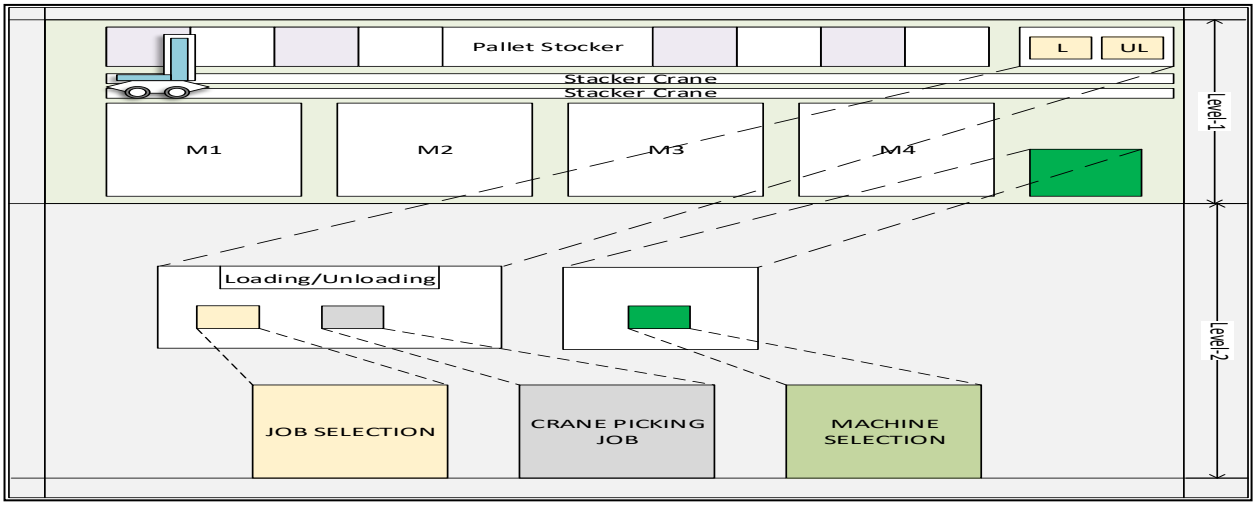

Figure 3. The CPN hierarchy for the loading/ unloading module

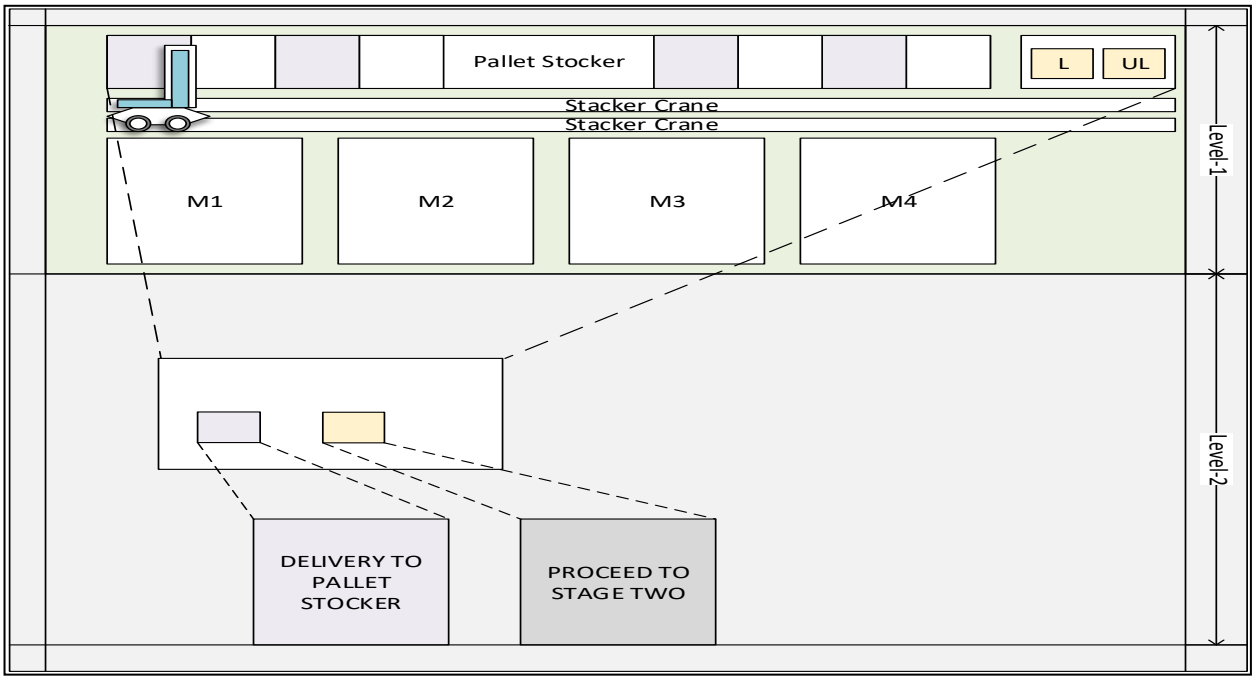

Figure 4. The CPN hierarchy for the delivery module

identical machines (M1, M2, M3, and M4), a pallet stocker, a stacker crane, and a loading/unloading station as shown in Figure 1. The workpieces in this case study were independent jobs which were processed in one or two stages. The production system would begin when there was a workpiece waiting in a loading/unloading station, which had unlimited capacity. The Automated Storage and Retrieval System (AS/RS) would check the machine availability. If there were no machine available, then the AS/RS would order a stacker crane to wait in the loading/unloading station until a machine was available. However, if there were more than one machines available, the system would choose a machine randomly.

After a machine had been chosen, the system would check the list of the workpieces waited in the loading/ unloading station. If there were more than one workpieces at the loading/unloading station, then it would be sorted based on the Shortest Processing Time (SPT) method. The workpiece with the shortest processing time would become the top priority. A stacker crane, which only could pick up one workpiece at a time, would be ordered to pick up the priority workpiece from the loading/unloading station and to drop it off to the chosen machine. The stacker crane movement assumed to be one minute. Each machine could contain two workpieces, one in the process slot and the other in the buffer slot. If a workpiece in the process slot has been processed,

then the Automatic Pallet Changer (APC) would automatically rotate it with a workpiece in the buffer slot. The APC rotation time was negligible. Each machine had a cutting tool to process the workpiece. This paper assumed the cutting tools had an unlimited lifetime.

The top priority of a stacker crane movement was the finished workpiece. The stacker crane would pick the finished workpiece which had two stages process to a loading/unloading station for a setup process. The setup process time was also negligible. Otherwise, it delivered a finished workpiece with one stage process to a pallet stocker, as the final storage place. This process was repeated until no workpiece waited in the loading/unloading station.

Hierarchy Module Design

CPN did not only focused on modelling a specific class, but also broad classes of systems; i.e. concurrent systems. The CPN constructed the model into a set of modules. The module's concepts based on the hierarchical structuring mechanism, allowing a module to have a set of submodules (Jensen *et al.* [18]). In this paper, we proposed three modules: machine, loading/ unloading, and delivery.

The Hierarchy of a Machine Module

A machine module described the state and the events for the procedure in all machines. There were three submodules in the machine module: machineing process, pick-up request, and picking mechanism. The machining process submodule was used to model the procedure of a workpiece operation which is delivered to a machine. The pick-up request submodule was work as a sensor which sent a pick-up request of a finished workpiece from a machine. The picking mechanism submodule was used to model the respond of a stacker crane to pick-up a request from a machine. Figure 2 shows an example of the CPN hierarchy for Machine 1 (M1).

The Hierarchy of Loading/Unloading Module

The Loading/unloading module described the state and the events for the procedure in a loading/ unloading station. There were three submodules in the loading/unloading module: job selection, picking job, and machine selection. The job selection submodule was used to model the procedure of a workpiece selection in a loading/unloading station. The picking job submodule was used to model the procedure of a workpiece pick-up process in a loading/unloading station. The machine selection submodule was used to model the procedure of a machine selection which will receive a workpiece. Figure 3 shows the CPN hierarchy for the loading/unloading module.

The Hierarchy of Delivery Module

The delivery module described the state and the events for a delivery procedure. There were two submodules in the delivery module: delivery to pallet stocker and proceeded to stage two. The delivery to pallet stocker submodule was used to model the procedure of a workpiece delivery from a machine to a pallet stocker. The proceed to stage two submodule, only applied for a workpiece with stage two. This submodule was used to model the procedure of a workpiece delivery from a machine to a loading/ unloading station. Figure 4 shows the CPN hierarchy for the delivery module.

CPN Submodule Design

The Machining Process Submodule

The machining process submodule had twelve places and three transitions for each machine. The places in the machining process submodule were Buffer_1, Inbound_M1, M1_Buffer, M1_Ready_To_Proc, SP1, Available_Mach, Processed_M1, Processed _ Job _ at _M1, M1_Waiting_For_Pickup, C1, M1_Not_Ready, and M1_Signaling. Buffer_1 was an output from the picking job submodule. Buffer_1 stored a token to identify a workpiece in the buffer slot of a machine. Inbound_M1 was used to record the arrival time of a workpiece in a machine. M1_Buffer stored a token (b) which represents the availability of a machine to process a workpiece. M1_Ready_To_Proc notified whether a workpiece was ready to be processed. SP1 was a place to store a job stage of operation data. Available_Mach was an output place to store a token. It was used as input for a machine selection submodule. Processed_M1 was an output place to record a workpiece finished processing in a machine. Processed_Job_at_M1 showed a workpiece that finished processing. M1_Waiting_For_Pickup was a place for a workpiece that has been rotated to buffer and to wait to be picked up by a stacker crane. C1 was a place to record the total time at a machine. One token r1 was in the place M1_Not_Ready shows M1 did not have any finished workpiece to be picked up by a stacker crane. M1_Signaling tells the stacker crane that there was a finished workpiece waiting to be picked up at a machine. Figure 5 shows the machining process submodule.

The transitions in the machining process submodule were M1_Setup_Job, M1_Processing_Job, and A. M1_Setup_Job transition was enabled when there were one token in Buffer_1 and one token in M1_Buffer. M1_Setup_Job would consume those tokens and produce three tokens: at Inbound_M1, at SP1, and M1_Ready_To_Proc. M1_Processing_Job transition was enabled where there were one token in SP1 and one token in M1_Ready_To_Proc. M1_

Processing_Job would consume those tokens and produce four tokens: at Available_Mach, at Processed_M1, at C1, and Processed_Job_at_M1. A transition was enabled when there was one token in Processed _ Job _at_M1 and 1one token in M1_Not_ Ready. A transition would consume those tokens and produce two tokens: at M1_Waiting_For_Picking and M1_ Signalling.

The Pick-up Request Submodule

The pick-up request submodule had eight places and eight transitions. Places in the pick-up request submodule were M1_Signalling, Req_Idle, Waiting_ to_be_Processed, Waiting_for_Req_Fulfillment, and Sending_Pickup_Req_to_C. M1_Signalling was input from the machining process submodule. One token (r1) in the M1_Signalling place informed there was

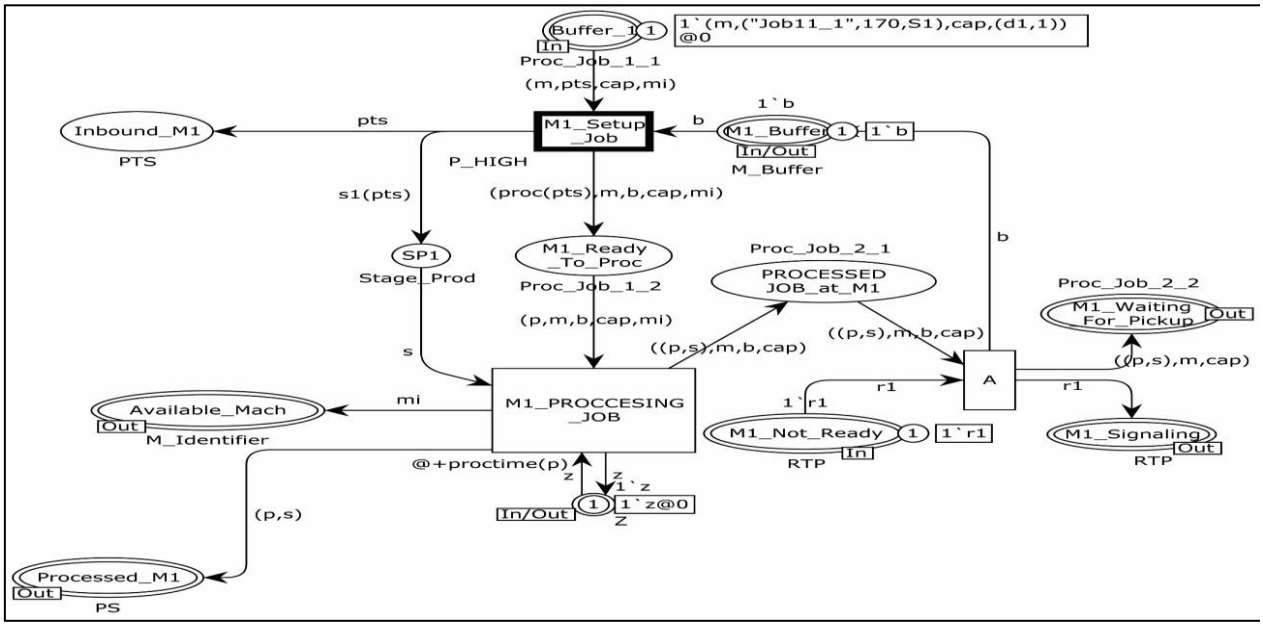

Figure 5. The machining process submodule

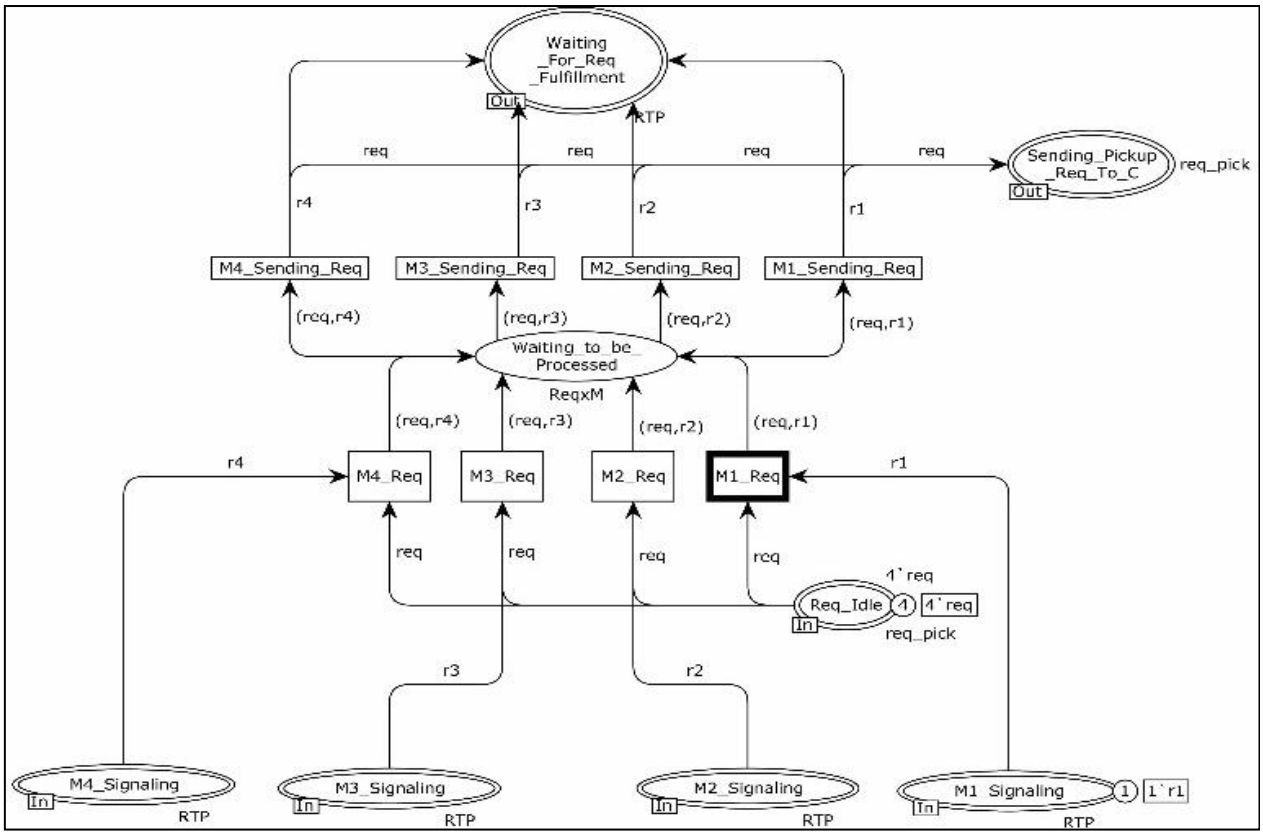

Figure 6. Pick-up request submodule

one finished processed workpiece in M1 and needed to be picked up by a stacker crane. Req_Idle was input from the picking job submodule. Four tokens (req) were in the Req_Idle place. In each machine, only one finished workpiece can be picked up. Waiting_to_ be_ Processed was a place to store all request tokens for every machine. Waiting_for_Req_ Fulfillment identifies which machine had sent a pick-up request and waited for a stacker crane. Sending_ Pickup_ Req_ To _C was used to call a stacker crane for a pick-up process. Figure 6 shows the pick-up request submodule.

The transitions in the pick-up request submodule were M1_Req and M1_Sending_Req. M1_Req transition was enabled when there were one token in M1 and minimum one token (req) in the Req_Idle. This transition would produce one token in the Waiting_to_be_Processed place. M1_Sending_Req transition was enabled when there was one token in the Waiting_to_be_Processed place. This transition would produce one token in the Waiting_ for_ Req_ Fulfilled place.

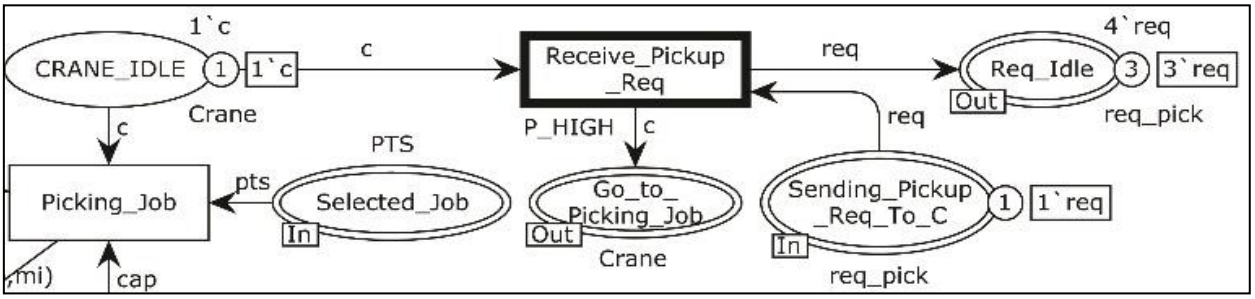

Figure 7. The receive pick-up request

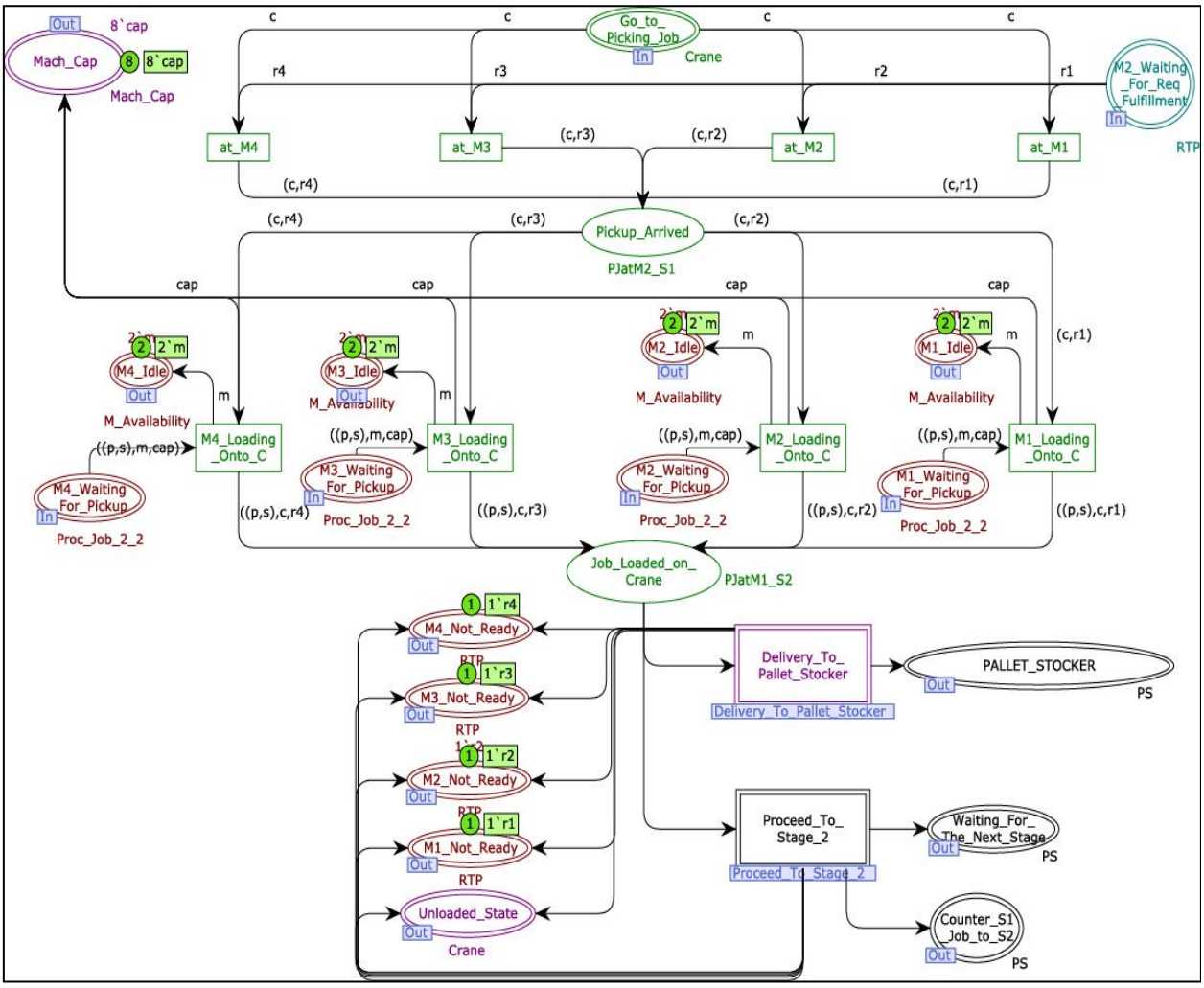

Figure 8. Picking mechanism submodule

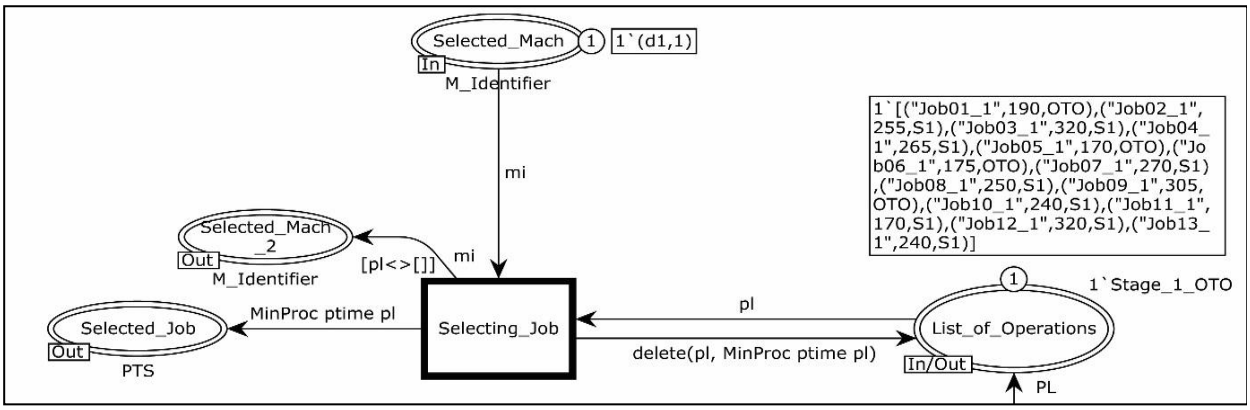

Figure 9. Job selection submodule

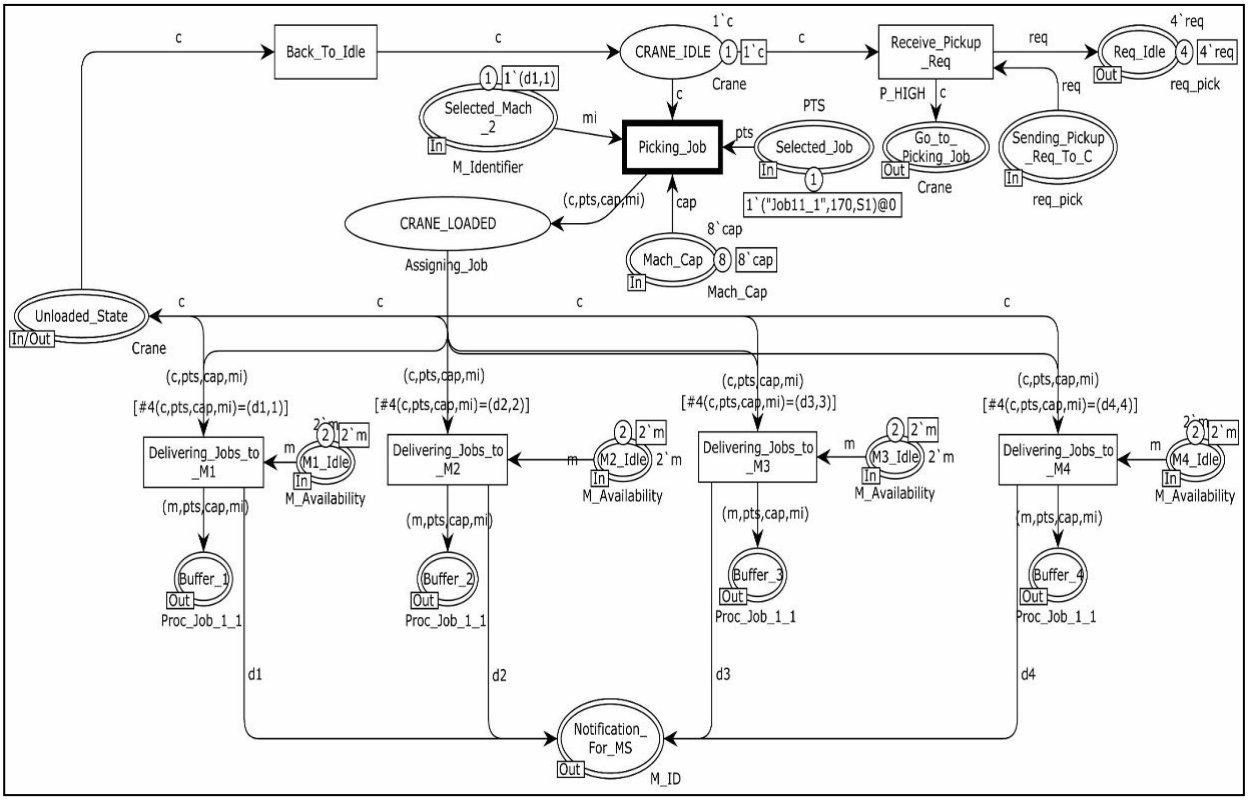

Figure 10. Picking job submodule

The receive pick-up request process was a part of the pick-up request submodule. The Receive pick-up request was only executed when there was minimum one machine sends a pick-up request. Receive pick-up had four places and one transition. Places in the receive pick-up request were Crane_Idle, Sending_ Pickup_Req_To_C, Go_To_Picking_Job, and Req_ Idle. Crane_Idle was a place to store one idle token (c) of the crane. One token (c) in Crane_Idle place informed the crane is idle and ready to do a pick-up or a delivery process. Sending_Pickup_Req_ To_C was an output from the pick-up request submodule. One token (req) in Sending_Pickup_ Req_To_C informed there was a pick-up request in the system. Go_To_ Picking_Job was an output. It stored command and sent it to the stacker crane. Req_Idle was a place to store all idle pick-up requests because the stacker crane is busy. Three tokens (req) in Req_Idle informed there were three idle pick-up requests. Figure 7 showed the receive pick-up request process.

The transition in the receive pick-up request was only Receive_Pickup_Req. This transition was enabled when there were one token (c) in Crane_Idle place and one token (req) in Sending_Pickup_Req_ To_C. Receive_Pickup_Req would consume those tokens and produce two tokens: at Go_To_Picking_ Job and Req_Idle. P_HIGH ensured this process would be the top priority.

Picking Mechanism Submodule

The picking mechanism submodule had twenty-one places, and ten transitions. Places in the picking mechanism submodule were M1_Not_Ready, M1_ Waiting_For_Pickup, M1_Idle, Unloaded_State, Pallet_ Stocker, Waiting_For_The_Next_Stage, Counter_S 1_Job_To_S2,Job_Loaded_On_Crane, Pickup_ Arrived, Mach_Cap, Go_To_Picking_Job, and Waiting_For_ Req_Fulfillment. M1_Not_Ready was a place to store token (r1). It described M1 was not ready to send a pick-up request. M1_ Waiting_ For_Pickup described a workpiece was waiting in a buffer slot of M1. M1_Idle described the slot availability in M1. Two tokens (m) in the M1_Idle place indicated that there were two available slots in M1. Unloaded_State states that a stacker crane was empty. Pallet_Stocker was a place to store a finished workpiece. Waiting_For_The_Next_Stage was used to store a workpiece which had a stage-two operation. Counter_S1_Job_To_S2 was used to record which job had finish stage-two operation. Job_Loaded_On_ Crane described a workpiece had been loaded on a crane. Pickup_Arrived stated a stacker crane had arrived in a machine. Mach_Cap was a place to store the capacity of an available slot for all machines. Eight tokens (cap) showed all machine slots were empty. Go_To_Picking_Job stated that the crane was going to a machine. Waiting_For_Req_Fulfillment was a place to store which machine is targeted by a stacker crane. Figure 8 shows the picking mechanism submodule.

The transitions in picking mechanism submodule were at_M1, M1_Loading_Onto_C, Delivery_To_ Pallet_Stocker, and Proceed_To_Stage_2. The at_M1 transition was enabled when there was one token

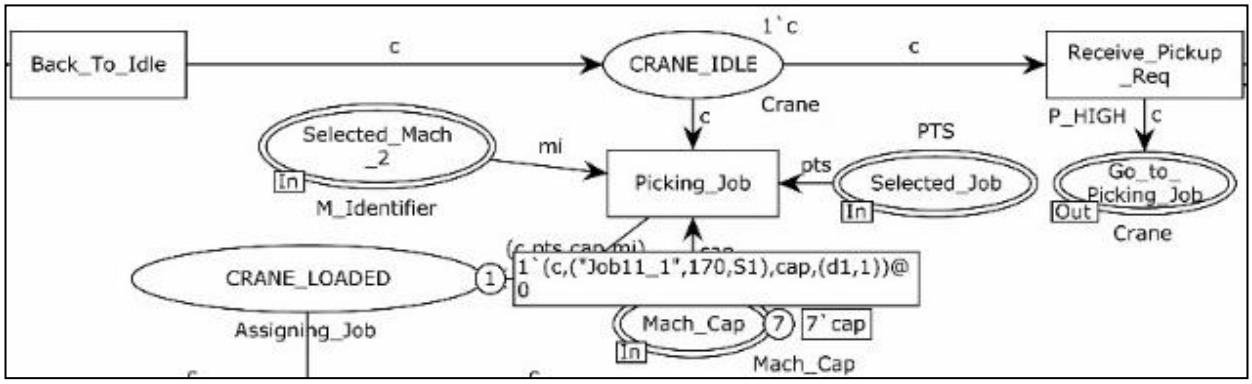

Figure 11. Picking job submodule (enabled)

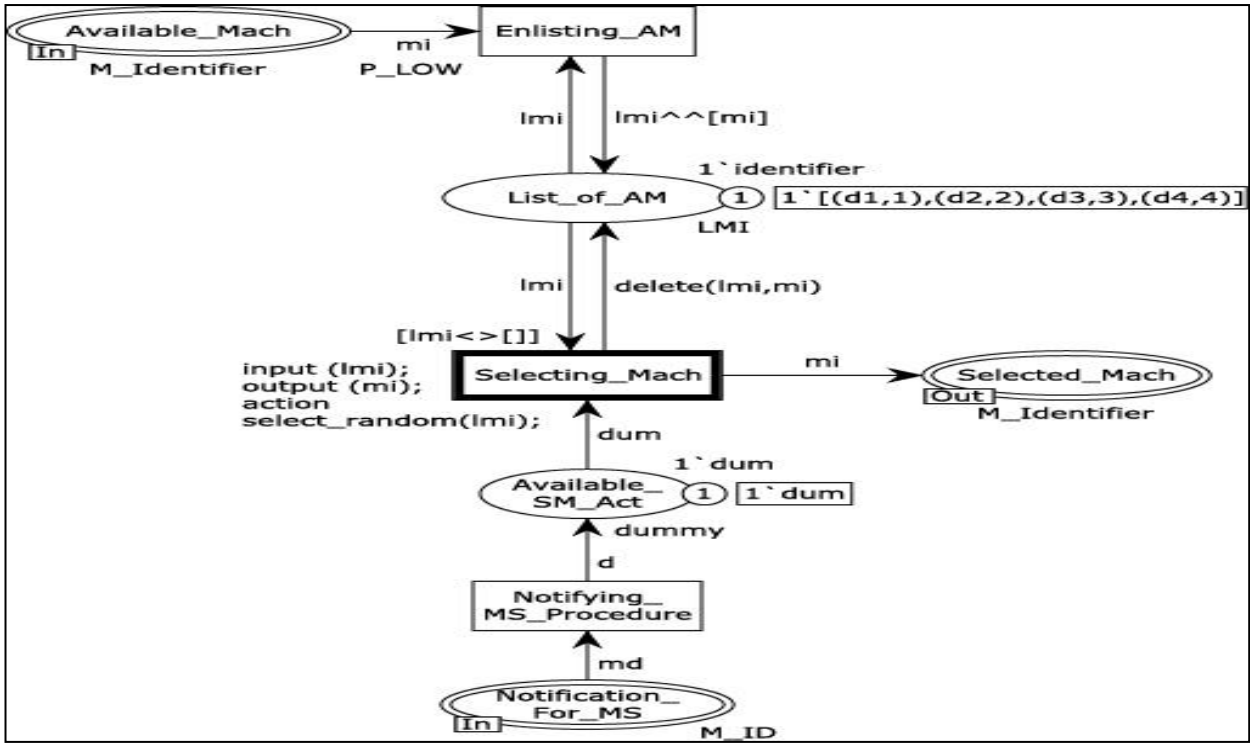

Figure 12. Machine selection submodule

(r1) in Waiting_For_Req_Fulfillment. This transition would produce one token in the Pickup_Arrived. The m1_loading_onto_c transition was enabled when there were one token in Pickup_Arrived and one token in M1_Waiting_For_Pickup. This transition would produce a token in the Job_Loaded_on_Crane and reduce a token in the M1_Idle. Delivery_To_Pallet_Stocker transition was enabled when there was one token in Job_Loaded_on_Crane has already had been processed. This transition would produce a token in the Pallet_Stocker place. The proceed_to_stage_2 transition was enabled when there was one token in Job_Loaded_on_Crane that had a stage-two operation. This transition would produce one token in Waiting_For_The_Next_Stage and one token in Counter_S1_Job_To_S2.

Job Selection Submodule

The job selection submodule had four places and one transition. Places in the job selection submodule were List_of_Operations, Selected_Mach, Selected_Mach_ 2, and Selected_Job. List_of_Operations was a place to store the list of workpieces which would be sent to a machine. The list was described in the Stage_1_ OTO. A token ("Job01_1",190, OTO) in Stage_1_OTO informed a workpiece number one with operation time 190 minutes and only had one stage. Another example, a token ("Job02_1",255, S1) informed a workpiece number two with operation time 255 minutes and had two stages. Selected_Mach was output from the machine selection submodule. One token (d1,1) informed machine one

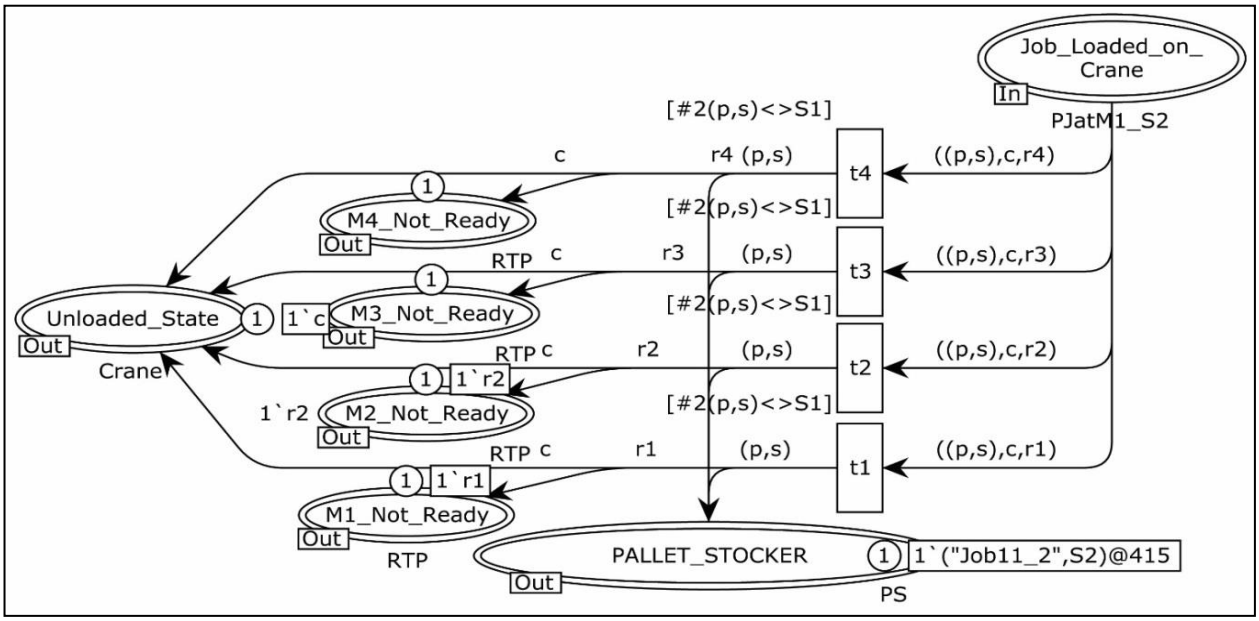

Figure 13. Delivery to pallet stocker submodule (enabled)

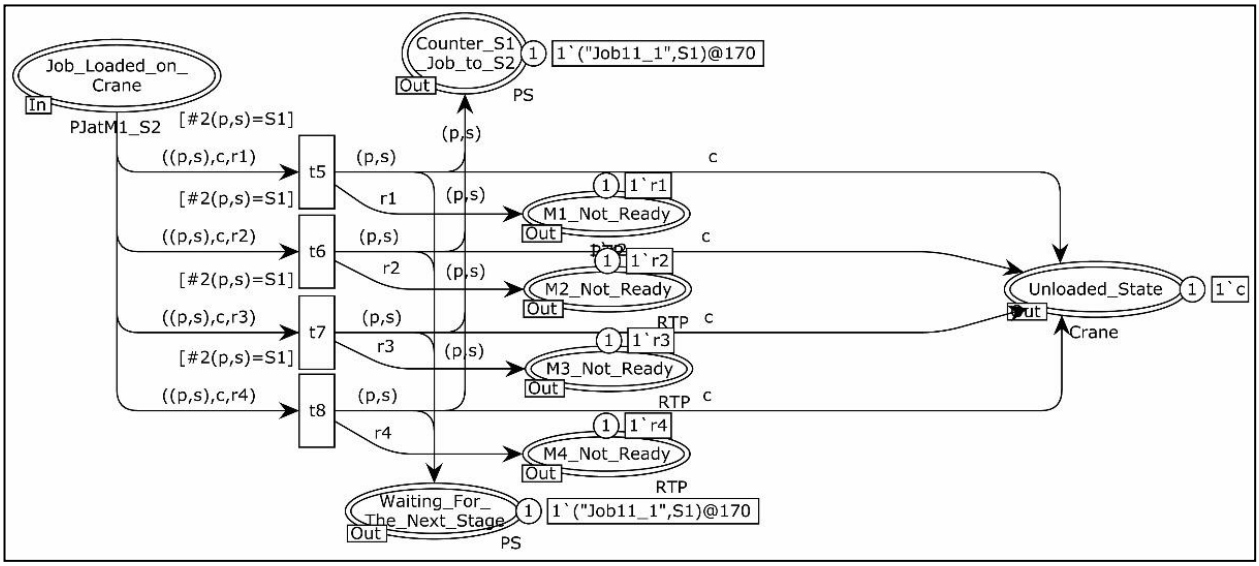

Figure 14. Proceed to stage two submodule (enabled)

was selected. Selected_Mach_2 was a place to store data of the selected machine and ready to be used to select a workpiece. Selected_Job was a place to store data of the selected workpiece which was delivered to a machine. Figure 9 shows job selection submodule.

The transition in the job selection submodule was Selecting_Job. This transition was enabled when there was one token (identifier) in Selected_Mach and one token Stage_1_OTO in List_of_Operations. If there were more than one token in the Stage_1_OTO list, the token would be chosen by the Shortest Processing Time (SPT) method. In Figure 6, workpieces with the shortest operation time (170 minutes) were workpiece 5 and workpiece 11. Therefore, those workpieces would be selected randomly. Selecting_ Job would consume those tokens and produce a token (identifier) in Selected_Mach_2. This token represents which machine would receive a workpiece with the shortest operation time.

Picking Job Submodule

The Picking job had seventeen places and seven transitions. However, there were only five places and one transition for picking job submodule in the loading/unloading station. Places in the picking job submodule were Crane_Idle, Mach_Cap, Selected_ Job, Selected_Mach_2, and Crane_Loaded. Crane_ Idle was a place to store one idle token (c) of the crane. Mach_Cap described as the total capacity of all machines. Eight tokens (cap) in Mach_Cap show that the slots in all machines were empty. Selected_Job was a place to store a workpiece which would be delivered by a stacker crane. Selected_Mach_2 described as a place to store the targeted machine. Crane_Loaded shown a workpiece which had already been loaded in a stacker crane. Figure 10 shows the picking job submodule.

The transition in the picking job submodule was Picking_Job. This transition was enabled when there was an idle token (c) in Crane_Idle, a workpiece token in Selected_Job, a machine target token in Selected_ Mach_2, and a token (cap) in Mach_Cap. The Picking_Job transition would consume all of those tokens and produce a token in the Crane_Loaded. In Fig. 11, Crane is loaded with a workpiece number 11 and deliver it to Machine 1.

Machine Selection Submodule

The machine selection submodule had five places and three transitions. Places in the machine selection submodule were Available_Mach, Notification_ for_ MS, List_of_AM, Available_SM_Act, and Selected_ Mach. Available_Mach was a place to store a token (identifier) that used as input to machine selection submodule. Notification_for_MS was a notification to execute machine selection submodule because the stacker crane has delivered a workpiece and ready to deliver the next workpiece. Available_Mach and Notification_for_MS were input for the machine selection submodule. They could be found in the other submodules. List_of_AM was a list of the available machines. Token identifier value (d1,1) in the List_of_AM showed the Machine 1 was available, (d2,2) showed Machine 2 was available, and so on. Available_SM_Act described the alternative of machine selection action that could be done. A token (dum) in the Available_SM_Act restricted only one machine can be selected at a time. Selected_Mach was a chosen machine to process a workpiece. Selected_Mach was an output of the machine selection submodule. It could be used as input for the other submodules. Figure 12 showed the machine selection submodule.

The transitions in machine selection submodule were Selecting_Mach, Enlisting_AM, and Notifying_MS_ Procedure. The selecting_mach transition was enabled when there were a token (identifier) in List_of_AM and a token (dum) in Available_SM_Act. Selecting_Mach would consume those tokens and generate an output in two places. First, the Selecting_ Mach transition would reduce token (identifier) list in the List_of_AM according to the selected machine. Second, the Selecting_Mach transition would produce a token (identifier) in the Selected_Mach, which represented the selected machine. If there were more than one token (identifier) in the List_of_AM, the system would choose a machine randomly. Enlisting_AM transition was enabled when there was an input token (identifier), for example (d1,1), in Available_Mach place. Enlisting_AM would insert this token to a list in List_of_AM, for example ([(d2,2),(d3,3),(d4,4)]). When Enlisting_AM was enabled, the result is ([(d2,2), (d3,3),(d4,4),(d1,1)]). Notifying_MS_Procedure was enabled when there was a token (dum) in Notification_for_MS. Notifying_MS_Procedure would consume this token and produce a token (dum) in Available_SM_Act.

Delivery to Pallet Stocker Submodule

The Delivery to pallet stocker submodule had seven places and four transitions. Places in the delivery to pallet stocker submodule were Job_Loaded_On_ Crane, M1_Not_Ready, Unloaded_State, and Pallet_ Stocker. Job_Loaded_On_Crane was input from the picking mechanism submodule. This place states that a workpiece had been loaded on a stacker crane. A token ("Job11_2", S2) in Job_Loaded_On_Crane place informed job 11 in a stage-two operation had been loaded on a stacker crane. M1_Not_Ready was a place to store token (r1) described M1 was not

ready to send a pick-up request. Unloaded_State stated that the stacker crane was empty. A token (c) informed the crane was not empty. Pallet_Stocker was a place to store a finished workpiece. A token ("Job11_2", S2) in Pallet_Stocker informed the job 11 in the stage-two operation had been delivered to a pallet stocker.

The transitions in the delivery to pallet stocker submodule were t1, t2, t3, and t4. T1 was enabled when there was a token (r1) in Job_Loaded_ On_ Crane. This transition will produce tokens in three places: at Unloaded_State, M1_Not_Ready, and Pallet_Stocker. Figure 13 shows the Enabled Delivery to pallet stocker submodule.

Proceed to Stage Two Submodule

The Proceed to stage two submodule had eight places and four transitions. Places in the proceeded to stage two submodule were Job_Loaded_On_ Crane, Counter_S1_Job_To_S2, M1_Not_Ready, Waiting_For_The_Next_Stage, and Unloaded_State. Job_ Loaded_On_Crane was input from the picking mechanism submodule. Counter_S1_Job_To_S2 was used to store which workpiece had already been delivered to a loading/unloading station for a stage-two operation. M1_Not_Ready was a place to store token (r1). It described M1 was not ready to send a pick-up request. Waiting_For_The_Next_Stage was used to store a workpiece which had a stage-two operation. Unloaded_State states that a stacker crane was empty.

The transitions in the proceed to stage two submodule were t5, t6, t7, and t8. The t5 transition was enabled when there was a token (r1) in the Job_ Loaded_On_Crane place. The t5 transition would produce tokens in four places: at Counter_S1_ Job_to_S2, M1_Not_Ready, Waiting_For_Next_ Stage, and Unloaded_State. Figure 14 shows the Enabled Proceed to stage two submodule.

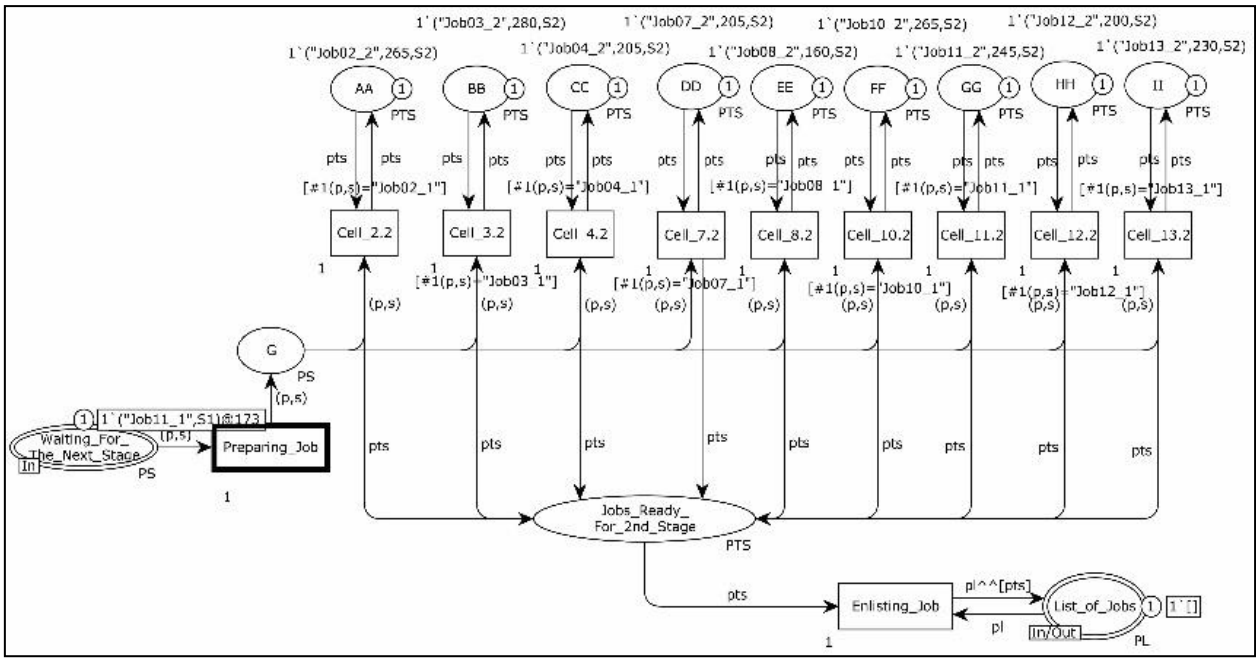

Figure 15. Stage two procedure

Table 1. Job operation data

Work piece	Stage	CPN Name	Operation Time (min)	Work piece	Stage	CPN Name	Operation Time (min)
1	1	01_1	190	8	2	08_2	160
2	1	02_1	255	9	1	09_1	305
	2	02_2	265	10	1	10_1	240
3	1	03_1	320		2	10_2	265
	2	03_2	280	11	1	11_1	170
4	1	04_1	265		2	11_2	245
	2	04_2	205	12	1	12_1	320
5	1	05_1	170		2	12_2	200
6	1	06_1	175	13	1	13_1	240
7	1	07_1	270		2	13_2	230
	2	07_2	205	14	1	14_1	155
8	1	08_1	250	15	1	15_2	165

Stage two procedure was a part of the proceed to stage two submodule. The Stage two procedure was used to update the data on the finished workpiece that sends back to a loading/unloading station. The Stage two procedure had thirteen places and eleven transitions. Places in stage two procedure were AA, BB, CC, DD, EE, FF, GG, HH, II, G, Waiting_ For_The_Next_ Stage, Job_Ready_For_2nd_Stage, and List_of_Jobs. Places AA until II used to store the stage-two information for every workpiece which had the stage-two operation. A token ("Job02_2",265, S2) in place AA informed a workpiece number 02 had a stage-two operation time 265 minutes. G was used to store a workpiece that had been prepared for a stage-two operation. Waiting_For_The_Next_Stage was a place to describe a workpiece that had been arrived in a loading/unloading station and waited for a stage-two operation. A token ("Job11_1", S1) showed a workpiece number 11 was waiting for a stage-two operation. Job_Ready_For_2nd_Stage was used to store the data of a workpiece that had been updated and ready for a stage-two operation. List_of_Jobs was a place to store a workpiece that will be delivered to a machine. Figure 15 showed a stage two procedure.

The transitions in the stage two procedure were Cell_2.2, Cell_3.2, Cell_4.2, Cell_7.2, Cell_8.2, Cell_ 10.2, Cell_11.2, Cell_12.2, Cell_13.2, Preparing_Job, and Enlisting_Job. Cell_2.2 until Cell_13.2 transitions were enabled when there were a token in each corresponding place (AA, BB,…, II) and a token in the G places. Each transition would produce a token in the Jobs_Ready_For_2nd_Stage and reduce a token in its similar places (AA, BB,…, II). The Preparing_Job transition was enabled when there was a token in Waiting_For_The_Next_Stage places. This transition would produce a token in the G place. The Enlisting_Job transition was enabled when there was a token in Jobs_Ready_For_ 2nd_Stage and a token in List_of_Jobs. This transition will update the token in the List_of_Jobs place.

Table 2. List of machine process

Machine	CPN name					
M1	11_1	06_1	11_2	07_1	12_1	07_2
M2	14_1	13_1	08_1	13_2	08_2	03_1
M3	05_1	10_1	04_1	10_2	04_2	03_2
M4	15_1	01_1	02_1	09_1	02_2	12_2

Table 3. Simulation verification

Machine 2		
CPN name	Start time	End time
14_1	3	158
13_1	158	398
08_1	398	648
13_2	648	878
08_2	878	1038
03_1	1038	1358

CPN Tools Simulation Performance Report
Net: /cygdrive/C/Users/Laws/Desktop
SPT Monitor ALL Rev.cpn

Note that these statistics have been calculated for data that is not necessarily independent or identically distributed.

Timed statistics

Name	Unit	Count	Duration	Utilization	Total Runtime
Marking_size_Crane'Stacker_Crane_Busy	Stacker Crane	98	144	8.74%	1647
Marking_size_M1_Proc_Job'M1_Busy	Machine M1	14	1385	84.09%	1647
Marking_size_M2_Proc_Job'M2_Busy	Machine M2	14	1355	82.27%	1647
Marking_size_M3_Proc_Job'M3_Busy	Machine M3	14	1425	86.52%	1647
Marking_size_M4_Proc_Job'M4_Busy	Machine M4	14	1380	83.79%	1647

Simulation steps executed: 436
Model time: 1467

Generated: Fri Dec 8 18:27:42 2017

Figure 16. Performance report

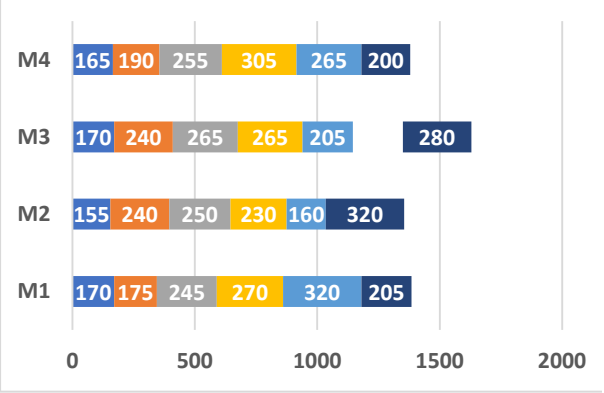

Figure 17. Gantt chart

Results and Discussions

Data

The model was tested on the real-world industry data based on Setiawan *et al.* [14]. The Complete data set is given in Table 1. There were 15 workpieces. Workpiece number 1, 5, 6, 9, 14, and 15 only needed one stage operation. Meanwhile, other workpieces needed a two-stage operation.

Result

Based on the performance report shown in Figure 16, it is known that the simulation needs 436 steps (enabled transition) for one cycle with makespan 1.647 minutes. The utilization level for all machines was good; it was around 82-86%. However, the utilization level for the stacker crane was relatively low (around 8,74%). The list of machining process in each machine is shown in Table 2.

Table 3 shows the simulation verification results, for example in Machine 2. Workpiece number 14 did not start on 0. It was because the stacker crane needed three minutes to pick-up a workpiece in a loading/ unloading station, move, and drop off a workpiece

number 14 to the Machine 2. The duration time in Machine 2 was 1355 minutes. This duration was the same as the performance report of Machine 2 duration time as shown in Figure 16. Figure 17 shows the Gantt chart of the model simulation.

Conclusions

Setiawan *et al.* [14] used Pharo 3.0 to simulate an FMS production scheduling. Pharo 3.0, required a new class definition with all interactions if the model is modified (e.g. added stacker crane). This paper showed an FMS production scheduling simulation using CPN. CPN developed with a set of sub-modules. This made the model more understandable. The CPN made a group of processes based on a function, location, machine, material flow, and more importantly in this paper, a stacker crane. FMS modelled by focusing on its stacker crane made the simulation more dynamic. Furthermore, the problem identification would be much easier.

In this paper, the FMS generated 1.647 minutes makespan; it was 5,23% greater than the Setiawan *et al.* approach [14], i.e., 1.565 minutes. The makespan's gap occurred because Setiawan *et al.* [14] did not consider the stacker crane. However, the stacker crane utilization in this paper was relatively low around 8,74%. This result happened because the duration of every stacker crane movement was assumed to be one minute. The assumption made the stacker crane time movement was much smaller compared to the job operation time. Therefore, the stacker crane utilization was low. For further improvement, the FMS can be simulated by considering the real stacker crane time movement to show a realistic stacker crane utilization level.

References

1. Shivanand, H.K., Benal, M. M., and Koti, V. *Flexible Manufacturing System*, New Age International (P) Limited Publishers, New Delhi, 2006.
2. Majija, N., Mpofu, K., Modungwa, D., Conceptual Development of Modular Machine Tools for Reconfigurable Manufacturing Systems *in* Azevedo, A., ed., *Advances in Sustainable and Competitive Manufacturing Systems*, Lecture Notes in Mechanical Engineering, Springer International Publishing Switzerland, 2013, pp. 467-478.
3. Bohn, R. and Jaikumar, R., *From Filing and Fitting to Flexible Manufacturing*, Now Publishers Inc., USA 2005.
4. Setiawan, A., Budiyanto, S., and Martawirya, Y.Y. Pengembangan Model Penjadwalan Perawatan dengan Semi Markov Process untuk Fasilitas Flexible Manufacturing System. *Proceedings Seminar Nasional Tahunan Teknik Mesin* XV, Bandung, 2016.
5. Sule, D.R., *Production Planning and Industrial Scheduling*. USA: CRC Press, 2008.
6. Zhan, Q., Manier, H., and Manier, M., Metaheuristics for Job Shop Scheduling with Transportation *in* Jarboui, B., Siarry, P., And Teghem, J., ed., *Metaheuristics for Production Scheduling*, Great Britain and USA: ISTE Ltd and Wiley, 2013, pp. 465-493.
7. Sahraeian, A., Minimizing Makespan in Flow Shop Scheduling Using a Network Approach *in* Righi, R., ed., *Production Scheduling*. London: IntechOpen, 2012, pp. 47-64.
8. Setiawan, A., Susan, and Pakpahan, E.K.A., Penjadwalan Job Shop pada Empat Mesin Identik dengan Menggunakan Metode Shortest Processing Time dan Genetic Algorithm, *Jurnal Telematika* ITHB, 9**,** 2014, pp.19-24.
9. Setiawan, A., Wangsaputra, R., Halim, A.H., and Martawirya, Y. Y., A Production Scheduling Model Considering Cutting Tools for an FMS to Minimize Makespan, *Proceedings of the Asia Pacific Industrial Engineering & Management System Conference*, Vietnam, 2015.
10. Pakpahan, E.K.A., Kristina, S., and Setiawan, A., Proposed Algorithm to Improve Job Shop Production Scheduling Using Ant Colony Optimization Method. *IOP Conf. Series: Materials Science and Engineering* 277 01205, 2017.
11. Sitepu, T.E.N., Setiawan, A., and Candra, A.K., Penjadwalan Job Shop Dua Stages dan Penentuan Perkakas Potong pada Flexible Manufacturing System Menggunakan Genetic Algorithm, *Jurnal Telematika* ITHB, 12, 2017, pp. 123-132.
12. Setiawan, A., Wangsaputra, R., Martawirya, Y.Y., and Halim, A.H., An FMS Dynamic Production Scheduling Algorithm Considering Cutting Tool Failure and Cutting Tool Life, *IOP Conf. Series: Materials Science and Engineering* 114 012052, 2015.
13. Setiawan, A., Wangsaputra, R., Halim, A.H., and Martawirya, Y. Y., A Job Rescheduling Model Considering Cutting Tool Failure and Cutting Tool Life for A Flexible Manufacturig System, *Proceedings of the Asia Pacific Industrial Engineering & Management System Conference*, Taiwan, 2016.
14. Setiawan, A., Qashmal L., Wangsaputra, R., Martawirya, Y. Y., and Halim, A. H., An Object-oriented Modelling of Production Scheduling for Flexible Manufacturing System, *Journal of Applied Mechanics and Materials*, 842, 2016, pp. 345-354.
15. Gradisar, D. and Music, G., Automated Petri-Net Modelling for Batch Production Scheduling

in Pawlewski, P., ed., *Petri Nets – Manufacturing and Computer Science*. Croatia: Intech, 2012, pp. 3-26.
16. Pan, Y. A., A Computationally Improved Optimal Solution for Deadlocked Problems of Flexible Manufacturing Systems Using Theory of Regions *in* Pawlewski, P., ed., *Petri Nets – Manufacturing and Computer Science*. Croatia: Intech, 2012, pp. 51-73.
17. Yasuda, G., Design and Implementation of Hierarchical and Distributed Control for Robotic Manufacturing Systems Using Petri Nets *in* Pawlewski, P., ed., *Petri Nets: Applications*. Croatia: Intech, 2010, pp. 379-392.
18. Jensen, K. and Kristensen, L.M., and Wells, L., Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems, *International Journal of Software Tools Technology Transferer*, 9, 2007, pp.213-254.
19. Igei, P., Cugnasca, C.E., Junqueira, F., Miyagi, P.E., and Garcia, J.I.,Modelling of Collaborative Production Systems Using Coloured Petri Nets, *International Conference on Pervasive and Embedded Computing and Communication System*, 2011.
20. Rocha de Carvalho, H.J. and Porto, A.J.V., A Colored Petri Net Based Modelling and Simulation of Multi-Product Manufacturing Systems, 19th *International Conference on Production Research*, Chile, 2007.
21. Long, F., Zeiler, P., and Bertsche, B., Potentials of Coloured Petri Nets for Realistic Availability Modelling of Production Systems in Industry 4.0, 25th *European Safety and Reliability Conference*, Switzerland, 2015.