Journal of Novel Applied Sciences
Available online at www.jnasci.org
©2013 JNAS Journal-2013-2-S2/906-912
ISSN 2322-5149 ©2013 JNAS

Scheduling Algorithm by Business Process Management System

Mehdi Effat Parvar[1], Masoud Mobasheri Demne[2*] and Rama Nikbakhsh[3]

1- ECE Department, Ardabil Branch, Islamic Azad University, Ardabil, Iran
2- MS of Software Engineering, IRIB, IT Department
3- MS of public administration, IRIB, IT Department

***Corresponding author*:** Masoud Mobasheri Demne

ABSTRACT: In this article we try to present a method for scheduling the usage time of recourses in systems which base on processes concepts. At first we explain the concepts of BPMS , and then use BMPS' concepts to simulate a group of related systems. After that we simulate the model by using a method similar to the Markov's model. In the end we, by using a purposed algorithm, offer a table for dividing the resources fairly.

***Keywords*:** Scheduling, BPMS, SOA, Markov.

Abbreviations
SOA: Service-oriented architecture
BPEL: Business Process Execution Language
WSFL: Web Services Flow Language
XSD: XML Schema Definition
WSDL: Web Services Description Language

INTRODUCTION

In the past, it was used to use single usage systems for every process or tasks. However as time goes on we understand that couldn't connect these independent systems tougher easily or at all, because all of them were designed for a specific task. Also the design of these systems was focused on different parts of an organization instead of all its parts, and so they couldn't fulfill the goals of the organization as desirable as possible. In this way the idea of integrated systems was born. In this model a system is constructed of some independent sub-systems that can communicate with one another, and so it can solve the mentioned problem.

BPM is a flexible method for implanting and controlling the process of business which lead to optimization and increasing productivity in organizations. Business processes coordinate and connect people, information systems, and businesses' partners in the processes, which leads to better customer services. It implements the production line in a way that every process becomes optimized during its lifetime cycle, and in the end leads to the optimization of business processes.

Organizations use BPMS to increase the efficiency of their operational activities. BMPS especially regulates the number of communications between systems, business processes, and human interactions. The desired results include economical thrift, automation of processes' path and employees' duties, and avoiding useless activities. It also adds activities and features like decision path-finding, data or forms transfer.

BPMS systems, from the technological point, are based on SOA standards, and have sets of tools for implementing processes. Even the implementation of processes by using SOA is obvious, but there is no clear picture of the relation between BPM and SOA. So sometimes they are considered as antagonist and sometimes as tantamount of one another. In general, there are two models for interaction between BPM and SOA.
SOA contains BPM: in this pattern we look to processes as services.
SOA with BPM: in this pattern the first goal is optimizing processes and the second one is efficiency

MATERIALS AND METHODS

Getting familiar with the concept of BMPS impplentationIMPLEMENTATION

BPEL is a language for describing processes and contains a standard for that purpose. Its origin goes back to WSFL and XLANG.

XLAN is a rich set of high level structure which is used for describing business processes. In the same way that XML, XSD, and WSDL are standards for using Internet, XLAN is for supporting works with .Net based on objects and messages.

Introducing BPEL and its Concepts

1. Process: a set of workflow
2. Task: a work or set of works that a process needs them to be accomplished for finishing.
3. Relationships: type of relations between tasks of one process and their sequence
4. Start of Process: showing the start point of the process and the first task that must be performed.
5. End of Process: showing the end point of the process and the last task that must be performed.

Defining and Solving the Problem

In any working system, we have limited resources which their usage times should be defined from the beginning. We have some specific users that need to have access to these resources in a precise order sequence and then release them, for doing their defined tasks. Each of these users has distinct workflow for themselves which performed on one or some machines. When more two or more users need to use one recourse simultaneously, it should be divided between them based on some priorities. In picture 1 we see a sample of process that described above.

Problem's Assumptions

1. Don't have Context Switch between processes
2. Tasks are Atomic
3. Number of machines is limited.
4. There are no limitations in the number of processes, and tasks and relationships between the tasks of one process.
5. Number of recourses is restricted.
6. The maximum numbers of resources which can assign for each process is defined.
7. If one recourse doesn't use by a process, other processes can take that recourse.
8. Number of allocated recourses for each process can't change.
9. Processes aren't depended to machines.

Purposed Method: We need a method for optimum usage all of the resources by processes, without facing Starving or Deadlock problems and considering maximum equanimity with extreme speed. For doing it we do the following steps:

1. Processes' required tasks matrix: P(i, j) showing the need of a process to tasks

Rules: For simulating processes we build a matrix which its row index contains the number of processes and its column index contains the number of tasks. Number of rows and columns are equal to number of processes and tasks. In this matrix for showing the start point of a process, we have number 0 in row 0 of a column and number 1 in the corresponding cell. If task i leads to task j, then put number 1 in row i and column j. If we have logical AND between tasks i and j, then put number 1 in row i and column j and the corresponding cell. If we have logical OR between tasks i and j, then put number 1 in row i and column j and number 0 in the corresponding cell.

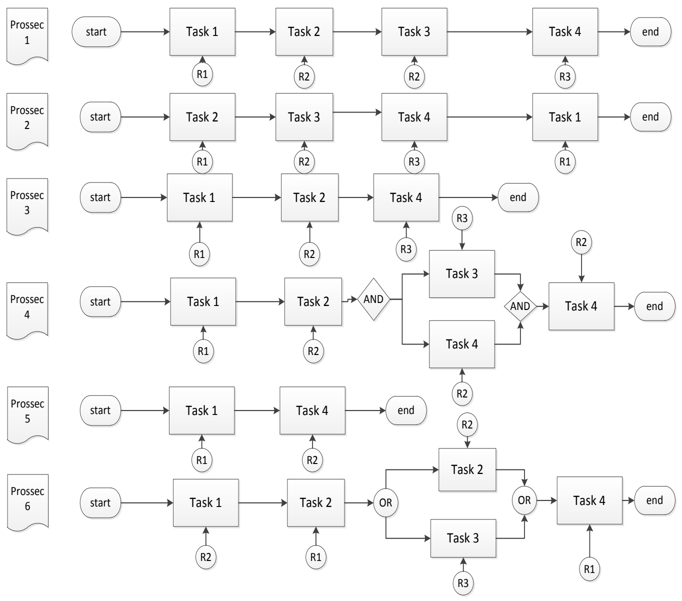


Figure 1

Pseudo Code

```
1. COUNT i, j=MAX (TASK) IN Process
2. IF COLUMN IN P (0, j)=0 AND ROW IN P (i, 0)=1 THEN START Process
3. IF TASK i $\rightarrow$TASK J THEN INSERT INTO P (i, j) =1 ELSE INSERT INTO P (i, j) =0
4. IF TASK i AND TASK j THEN INSERT INTO P (i, j) =1 , P (j, i)=1
5. IF TASK i OR TASK j THEN INSERT INTO P (i, j)=1 , P (j, i)=0
6. IF P (i, j)=P (j, i)=1 THEN END Process
```

Processes' required tasks matrix

$$P1(Ti,Tj)=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\0&0&0&1\end{bmatrix}\quad P2(Ti,Tj)=\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{bmatrix}\quad P3(Ti,Tj)=\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

$$P4(Ti,Tj)=\begin{bmatrix}0&1&0&0\\0&0&1&1\\0&0&1&1\\0&0&0&1\end{bmatrix}\quad P5(Ti,Tj)=\begin{bmatrix}0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{bmatrix}\quad P6(Ti,Tj)=\begin{bmatrix}0&1&0&0\\0&1&1&0\\0&0&0&1\\0&0&0&1\end{bmatrix}$$

2. Tasks' required resources matrix: N (i, j) showing the needs of a process' tasks to resources
Rules: If in one process a task needs a recourse, in this matrix we put number 1 in corresponding row and column.
IF TASK i $\rightarrow$RESOURCE J THEN INSERT INTO N (i, j) =1 ELSE INSERT INTO N (i, j) =0

Tasks' required resources matrix

$$P1(Ti,Rj)=\begin{bmatrix}1&0&0\\0&1&0\\0&1&0\\0&0&1\end{bmatrix}\qquad P2(Ti,Rj)=\begin{bmatrix}1&0&0\\1&0&0\\0&1&0\\0&0&1\end{bmatrix}\qquad P3(Ti,Rj)=\begin{bmatrix}1&0&0\\0&1&0\\0&0&0\\0&0&1\end{bmatrix}$$

$$P4(Ti,Rj)=\begin{bmatrix}1&0&0\\0&1&0\\0&1&1\\0&1&0\end{bmatrix}\qquad P5(Ti,Rj)=\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\\0&1&0\end{bmatrix}\qquad P6(Ti,Rj)=\begin{bmatrix}0&1&0\\1&0&0\\0&1&1\\1&0&0\end{bmatrix}$$

3. Rivalry Matrix: The matrix forms when some processes start using resources from different machines, and it's the direct result of peer-to-peer multiplying of processes' required tasks matrix.

Rivalry (i, j)=NP1 (i, j)*NP2 (i, j)*…..

Rules: if one row and one column in the rivalry matrix is 1, it shows that two processes need that recourse for a same task simultaneously.

1. IF ROW OR COLUMN IN Rivalry (i, j)=1 THEN P1 OR P2 OR Pn NEED RESOURCE J FOR TASK

4. State Matrix: It has one row and two columns.

Rules: $i \neq 0$, $j \geq 1$

1. First column is the sum of tasks if each process.

2. Second column is the sum of process' tasks need for resources.

Competition problem between several tasks of several processes

If we have competition between one or several tasks of several processes, then we will have competition for obtaining resources. For solving the problem by considering speed and equanimity between processes, we offer the following solution.

1. Forming the latest sate matrix: It's used for storing the last state of rival processes, and has one row and three columns LS [F-R, S-R, T-R]

First Column or F-R: Showing the number of remaining tasks of processes

Second Column or S-R: Showing the number of completed tasks of processes

Third Column or T-R: Showing the number of required resources for finishing each process

For solving the problem in three attempts, the following algorithm forms (Figure 2). It's necessary to remind that if parameters are equal, we will go to next step of algorithm.

Competition over holding machine (M_C)

In the case that several processes need a machine simultaneously, the winner process is the one which has more tasks for finishing.

Example for purposed problem: We have 6 processes that run in 2 machines simultaneously. If they face competition, we disturb the resources by using the proposed algorithm. For understanding the model, some steps are presented graphically.

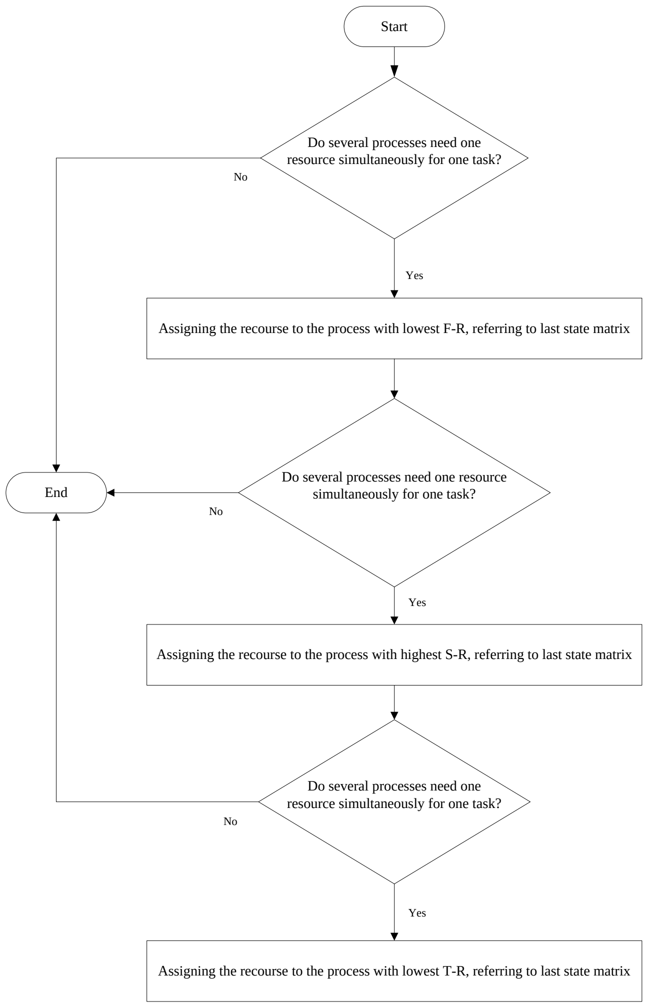
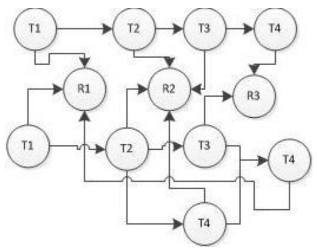

Start
Do several processes need one resource simultaneously for one task?
No
Yes
Assigning the recourse to the process with lowest F-R, referring to last state matrix
Do several processes need one resource simultaneously for one task?
End
No
Yes
Assigning the recourse to the process with highest S-R, referring to last state matrix
Do several processes need one resource simultaneously for one task?
No
Yes
Assigning the recourse to the process with lowest T-R, referring to last state matrix

Figure 2

P1 VS P4: Running processes 1 and 4 simultaneously in two machines



Graph of tasks' required resources for processes 1 and 4
Rivalry matrix, state matrix, and last state matrix of two processes

Solving problem by using the proposed algorithm

Table 1

M0	P1T2R2	P1T3R2	P1T4R3	P4T3R3	
M1	P4T1R1	F-R_R2	P4T2R2	P4TT4R2	P4T4R4

The above table shows the simultaneous running of processes 2, 5, and 6 in two machines

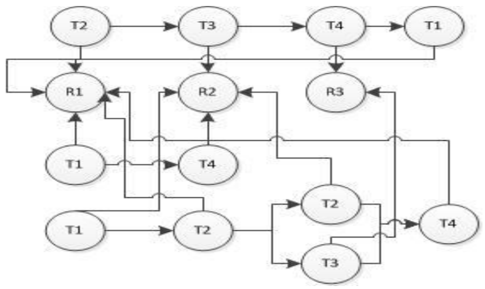


Figure 7. Graph of tasks' required resources for processes 2, 5 and 6

$$R(2,5,6) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$SP\,2 = \begin{bmatrix} 4 & 4 \end{bmatrix}$$

$$SP\,5 = \begin{bmatrix} 2 & 2 \end{bmatrix}$$

$$SP\,6 = \begin{bmatrix} 5 & 4 \end{bmatrix}$$

$$LS(2,5,6) = \begin{bmatrix} 4 & 0 & 4 \\ 2 & 0 & 2 \\ 5 & 0 & 5 \end{bmatrix}$$

Solving the problem with the proposed algorithm

Table 2

M0	P5T1R1	P4T2R2	P2T1R1	P2T3R2	P2T4R1	
M1	P6T1R2	P6T2R1	P6T2R3	P6T2R3		P6T4R1
	F_R_R1	S_R_P1			F_R_R1	

The above table shows the simultaneous running of process 1 with process 3 and process 4 with process 6 in two machines.

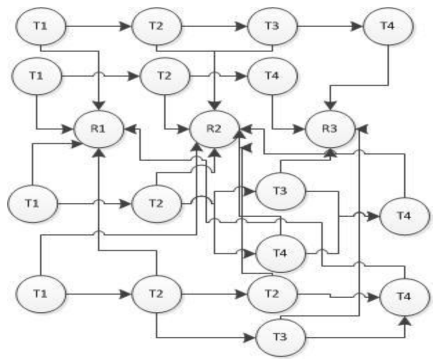


Figure 8. Graph of tasks' required resources for processes 1, 3, 4, and 6

$$R(1,3,4,6) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$SP1 = \begin{bmatrix} 4 & 4 \end{bmatrix}$$

$$SP3 = \begin{bmatrix} 3 & 3 \end{bmatrix}$$

$$SP4 = \begin{bmatrix} 5 & 5 \end{bmatrix}$$

$$SP6 = \begin{bmatrix} 5 & 5 \end{bmatrix}$$

$$LS(1,3,4,6) = \begin{bmatrix} 4 & 0 & 4 \\ 3 & 0 & 3 \\ 5 & 0 & 5 \\ 5 & 0 & 5 \end{bmatrix}$$

Table 3

M0	P3T1R1	P3T2R2	P1T2R2	P3T4R3	P1T3R3	P1T4R4	P4T4R2	P4T4R2	
M1	P6T1R2	P1T1R1	P4T4R1	P6T2R1	P6T2R2	P4T4R3	P4T4R3	P6T3R3	P6T4R1
	F_R_R1	S_R_R1	F_R_R1	MC	F_R_R3	S_R_R3	T_R_R3		

The above matrix shows the solution for simultaneous run of processes without facing deadlock or indefinite expectation of tasks for holding machines. In the first row you see the first machine and the name of the process that is running. In the second row you see the name of task and process that win the competition with other processes, and in the third row come the name of the rule which buy using it the process won.

RESULTS AND DISCUSSION

The proposed algorithm isn't only for solving scheduling problem in running tasks of processes which are based on BPMS. It can be used as a solution for scheduling works when we have limited resources.

Parameters of this algorithm are chosen arbitrary, and we assumed them the by considering the speed of finishing the process in the first place and dividing the resources (machines) fairly in the second place. These parameters can be change based on the requirements.

REFERENCES

Leymann F. 2001. Web Services Flow Language (WSFL 1.0), *IBM Software Group*, May

Snell J. 2001. Web services insider, Part 1: Reflections on SOAP, Software Engineer, Emerging Technologies, *EMC*, 01 April

Marinescu DC. 2002. Internet-Based Workflow Management: Towards a Semantics Web, A *Wiley-Interscience Publication, JOHN WILEY & SONS*, INC

Weske M. 2007. Business Process Management: Concepts, Languages, Architectures, *Springer*

Jeston J, Nelis J. 2006. Business Process Management Practical Guidelines to Successful Implementations, *Elsevier*, first edition.

Erl T. 2007. Service-Oriented Architecture: Concepts, Technology, and Design", *Prentice Hall*