Modeling and Simulating Enterprise Architecture Activities Using a Non Preemptive Multiprocessor System

Augusto Garcia Fabbri, Franciny Medeiros Barreto, and Joslaine Cristina Jeske de Freitas
Federal University of Jataí, gufabbri, francinymedeiros, joslaine@gmail.com

***Abstract* - Enterprise Architecture (EA) is a way for organizing the operations and structure of a business. It is also defined as a set of artifacts that describe the objects of an organization or an enterprise that include IT (Information Technology) alignment documentation, organizational models, reusable components, architectural patterns, and guiding principles of the design and evolution of its objects. In order to introduce technological advances and help companies to define a corporate strategy for maintaining their capacity, this article presents a model for EA using a non-preemptive multiprocessing system. Colored Petri nets (NPCs) make it possible to model very large and complex systems because they can represent data types and different levels of abstraction. The complex color sets, like arrays of records, are applied in the models of scheduling used in this paper (one and two processors) to simplify the model and increase the abstraction capability compared to models that do not used complex systems. The proposed models automatically execute processes with input times, service times and the name of a single non-preemptive method. Beside this, they calculate the waiting and turnaround time of processes further idle times of the one or two processors. Thus, all necessary details related to scheduling and running processes are obtained for processing. In addition, the results of the comparison between models with one or two processors show that there is a significant decrease in the final execution time for the models with two processors.**

Index Terms - Colored Petri Net, CPN Tools, Enterprise Architecture, Multiprocessor.

INTRODUCTION

The Enterprise Architecture (EA) is used to produce results and manage changes with quality it is also the foundation of a corporations to leverage technological innovations. EA does not have just one definition, for one of the pioneers in the field; John Zachman is a set of representations that are relevant to a company's description [1]. It is extremely important for the business to define a strategy to improve the integration of the business and a great capacity of quickly change e agility [2]. This work is based on [3] and aims to find solutions for companies to define a great corporate strategy and to introduce technological advances in the business world. The idea is to add a processing core in the model presented in [3], and thus improve the distribution of processes for the introduction and execution of corporate strategy. Therefore, as the main objective, the paper aims to provide a non-preemptive multiprocessing modeling base and compare the results obtained with the results presented in [3].

Colored Petri Nets, unlike classic Petri nets, allow the coloring of the tokens, which can then represent complex types of data, reduce the models sizes and describe different types of resources and processes. Using this type of approach makes possible to obtain a higher level of abstraction from the models. Employing Colored Petri Nets with the CPN Tools optimizes the creation, simulation, and extraction of model data. This model computes as final result the service and waiting time of the processes, and its goal is to make a comparison between both models, single and multiprocessor model.

The paper will be presented as follows. Section 2 introduces the concepts of EA and CPN. Section 3 presents the model specification. Section 4 shows initial marking and model of system. Section 5, functions of the model. Section 6, detailed operation of model. Section 7, shows the results of simulations. Finally, section 8 concludes the paper and provides references for additional works.

THEORETICAL FRAMEWORK

Enterprise Architecture is defined by one of the pioneers in the field, John A. Zachman as: "A set of descriptive representations that are relevant to a company's description, which can be produced in accordance with management requirements and maintained during the its useful life" [1].

The importance of architecture within a company refers to the success or failure of the business, where integration, capacity for change and agility are of the utmost importance for a good strategy to be defined, integrating all its elements.

According to [2] is the way to execute and operate the company's global strategy in order to maintain a competitive advantage. In [4], EA has some advantages when implemented:

- Ability to unify and integrate data across the enterprise and establish connections with external partners.

- Ability to unify and integrate business processes across the enterprise.
- Reduced delivery time for a solution and development costs by maximizing the use of business models
- Greater agility in business changes.
- Promptly available company documentation.
- Ability to create and maintain a common vision of the future in a shared way for the business and Information Technology areas.

EA can be structured around four objectives: effectiveness, efficiency, agility and durability. So the focus is on delivery rather than strategy development. [2].

The Petri nets were originated in the thesis of Carl Adam Petri, titled Communication with Automata. It is a theory that adapts to a large number of applications in which notions of events and simultaneous evolutions are important [5]. They are considered as a graphical and mathematical tool of formal representation, allowing modeling, analysis and control of systems. It can be seen as a bipartite and directed graph with two types of nodes, called places and transitions and connected by means of directed arcs.

A place can be interpreted as a condition, partial state, existence of resource, among others. A transition is associated with events that occur in a system. The token indicates a condition being checked, such as a certain feature available.

Colored Petri Nets, as well as all other variants, have emerged from the thesis of Carl Adam Petri. They were idealized by Kurt Jensen, from his article entitled Colored Petri Nets and the Invariant Method, inducing a series of studies on the coloring of the chips. They have a high level of abstraction, reducing the size of the models, allowing numerous models with a degree of difficulty inferior to the other structures of Petri nets. A color is assigned to each element of the model, designing the description of different features and processes [6].

CPN Tools is a developed tool capable of working with different types of Petri nets, including Colored Petri Nets. It enables the creation, simulation and extraction of reports that determine the viability of the models. The tool optimizes the creation, manipulation and simulation of the models, making it a good combination when using Petri nets[6].

Some advantages can be highlighted when working with Petri nets and the CPN Tools tool, such as:

- Ease of making small adaptations of models after comparing results.
- Integration of complex processes and data into a single model.
- Simulation environment that follows a trigger sequence step by step.

An introduction to the practical use of CPN can be found at [7].

MODEL SPECIFICATION

In this paper a model is based in [3] and a timed Colored Petri Nets formalism of two processors scheduling with non preemptive scheduling methods names FCFS, SJF, HRRN and PR is presented.

In presented model, complex color sets like arrays of records are applied. Using complex color sets simplifies the model and increases its performance in comparison with similar models that do not used complex systems. Presented model accepts processes, their input times and service time beyond name of a single non-preemptive scheduling method as input and automatically executes the processes. This model compute waiting time, turnaround time of processes and idle times of the processor. Model is designed so that all the necessary details regarding scheduling and running processes can be obtained for subsequent processing, and goal of the model is not only running of processes.

INITIAL MARKING AND MODEL OF SYSTEM

Initial marking of the model is defined as follows:
Val initialProcess =
[{PI=1, IT=1, ST=4, WT=0, ES=0, PR=(2,0)},
{PI=2, IT=2, ST=3, WT=0, ES=0, PR=(1,0)},
{PI=3, IT=3, ST=2, WT=0, ES=0, PR=(2,0)},
{PI=4, IT=4, ST=3, WT=0, ES=0, PR=(3,0)},
{PI=5, IT=5, ST=3, WT=0, ES=0, PR=(1,0)},
{PI=6, IT=1, ST=3, WT=0, ES=0, PR=(4,0)}]

The initial marking of this article differs only in the initial time, where all the processes had their time reduced so that they entered the queue of finished processes in a single time, in order to analyze the behavior of the model and to verify the difference between the models. The table I shows the attributes of the processes to be executed in the model.

TABLE I
INPUT PROCESSES

Process Index	Input Time	Service Time	Priority
P1	1	4	2
P2	2	3	1
P3	3	2	2
P4	4	3	3
P5	5	3	1
P6	1	3	4

It is important to note that although the input time is different, when starting the simulation, all the process will reach the time and go to the ready queue.

I. Model of the System

The figure I shows a CPN model proposed on this article.

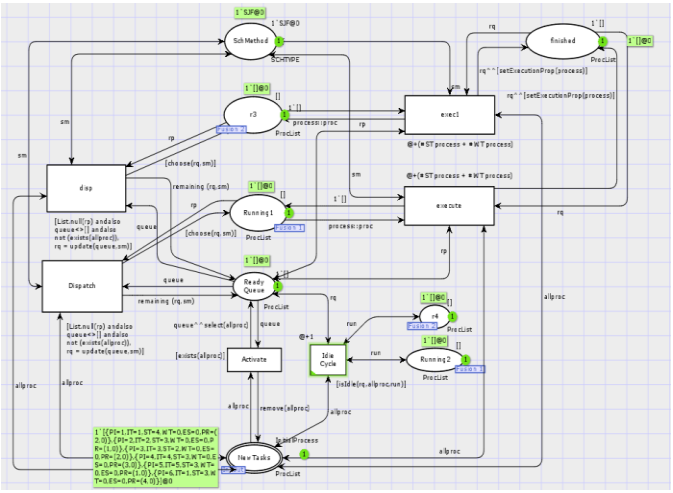

FIGURE I
CPN MODEL

In this model were added a new processor and a new dispatcher in order to expedite the processing of the queue. Also the final time computed was changes to the sum of the waiting times and service time the processes.

FUNCTION OF THE MODEL

All the functions used in this model can be found in [3]. Only the functions Select and Remove that are not described are going to be elucidated below.

```
fun select(Li:ProcList) : ProcList =
  let  val   n = List.length(Li)
       val   L = ref  []
       val   i = ref 0
  in
     while !i< n   do (
         let val p = List.nth(Li,!i)
         in if onTime(p) then
                  if  List.length(!L)  =  0
then
                        L :=[p]
                  else L :=  !L ^^ [p]
            else ()
         end;
         i := !i + 1 );     (* while i *)
     !L
  End
```

The function select receives a list of processes as input and checks the list from the beginning to the end. Calling the function onTime, it will verify if there is a process in the list that input time is reached based on the current simulation time, if there is, that process will be added to the output list that the function produces as output.

```
fun remove(Li:ProcList) : ProcList =
  let  val   n = List.length(Li)
       val   L = ref  []
       val   i = ref 0
  in
     while !i< n   do (
         let val p = List.nth(Li,!i)
         in if onTime(p)=false then
             if List.length(!L) = 0 then
                        L :=[p]
                  else L :=  !L ^^ [p]
            else ()
         end;
         i := !i + 1 );     (* while i *)
     !L
  End
```

Function remove also receives a list of process as input parameter and produces a list of process as output. It checks if there is a process in the input list which input time is not reached based on the current time simulation. All the process that input time was not reached will be added to the output list and then return to the place NewTasks.

DETAILED OPERATION OF MODEL

The operation of transitions, arc inscriptions and guard transitions are detailed explained in [3]. This section is intended to explain the difference in functionality when incrementing one process core in the model..

When initiating the simulation, just like the single processor model, all the processes are in place ReadyQueue. The input time of the processes will be checked by the function exists, when reached, the processes will go to the ReadyQueue place. Then the difference emerges. In the model proposed in this paper, only one process can be computed at a time, but the process can be inside one processor waiting for the other to finish, decreasing the time with they are executed, as shown in the figure II.

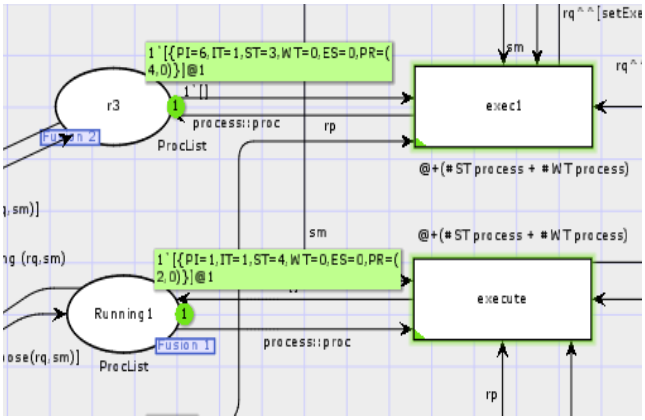

FIGURE II
PARALLEL PROCESSES RUNNING

Observing that both entered the processors when the simulation time is at @1 unit of time. At the end of the execution of these two processes, the time computed is at

@8 units of time, which is the sum of the services times of both processes and the initializing time of the simulation.

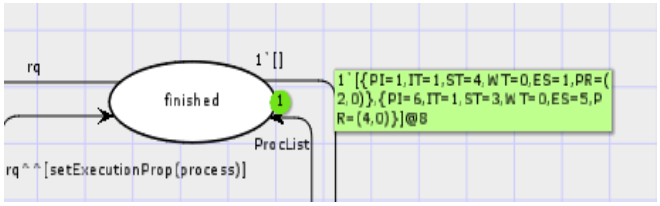

FIGURE III
CURRENT FINAL TIME OF PROCESSES

In the single processor model, since the two processes had reached their input time and were in the place ReadyQueue, one of the process would have its waiting time equal to the execution time of the other, and the final time would be the sum of the services times with the waiting time, totalizing @12 units of time at the end of execution.

RESULTS OF SIMULATION

The model was replicated 15 times using the processes of Table I in all of the scheduling methods described. The table II shows the comparison between the multiprocessor model and the model proposed in [3].

TABLE II
FINAL TIMES OF BOTH SIMULATIONS

Scheduling Method	Multiprocessor	Single Processor
	Final time	
SJF	76	115
HRRN	88	184
PR	82	132
FCFS	98	192

The resulting times of the multiprocessor model are the arithmetic mean of the fifteen replications. The time resulting from the single processor model does not change over the replications.

The FinalTasks place computes the waiting time added to the service time of the processes, resulting in the final time of the simulation of the model, this change was to take the waiting time into account. When modifying the input time, al the processes will be ready to run when the simulation starts, making the queue bigger in order to demonstrate the difference of computing power of models.

CONCLUSIONS

In order to introduce technological advancements and help companies define a corporate strategy to maintain their capacity, this article presented a model for EA using a non-preemptive multiprocessing system and compared with the model presented in [3].

Using the CPN tools, it was possible to combine the representation capacity of Petri nets and the power of the ML programming language with data types and different levels of abstraction. Complex color sets, such as record matrices, have been applied to the sizing models used in this document (one and two processors) to simplify the model and increase the abstraction capability.

The proposed models automatically executed processes with input times, service times and the name of a single non-preemptive scheduling method. In addition, they calculate the wait time and process return time, and idle times of one or two processors. Thus, all necessary details related to the scheduling and executions of processes were obtained for processing.

The results of the simulation comparison of the models showed that the two-processor model obtained a lower simulation time than the the single processor model. It can be observed a decrease of approximately 56% of the total time in relation to the single processor model. This shows that adding a process core to the model brings a considerable advance when running processes and can be helpful to introduce a better corporate strategy, improving the results in the business environment.

REFERENCES

[1] Zachman, J. A. Enterprise architecture: The issue of the century. Database Programming and Design, v. 10, n. 3, p. 44–53, 1997.
[2] Hoogervorst, J. Enterprise architecture: Enabling integration, agility and change. International Journal of Cooperative Information Systems, World Scientific, v. 13, n. 03, p.213–233, 2004.
[3] Pashazadeh, S.; Niyari, E. A. Modeling enterprise architecture using timed colored petri net: Single processor scheduling., 2014.
[4] Lapalme, J. Three schools of thought on enterprise architecture. IT professional, IEEE, v. 14, n. 6, p. 37–43, 2012.
[5] Cardoso, J.; Valette, R. Redes de Petri : Editora da UFSC, 1997.
[6] Jensen, K. and Kristensen, L.. Coloured Petri Nets. Springer. 2009.
[7] Kristensen, L. M., Jrgensen, J. B., and Jensen, K. Application of Coloured Petri Nets in System Development. In Lecture on Concurrency and Petri Nets, Jorg Desel, Wolfgang Reisig and Grezegorz Rozenberg (Eds.), Springer, LNCS 3089, pages 626–685. Springer-Verlag, 2004.

AUTHOR INFORMATION

Augusto Garcia Fabbri, undergraduate student in computer science, Federal University of Jataí.

Franciny Medeiros Barreto, doctoral student in computer science, Federal University of Uberlândia

Joslaine Cristina Jeske de Freitas, Professor, Department of Computer Science, Federal University of Jataí.