Jin, Y., Zhou, H., Yang, H., Zhang, S. and Ge, J. (2017) 'An approach to locating delayed activities in software processes', *International Journal of Automation and Computing*. doi: 10.1007/s11633-017-1092-9.

DOI: XXX

An Approach to Locating Delayed Activities in Software Processes

Yunzhi Jin[1] Hua Zhou[1,2,3] Hongji Yang[4] Sijing Zhang[5] Jidong Ge[6]
[1]School of Software, Yunnan University, Kunming 650091, China
[2]Key Laboratory for Software Engineering of Yunnan Province, Kunming 650000, China
[3]Research Center of Cloud Computing of Yunnan Province, Kunming 650000, China
[4]Centre for Creative Computing, Bath Spa University, SN130RP, UK
[5]Department of Computer Science and Technology, University of Bedfordshire, LU1 3JU, UK
[6]Software Institute, Nanjing University, 210093, China

Abstract: Activity is now playing a vital role in software processes. To ensure the high-level efficiency of software processes, a key point is to locate those activities that own bigger occupation probabilities of resource in average execution time, called delayed activities, and then improve them. To this end, we firstly propose an approach to locating delayed activities in software processes. Furthermore, we present a case study, which exhibits the high-level efficiency of the approach, to concretely illustrate this new solution. Some beneficial analysis and reasonable modification are developed in the end.

Keywords: Locating of the delayed activities, Software process, Stochastic Petri-nets, Markov chain, Probability transfer matrix.

1 Introduction

In 1987, Osterweil[1] put forward the view that software processes are software, too. This view has been accepted by a large number of scholars and tightly attracted their attention since then. In view of the Standard for Information Technology-Software Life Cycle Processes (ISO/IEC 12207 Standard)[2], a software process can be defined as a set of interrelated activities that transform inputs into outputs, and each process is further denoted in terms of its own constituent activities. These all show that activity is an indispensable constituent of software process. As stated in Li[3] that, "software processes denote a set of interrelated processes in the software life cycle. A software process provides a framework for managing activities that can very easily get out of control in software development.", and Pressman[4] that, "The software development's work products (programs, documentation and data) are produced as consequences of the activities defined by the software processes".

As we all know, the key point to success in software processes is to ensure that the activities could be finished on time. Unfortunately, in most of the cases, some activities that own bigger occupation probabilities of resource in average execution time always exist. In this paper, we call them the delayed activities. Informally speaking, to locate the delayed activities in software process is just as to find the activities of the critical path in Activity on Edge (AOE) Network. Similarly, we locate the delayed activities in software processes and improve them so that the time cost of these activities will reduce and as a result, the efficiency of these software processes will increase. Hence, it is essential to locate the delayed activities in software processes in order to improve the software processes.

To locate the delayed activities in a software process, the Petri-nets will be used as an efficient tool to model the software process itself. The Petri-nets were firstly proposed by Doctor C.A. Petri[5] in Germany in 1962 and have gained popularity for representation of complex logical interactions (say synchronisation, sequentiality, concurrency and conflict, etc.) among activities. The principles and applications of Petri-nets have been widely discussed and developed since then. For example, Van Der Aalst[6] introduced workflow management as an application domain for Petri-nets, proposed state-of-the-art results with respect to the verification of workflows, and highlighted some Petri-net-based workflow tools. Likewise, Van Der Aalst and Ter Hofstede[7] presented a petri-net-based verification approach of workflow task structures and developed a verification tool to illustrate the applicability of the approach. Besides, Hamadi and Benatallah[8] put forward a Petri net-based algebra, used to model control flows, as a necessary constituent of reliable Web service composition process. See also Ge et al.[9] for the MOPN-SP-net model, which is a multi-view software process model based on multi-object Petri-nets. Specifically, on the basis of Petri-nets, Molloy[10] discussed the isomorphism between the behavior of Petri-nets with exponentially distributed transition rates and the Markov processes. What is more, Barbot and Kwiatkowska[11] introduced the Stochastic Petri-nets (SPN) to demonstrate how DNA walkers can be modelled. By the related SPN basic theory and description method, Han et al.[12] studied the SPN model for basic activities of software process and their relations, and discussed the simulation strategies of SPN model. On all accounts, it is feasible to use Petri-nets or SPN to describe software processes.

The SPN uses time parameter to describe system performance indices and is suit for time performance evaluation

Research article
Manuscript received date; revised date
This work is supported by the NSF of China (61462091), High-tech Industrial Development Program of Yunnan Province, China (No. 1956, in 2012), New Academic Researcher Award for Doctoral Candidates of Yunnan Province of China (ynu201414), NSYF of Yunnan Province of China (2014FD006), and the Postgraduates Science Foundation of Yunnan University (ynuy201424).
Recommended by Associate Editor xxxx

of system. For instance, Ajmone Marsan et al.[13] proposed a class of Generalized Stochastic Petri Nets (GSPNs) for the performance evaluation of multiprocessor systems. Furthermore, relying on the SPN, Lei et al.[14] analyze the performance of Device-to-Device (D2D) communications with dynamic interference. Similarly, Dong et al.[15] employed an approach to predicting the performance of web service composition, Shan et al.[16] constructed a formalized model of vehicular 1553B Bus System and then analysed the performance of the vehicular 1553B Bus System through simulation experiment.

Although there are many studies on Petri-nets or SPN for performance analysis in numerous fields, few of them provide any form of support for the locating of delayed activities in software processes, whereas these delayed activities in fact are the major important factor of the project cycle among all activities.

In software process, many researchers nowadays focus on improving activities by these measures such as increasing resources, excavating and executing some parallelizable tasks in activities. However, they hardly take account of the delayed activities, which may make us get half the results with twice the effort once the activities are numerous. To let us get twice the results with half the effort, it is necessary to locate the delayed activities before improving them.

Specifically, in the year of 2010, Jiao[17] presented an example in economics field based on SPN to locate the core opinion leader reflected by the values of probabilities of the places with tokens. Inspired by this, we in this paper locate delayed activities by its values of probabilities of the places with tokens. Unfortunately, the principles and frameworks of locating delayed activities have not been proposed yet, and few of approaches are devoted to locating of the delayed activities in software processes.

In view of these, we in this paper introduce a framework of locating the delayed activities in software processes and perform an algorithm to calculate the probabilities of the place with tokens. Concretely, we at first build a Transaction Flow Diagram (TFD) of software process and then constructively transfer it to SPN. In addition, by noting that SPNs are isomorphic to homogeneous Markov processes as shown in [18], we draw isomorphic Markov chain (MC) and reachable marking graph of the SPN. Moreover, by calculating the equations on probability transfer matrix of the MC, we locate the places containing tokens with the bigger values of probabilities, which correspond to the delayed activities in software processes. Finally, we present a practical example to show the correctness and rationality of this algorithm.

The plan of this paper is as follows. In Section 2, we introduce the background of our research work. In Section 3, we build up a novel approach to locate delayed activities in software process. A case study example is followed to illustrate this new approach in Section 4. Finally, the significance and the further application of this approach is discussed in Section 5.

2 Background

In this section, we present some background concepts on SPN, TFD and Markov chain which will be used in our approach.

2.1 Stochastic Petri-nets

This section reviews some basic theories of Stochastic Petri-nets (SPN)[19, 20, 21] model.

Assume that $\Sigma = (S, T; F, M_0, \lambda)$ is a stochastic Petri-nets, in which $\Sigma = (S, T; F, M_0)$ is a prototype Petri-nets and $\lambda : T \to R_0$ the mapping from T to R_0 with R_0 the reachable marking set.

Suppose that $T = \{\mathrm{t}_1, \mathrm{t}_2, \dots, \mathrm{t}_n\}$, $t_i \in T$, $\lambda(t_i) = \lambda_i$ are nonnegative real values, which represent the occurrence rates of the transition t_i (when meet the conditions of an occurrence). When the transition t_i are fired, the corresponding timed delay d_i are random variables satisfying $d_i(\tau) = e^{-\lambda_i \tau}$, where τ is the related time. Therefore, the average timed delay $\overline{d_i}$ of transition t_i is determined by $\overline{d_i} = \int_0^\infty e^{-\lambda_i \tau} d\tau = \frac{1}{\lambda_i}$.

In view of that the memoryless characteristics of the random variables obey negative exponential distribution, if Σ is a bounded stochastic Petri-net, then RG(Σ), the reachable marking graph of Σ, is isomorphic to a finite Markov chain (MC), and the state space of the MC is a reachable marking set $R(M_0)$ of Σ.

Assume that $\Sigma = (S, T; F, M_0, \lambda)$ is a Stochastic Petri-net, $\lambda = [\lambda_1, \lambda_2, ..., \lambda_n] (n = |T|)$, $R(M_0)$ the reachable marking set of Σ. Suppose that $|R(M_0)| = r$, then the r-order matrix

$$Q = [q_{ij}]_{r \times r} \tag{1}$$

is called a probability transfer matrix of the Σ, where

$$q_{ij} = \begin{cases} -\sum\limits_{M_i[>} \lambda_k, & if(i = j), \\ \lambda_k, & if(i \neq j, \exists t_k \in T \text{ and } M_i[t_k > M_j), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

With the help of the probability transfer matrix, we can calculate the steady state probability Q of r states (corresponding to r reachable marks of Σ) in the Markov chain. In generally, Q is a r-dimension vector $\prod = [\pi_1, \pi_2, ..., \pi_r] (r = |R(M_0)|)$, where π_i denote the steady probabilities of marking M_i. Here, the r-dimension vector $\prod$ satisfy the following equations:

$$\begin{cases} \prod Q = \mathbf{0}, \\ \sum\limits_{i=1}^{r} \pi_i = 1, \end{cases} \tag{3}$$

where Q is the probability transfer matrix as shown in (1) and (2). By solving (3), the vector $\prod$ can be obtained uniquely.

Using the steady state probability vector $\prod$, the actual system can be simulated by Stochastic Petri-nets for all kinds of performance evaluation. For example, the probabilities of the state set satisfied some special conditions can be worked out.

Set B a subset of $R(M_0)$. Then a marking M is an element of the subset, if and only if M meets some special conditions (representing a certain performance of the system). Therefore, we can calculate the probability of marking subsets B by $\prod = [\ \pi_1, \pi_2, \dots, \pi_r\ ]$ as follows

$$\rho(B) = \sum_{M_i \in B} \pi_i,$$

where π_i are the steady probabilities of marking M_i.

2.2 Transaction Flow Diagram

The Transaction Flow Diagram (TFD) is a graphic representation of the physical route or flow of communication associated with a business process. Moreover, it is used to structure and order a complex business system, or to reveal underlying structure of the business processes and their interaction. Additionally, it describes a completed specific business process focussed on business processing, and does not involve data.

It is wealth to mention that designing a TFD is of significance. Firstly, as a tool of exchanging ideas between system analysts and managers, the TFD is the basis for the successive system analysis of system analyst. Secondly, with the help of the TFD, the business processes that can be well processed by computers could be directly mined by system analysts. What is more, it is very helpful to analyse the reasonableness of business process by virtue of the TFD.

In what follows, we introduce the fundamental notations of TFD used in our paper, as shown in Fig.1. Sometimes, we have not use notations of Begin and End for convenience, or exceptions-the TFD is cyclic.

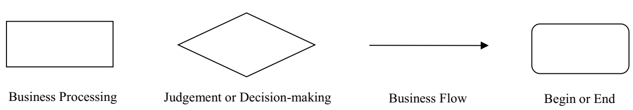

Fig. 1 The fundamental notations of TFD

2.3 Markov Chain

The research of a new vital type of chance process, in which the outcome of a given experiment can affect the outcome of the next one, was proposed by Markov in 1907. This type of process is named after Andrey Andreyevich Markov and called a Markov chain. Generally speaking, it possesses a property characterized as "memorylessness", called the Markov property, that is, the probability distribution of the next state depends only on the current state and not on the sequence of events that preceded it.

A Markov chain is described as follows[22, 23]: we have a set of states $\mathbf{S} = \{\ s_1, s_2, \dots, s_r\ \}$, the process starts in one and only one of these states at a given time and moves successively from one state to another. Each move is a step. If the chain is currently in state s_i, then it moves into the state s_j at the next step with a probability denoted by p_{ij}. The probabilities p_{ij} are called transition probabilities, and the probability does not depend on which states the chain was in before the current state. The process can remain in the state it is in, and this occurs with probability p_{ii}. An initial probability distribution, defined on $\mathbf{S}$, specifies the starting state. Usually this can be done by specifying a particular state as the starting state. We also employ the $r \times r$ transition matrix $\mathbf{P}$ with those p_{ij} to completely specify the Markov chain.

R. A. Howard[24] provides us with a vivid description of a Markov chain as a frog jumping on a set of lily pads. The frog starts on one of the pads and then jumps from lily pad to lily pad with the appropriate transition probabilities.

There are several kinds of Markov chains. Particularly, this paper involves only the *finite ergodic chain*. Here, an *ergodic chain* is one whose states come from a single ergodic set or equivalently-a chain in which it is possible to go from every state to every other state. While a *finite Markov chain* is a stochastic process which moves through a finite number of states, and for which the probability of entering a certain state depends on the last state occupied. What is more, the *finite Markov chain* starts in some state and undergoes transitions from one state to another successively on a state space.

So far, the Markov chains have been extensively applied in a large number of statistical models and even more areas. For instance, the Markov chain method has been suggested as a means of characterizing or summarizing economic data and of projecting the time path of certain economic variables by G.G. Judge and E.R. Swanson[25]. Moreover, B.W. Jiang et al.[26] formulated saliency detection via absorbing Markov chain on an image graph model.

3 Main Idea

The main goal of this section is to build up an approach to locate the delayed activities in software process. Firstly, we introduce the principles of this approach in section 3.1. Additionally, the framework of locating delayed activities is proposed in section 3.2. Finally, we perform an algorithm on calculating probabilities of the places with tokens.

3.1 Principles

In software process, the activities a_i consume resources from State *i-start* to State *i-finish* (State *i-start* denote the states of activities a_i that are not executed, while State *i-finish* accomplished), and in $1/\lambda_i$ the average execution time. For each activity, we calculate their values of occupation probabilities $P(\mu_i{=}1)$ of resources in average execution time. An activity a_j is called a delayed activity if it owns a bigger value of occupation probabilities of resource than others in software process, which means that the activity a_j does not make the most of resources.

In a word, even though the average execution time reflects how long an activity takes, it can not well reveal how much these activities have been delayed. In fact, while an activity is delayed or not is justified by its occupation probabilities of resource in average execution time.

3.2 Framework of locating delayed activities

We propose a framework of locating delayed activities in software processes, as shown in Fig.2.

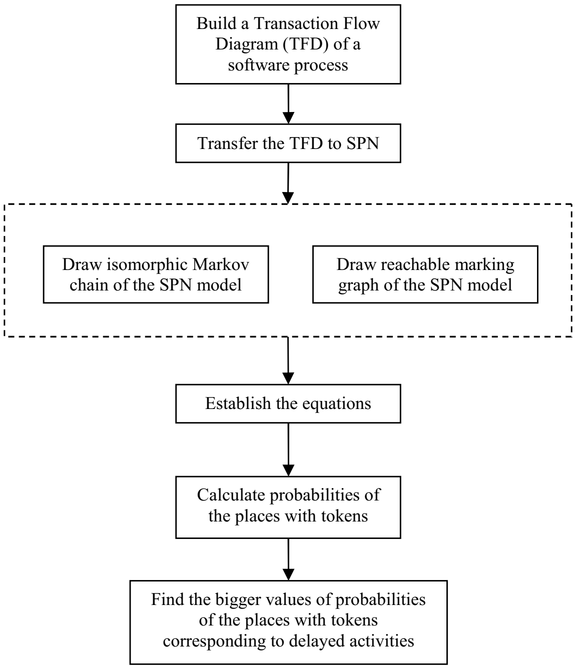

Fig. 2 The framework of locating delayed activities

This approach is based on a series of classical methods, say, Stochastic Petri-nets, Transaction Flow Diagram and Markov chain. By calculating the probability transfer matrix of the MC, we find that the bigger values of these state probabilities correspond to the delayed activities in software processes. This qualitative approach offers an effective method for locating the delayed activities in software processes.

3.3 Algorithm for calculating probabilities of the places with tokens

In what follows, we perform an algorithm for calculating probabilities of the places with tokens.

For any marking, the bigger values of steady state probability of these being μ_i tokens in each place correspond to the delayed activities in the software process. Here,

$$\mu_i = \begin{cases} 0, & \text{if place } P_i \text{ have none token in every marking,} \\ 1, & \text{otherwise.} \end{cases}$$

Assume that the Markov chain with r markings (M_1, M_2, ... , M_r) and reachable marking graph of SPN with m places (P_1, P_2, ... , P_m) have been achieved, then the probabilities of the places with tokens $P(\mu_i\!=\!1)$ can be calculated by the following Algorithm 1, in which $P(M_i)$ denote the stable probabilities of state marking M_i.

Next, we briefly prove the correctness of the algorithm by three steps. Here, we just give a general idea of proving. First, we prove that the balance equations of Continuous Time Markov Processes (CTMP), whose state spaces are finite, are equivalent to (3) provided that the sum of the properties of discrete stochastic valuables is 1. Then, we show that the solution of the (3) is unique. In the end, we calculate the steady state probability and order them from largest to smallest by the classical BubbleSort algorithm.

For the balance equations, one has the following result. Please see [27] for the details.

Lemma 1 Let $P(M_i)$= x_i ($1\leq x \leq r$), $\forall M_i \in [M_0 >$, for all M_u, $M_v \in [M_0 >$ and $M_i[t_v > M_v$, $M_u[t_u{>}M_i$, then the balance equations hold:

$$(\sum_v \lambda_v)x_i{=}(\sum_u \lambda_u x_u).$$

By Lemma 1, one can achieve r homogeneous equations

$$(\sum_v \lambda_v)x_i{-}(\sum_u \lambda_u x_u){=}\ 0\ (i{=}\ 1, 2,\ ...\ ,\ r), \tag{4}$$

which involves r independent variables (x_1, x_2, ... , x_r).

The equations (4) can be denoted as

$$\begin{cases} (x_1, x_2, ..., x_r)[q_{i1}]_{r\times 1} = 0, \\ (x_1, x_2, ..., x_r)[q_{i2}]_{r\times 1} = 0, \\ \quad\quad ... \\ (x_1, x_2, ..., x_r)[q_{ir}]_{r\times 1} = 0, \end{cases} \tag{5}$$

where q_{ij} are subject to (2).

Noting that the probability distribution of discrete stochastic valuables satisfies

$$P_X(k) = Prob\,[X = k]\,,\ \ \sum_{\text{all}k} P_X(k) = 1,$$

and that the states of Continuous Time Markov Processes are discrete, it follows that

$$\sum_{i=1}^{r} x_i{=}\ 1. \tag{6}$$

Therefore, the balance equations of CTMP (5) and (6) are equivalent to (3).

In what follows, we show the uniqueness of the solution of (3).

For Markov processes which are irreducible, aperiodic, and recurrent nonnull, the vector of steady state probabilities $\prod = [\pi_1, \pi_2, ..., \pi_r]$ is the unique solution[28] of (3). So it is reasonable to solve the state marking probability $P(M_i)$ by (3) with the help of MATLAB.

Finally, we compute the steady state probability of these being μ_i tokens in each place for any marking based on the unique solution of (3), and order them from largest to smallest by virtue of the classical BubbleSort algorithm.

Now, we present a case study to concretely illustrate the high-level efficiency and practicability of this approach.

4 Case Study

In this section, we use a commercial off-the-shelf (COTS) purchase processes in an enterprise to demonstrate our approach. Seven purchase process stages will be considered in our example, problem definition, the overall requirement specifications, the quantity and specification of components, seeking suppliers and requesting for proposal, the choice of suppliers, regular purchase and component performance evaluation.

In the following sections, we will give the corresponding TFD, SPN model, isomorphic Markov chain and reachable marking graph. The analysis results show that our approach efficiently helps to locate main factors that affect purchase processes.

4.1 TFD of COTS purchase process in an enterprise

In what follows, we use TFD to represent the specific purchase processes of our example. There are seven transactions according to the above definition. In TFD, one of the transactions (i.e., regular purchase, seeking suppliers and requesting for proposal) will be chosen to process according to the importance of review of quantity and specification of components. The TFD of purchase processes is shown in Fig.3.

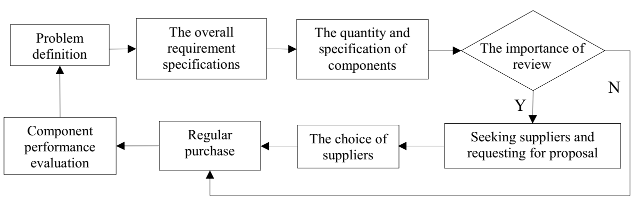

Fig. 3 TFD of COTS purchase process in an Enterprise

4.2 SPN model of COTS purchase process in an enterprise

According to the definition and modeling rules of SPN, the different purchase process stages can be viewed as different place elements, and transition denotes the changes of decision-making information acquisition ability in different moments, the value of place is either "0" or "1", where "0" denotes that the purchase phase information is completely unknown at the moment t, while "1" fully grasped. Different flow structures of purchase process correspond to different purchase progress situations. Then, we can establish a SPN model by virtue of the flow structures.

Let λ_i be the firing rate parameters of each transition. We say that the transition was fired in the average execution time $1/\lambda_i$, it means that, after the average execution time $1/\lambda_i$, the import place indicates that the enterprise obtains or masters the purchase phase information, while the export place indicates that the enterprise losses or unable to grasp the purchase phase information. In this way, each state marking of the SPN indicates that the purchase processes of the enterprise changes over time, and the state of overall activities corresponds to a state marking. Moreover, we can calculate the state marking probability in each place. Furthermore, by particularly applying the marking probabilities and the number of tokens in each place in a particular marking, for any marking, one can deduce the steady state probability of these being μ_i tokens in each place. Consequently, the values of these steady state probabilities can be used to reflect the ability of concrete information processing in the purchase phase, in which the bigger values correspond to the delayed activities.

Take the COTS purchase process in the enterprise as an example. Fig.4 shows the SPN model.

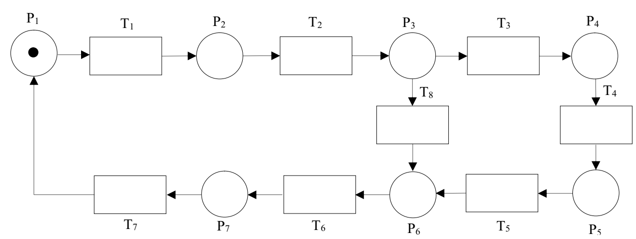

Fig. 4 The SPN model of COTS Purchase Process in an Enterprise

Table 1 The concrete meaning corresponding to each variable in Fig.4

Place	Meaning	Transition	Meaning
P_1	Problem information set	T_1	To measure the problem information
P_2	Requirement specification information set	T_2	To measure related requirements
P_3	The number and specification of components information set	T_3	To measure components in need of purchasing
P_4	List of suppliers and proposal information set	T_4	To filter namelist of suppliers
P_5	Supplier evaluation information set	T_5	To evaluate for the suppliers
P_6	Regular purchase information set	T_6	To measure the used component
P_7	Component evaluation information set	T_7	To remember the evaluation information
		T_8	To retrieve records of purchase information set

4.3 Isomorphic Markov chain and the reachable marking graph

In Fig.4, the firing rate of transitions (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8) obey negative exponential distribution. Just in order to reduce the complexity of locating the delayed activities in software processes, in this paper, we assume that, without loss of generality, the average execution time parameters have the values: $1/\lambda_1$=1, $1/\lambda_2$=1/2, $1/\lambda_3$=1/2, $1/\lambda_4$=1/3, $1/\lambda_5$=1, $1/\lambda_6$=1/3, $1/\lambda_7$=1/2, $1/\lambda_8$=1, which may be judged from the previous experience by software process domain experts. The Markov chain is shown in Fig.5, and the isomorphic reachable marking graph is shown in Table 2.

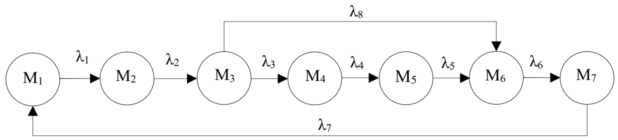

Fig. 5 Isomorphic Markov chain

Assuming an initial marking of one token in place P_1 and no tokens in the remaining places, then solving for the reachability set, we find seven states:

Table 2 The reachable marking graph corresponding to Fig.4

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
M_1	1	0	0	0	0	0	0
M_2	0	1	0	0	0	0	0
M_3	0	0	1	0	0	0	0
M_4	0	0	0	1	0	0	0
M_5	0	0	0	0	1	0	0
M_6	0	0	0	0	0	1	0
M_7	0	0	0	0	0	0	1

4.4 Calculating the steady probability

According to the SPN theory in section 2.1, we calculate the probability transfer matrix Q as

$$\begin{bmatrix} -\lambda_1 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\lambda_2 & \lambda_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\lambda_3-\lambda_8 & \lambda_3 & 0 & \lambda_8 & 0 \\ 0 & 0 & 0 & -\lambda_4 & \lambda_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\lambda_5 & \lambda_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\lambda_6 & \lambda_6 \\ \lambda_7 & 0 & 0 & 0 & 0 & 0 & -\lambda_7 \end{bmatrix},$$

which satisfies

$$H \cdot Q = \mathbf{0}, \tag{7}$$

where $H = \big(P(M_1),\ P(M_2),\ \dots,\ P(M_7)\big)$, and

$$\sum_{i=1}^{7} P(M_i) = 1.$$

We plug λ_1=1, λ_2=2, λ_3=2, λ_4=3, λ_5=1, λ_6=3, λ_7=2, λ_8=1 back into (7), then the steady probability of each state marking can be computed by the simultaneous linear equations

$$\begin{bmatrix} P(M_1) \\ P(M_2) \\ P(M_3) \\ P(M_4) \\ P(M_5) \\ P(M_6) \\ P(M_7) \end{bmatrix}^T \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -2 & 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -3 & 2 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & -3 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -3 & 3 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 & -2 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}^T. \tag{8}$$

By solving (8), we achieve the following steady-state marking probabilities:

Table 3 The steady-state marking probabilities

M_1	M_2	M_3	M_4	M_5	M_6	M_7
0.2813	0.1406	0.0938	0.0625	0.1875	0.0938	0.1406

Furthermore, making the most of the marking probabilities and the number of tokens in each place in a particular marking, one can easily deduce the steady state probability of these being μ_i tokens in each place for any marking. The precise token probability density functions are calculated as follows:

Table 4(a) The token probability density functions

$P(\mu_1{=}0)$	$P(\mu_2{=}0)$	$P(\mu_3{=}0)$	$P(\mu_4{=}0)$	$P(\mu_5{=}0)$	$P(\mu_6{=}0)$	$P(\mu_7{=}0)$
0.7187	0.8594	0.9062	0.9375	0.8125	0.9062	0.8594

Table 4(b) The token probability density functions

$P(\mu_1{=}1)$	$P(\mu_2{=}1)$	$P(\mu_3{=}1)$	$P(\mu_4{=}1)$	$P(\mu_5{=}1)$	$P(\mu_6{=}1)$	$P(\mu_7{=}1)$
0.2813	0.1406	0.0938	0.0625	0.1875	0.0938	0.1406

4.5 Results analysis

The results in section 4.4 show that:

(1) The value $P(\mu_4 = 1)$ is minimum, it means that the suppliers have strong information collection abilities in the process stage, and the work of purchase bidding is highly efficient.

(2) The values $P(\mu_3 = 1)$, $P(\mu_6 = 1)$ are quite small, which indicates that the enterprise has more ability in measuring components needed to be purchased, retrieving records of purchase information set and measuring the used component.

(3) The value $P(\mu_1 = 1)$ is maximum, which indicates that the occurrence probability of the enterprise is maximum. Hence, the measuring problem information stages are delayed activities.

The probability values place $P(\mu_1 = 1)$ and $P(\mu_5 = 1)$ are larger than others, it implies that the enterprise has less ability in measuring the problem information and evaluating for the suppliers. The enterprise should intervene actively in these two activities such that the ability will be enhanced in collecting and analyzing to improve these activities.

The result also shows that measuring the problem information and evaluating for the suppliers are the main factors that affect COTS purchase process in this enterprise, and that improving some activities can be more conducive for the enterprise. For example, it is of significance to train-up the staffs so as to improve their abilities of measuring problems information and analyzing assessment information. Besides, hiring some experts to analyze the problems and mine parallel activities is helpful as well.

The above discussion indicates that, only when probabilities of the places with tokens, which correspond to the delayed activities with bigger occupation probabilities of resource in average execution time, are efficiently calculated can we take some useful measures to improve those activities by the approach proposed. In a word, it is practical and rational to use our approach in software processes.

5 Conclusions

Activity is a core element in software process. In order to improve software process effectively, this paper proposes a new approach to explain how to locate the delayed activities in software processes relied on the related theory of SPN, TFD and Markov chain. The principles of this approach are firstly introduced and the framework of locating delayed activities is proposed. Then, we perform an algorithm on calculating probabilities of the places with tokens and prove it briefly. Finally, a case study is provided to show the high-level efficiency of the approach.

This approach might be helpful for software developers to locate the delayed activities in software development process so that some effective measures could be taken to improve these delayed activities as far as possible. Moreover, it might be also beneficial for project managers to locate delayed activities in manufacturing phase in order to shorten the project cycle.

In the future work, on the one hand, this approach can be applied to more fields, such as economics, biology, agriculture, social science and so on, for locating the delayed activities. On the other hand, we will try our best to perform some high-efficiency algorithms to calculate those values of the average execution time of activities precisely, instead of judging from the previous experience. Furthermore, we will make efforts to propose some more effective approaches to locate the delayed activities in software processes.

References

[1] L.J. Osterweil. Software Processes are Software Too. In *Proceedings of IEEE* 9^{th} *Software Engineering*, IEEE CS, Monterey, USA, pp. 2–13, 1987.

[2] R. Singh. International Standard ISO/IEC 12207 software life cycle processes. *Software Process Improvement and Practice*, vol. 2, no. 1, pp. 35–50, 1996.

[3] T. Li. *An Approach to Modelling Software Evolution Processes*, Berlin, Germany: Springer-Verlag, pp. 9, 2008.

[4] R.S. Pressman. *Software engineering: a practitioner's approach*, New York, America: McGraw Hill, 2000.

[5] C.A. Petri. Kommunikation mit Automaten, Ph. D. dissertation, Darmstadt University of Technology, Germany, 1962; translation: C.F. Greene. Supplement I to Tech. Report RADC-TR-65-337, Vol.1, Rome Air Development Center; Grifiss Air Force Base, New York, 1965.

[6] W.M. Van der Aalst. The application of Petri nets to workflow management. *Journal of circuits, systems, and computers*, vol. 8, no. 1, pp. 21–66, 1998.

[7] W.M. Van Der Aalst and A.H. Ter Hofstede. Verification of workflow task structures: A petri-net-baset approach. *Information systems*, vol. 25, no. 1, pp.43–69, 2000.

[8] R. Hamadi and B. Benatallah. A Petri net-based model for web service composition. In *Proceedings of ACS* 14^{th} *Australasian database conference*, Australian Computer Society Incorporated, Darlinghurst, Australia, pp. 191–200, 2003.

[9] J.D. Ge, H. Hu, Q. Gu, and J. Lü. Modelling Multi-View Software Process with Object Petri Nets. In *Proceedings of IEEE* 6^{th} *Software Engineering Advances*, IEEE, Tahiti, French, pp. 41–46, 2006.

[10] M.K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 913–917, 1982.

[11] B. Barbot and M. Kwiatkowska. On Quantitative Modelling and Verification of DNA Walker Circuits Using Stochastic Petri Nets. *Application and Theory of Petri Nets and Concurrency*, Springer International Publishing, pp. 1–32, 2015.

[12] Y.M. Han, X.L. Wu, C.Y. Yue. Model of software process and Monte-Carlo simulation analysis based on SPN. *Journal of Huazhong University of Science and Technology(Nature Science Edition)*, vol. 31, no. 7, pp. 37–39, 2003. (in Chinese with English Abstract)

[13] M. Ajmone Marsan, G. Conte and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, vol. 2, no. 2, pp. 93–122, 1984.

[14] L. Lei, Y. Zhang, X. Shen, C. Lin and Z. Zhong. Performance analysis of device-to-device communications with dynamic interference using stochastic Petri nets. *IEEE Transactions on Wireless Communications*, vol. 12, no. 12, pp. 6121–6141, 2013.

[15] Y.X. Dong, Y.N. Xia, Q.S. Zhu and Y. Huang. A stochastic approach to predict performance of web service composition. In *Proceedings of IEEE* 2^{nd} *Symposium on Electronic Commerce and Security*, IEEE CS, Nanchang, China, pp. 460–464, 2009.

[16] G.J. Shan, G.J. Wang, Y.Q. Dai and Y.Z. Wang. Performance Analysis of The Vehicular 1553B Bus System Using Stochastic Petri Net. In *Proceedings of IEEE* 3^{rd} *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, IEEE, Emeishan, China, pp. 405–408, 2013.

[17] L. Jiao. The Research Based on the Transfer in Organizational Buying Process to Seeking for Core Opinion Leader. *Economic Research Guide*, no. 95, pp. 178–181, 2010. (in Chinese with English Abstract)

[18] M. Molloy. On the integration of delay and throughput measures in distributed processing models, Ph. D. dissertation, University of California, USA, 1981.

[19] C. Lin. *Introduction to stochastic Petri-nets and system performance*, Beijing, China: Tsinghua University Press, 2005. (in Chinese)

[20] Z.H. Wu. *An introduction to Petri-nets*, Beijing, China: China Machine Press, 2006. (in Chinese)

[21] C.Y. Yuan. *The principle and application of Petri-nets*, Beijing, China: Publishing House of Electronics Industry, 2005. (in Chinese)

[22] J.G. Kemeny, H. Mirkill, J.L. Snell, and G.L. Thompson. Finite mathematical structures. *Journal of Symbolic Logic (Review)*, Prentice-Hall, New York, vol. 24, no. 3, pp. 221–222, 1959.

[23] C.M. Grinstead and J.L. Snell. *Introduction to probability*, New York, America: American Mathematical Society, 2012.

[24] R.A. Howard. *Dynamic Probabilistic Systems*, New York, America: John Wiley and Sons, 1971.

[25] G.G. Judge and E.R. Swanson. Markov chains: basic concepts and suggested uses in agricultural economics. *Australian Journal of Agricultural Economics*, vol. 6, no. 2, pp. 49–61, 1962.

[26] B.W. Jiang, L.H. Zhang, H.C. Lu, C. Yang and M.H. Yang. Saliency detection via absorbing Markov chain. In *Proceedings of IEEE* 14^{th} *International Conference on Computer Vision*, IEEE CS, Sydney, Australia, pp. 1665-1672, 2013.

[27] F.P. Kelly. *Reversibility and stochastic networks*, New York, America: Wiley Press, 1979.

[28] P.J.B. King and I. Mitrani. Numerical methods for infinite Markov processes. In *Proceedings of ACM SIGMETRICS Performance Evaluation Review*, ACM, pp. 277–282, 1980.

Algorithm 1 Calculating for probabilities of the places with tokens

Input: The average execution time parameters $(1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_n)$
Output: The values of steady state probability of these being μ_i tokens in each place for any marking

Begin
Dim $\mathbf{Q} = [q_{ij}]_{r\times r} \leftarrow \mathbf{0}$, $\prod = [\pi_1, \pi_2, \ldots, \pi_r] \leftarrow \mathbf{0}$, $P(M_i) \leftarrow 0$,
    $P(\mu_i = 1) \leftarrow 0$, $s \leftarrow 0$, $i \leftarrow 0$, $j \leftarrow 0$, $r \leftarrow 0$, $m \leftarrow 0$
$\triangleright$ Establish Probability Transfer Matrix $\mathbf{Q} = [q_{ij}]_{r\times r}$ as shown in (1), (2)
For i *from* 1 *to* r
  For j *from* 1 *to* r
    If $i \neq j$ *and* $M_i[t_k > M_j$ $(\exists t_k \in T)$ then
      $q_{ij} \leftarrow \lambda_k$
    Else $i = j$ then
      For h *from* 1 *to* r
        If $M_i[t_k > M_h(\exists t_k \in T)$ then
          $s \leftarrow s\text{-}\lambda_k$
        End if
      End for
      $q_{ij} \leftarrow s$
    Else
      $q_{ij} \leftarrow 0$
    End if
  End for
End for
Read $\lambda_1, \lambda_2, \ldots, \lambda_n$ $\triangleright$ Input values of parameters $\lambda_1, \lambda_2, \ldots, \lambda_n$
Solve the equations $\begin{cases} \prod Q = \mathbf{0}, \\ \sum_{i=1}^{r} \pi_i = 1, \end{cases}$ (3)with the help of MATLAB to obtain π_i, namely, the state marking probability P(M_i), and then output them.
$\triangleright$ Deduce the steady state probability of these being μ_i tokens in each place for any marking
For i *from* 1 *to* r
  $P(M_i) \leftarrow \pi_i$
  Print $P(M_i)$
End for
For i *from* 1 *to* m $\triangleright$ the loop of place P_m
  For j *from* 1 *to* r $\triangleright$ the loop of marking M_r
    If place P_i have one token in marking M_j then
      $P(\mu_i = 1) \leftarrow P(\mu_i = 1) + P(M_j)$
    End if
  End for
End for
Dim $a[m+1] \leftarrow \mathbf{0}, b[m+1] \leftarrow \mathbf{0}, c \leftarrow 0, d \leftarrow 0$
For i *from* 1 *to* m
  $a[i] \leftarrow P(\mu_i = 1)$ $\triangleright$ save values of probabilities
  $b[i] \leftarrow i$ $\triangleright$ save numerical order
End for
$\triangleright$ Order the values of steady state probability of these being μ_i tokens in each place for any marking from largest to smallest
For j *from* 1 *to* m
  For i *from* m *to* j
    IF $a[i] > a[i-1]$ then
      $c \leftarrow a[i-1]$
      $d \leftarrow (i-1)$
      $a[i-1] \leftarrow a[i]$
      $b[i-1] \leftarrow b[i]$
      $a[i] \leftarrow c$
      $b[i] \leftarrow d$
    End if
  End for
End for
$\triangleright$ Output the values of steady state probability of these being μ_i tokens in each place for any marking from largest to smallest
For i *from* 1 *to* m
  Print $P(\mu_{b[i]} = 1)$: $a[i]$
End for
End.

Yunzhi Jin received his M. Sc. degree in System Analysis and Integration of Yunnan University in 2013. Currently, he is a Ph. D. candidate in the Research Center of Cloud Computing of Yunnan Province, Yunnan University. His research interests include Software Engineering, System Analysis and Integration, Web and Distributed Computing.

E-mail: jyzynu@163.com
ORCID iD: 0000-0001-7355-1629

Hua Zhou received his B. Sc. and M. Sc. degrees in computer from the Jilin University, China, in 1987 and 1990, respectively, and the Ph. D. degree in software engineering from De Montfort University, UK in 2004. In 1984, he was a faculty member at Yunnan University, China. Currently, he is a professor in School Of Software at Yunnan University, China. He has published about 60 refereed journal and conference papers.

His research interest covers Software Engineering, System Analysis and Integration, Web and Distributed Computing.

E-mail: hzhou@ynu.edu.cn (Corresponding author)

Hongji Yang received his B. Sc. and M. Sc. degrees in computer from the Jilin University, China, in 1982 and 1985, respectively, and the Ph. D. degree in computing from Durham University, UK in 1994. In 1985, he was a faculty member at Jilin University, China, in 1989 at Durham University, UK, in 1993 at De Montfort University, UK and in 2013 at Bath Spa University, UK. Currently, he is a professor in School Of Humanities And Cultural Industries at Bath Spa University, UK.

He has published about 400 refereed journal and conference papers. His research interest covers Software Engineering, Creative Computing, Web and Distributed Computing.

Prof. Yang has become IEEE Computer Society Golden Core Member since 2010, also, he is a member of EPSRC Peer Review College since 2003. He is the Editor in Chief of International Journal of Creative Computing, InderScience.

E-mail: h.yang@bathspa.ac.uk

Sijing Zhang obtained his B. Sc. and M. Sc. degrees, both in Computer Science, from Jilin University, Changchun, China in 1982 and 1988, respectively. He earned a Ph. D degree in Computer Science from the University of York, UK in 1996. He then joined the Network Technology Research Centre (NTRC) of Nanyang Technological University, Singapore as a post-doctoral fellow. In 1998, he returned to the UK to work as a research fellow with the Centre for Communications Systems Research (CCSR) of the University of Cambridge. He joined the School of Computing and Technology, University of Derby, UK, as a senior lecturer in 2000. Since October 2004, he has been working as a senior lecturer in the Department of Computer Science and Technology, University of Bedfordshire, UK.

His research interests include wireless networking, data communications, schedulability tests for hard real-time traffic, performance analysis and evaluation of real-time communication protocols, QoS provision, vehicular ad hoc networks, and wireless networks for real-time industrial applications.

E-mail: Sijing.Zhang@beds.ac.uk

Jidong Ge obtained his Ph. D degree in Computer Science from Institute of Computer Software at CS Department of Nanjing University, China in 2007. Currently, he is an Associate Professor in Software Institute, Nanjing University.

His research interests include Software Engineering, Workflow, Process Mining, Petri nets, Distributed Computing, Cloud Computing, Big data, Services Computing, Software Architecture, Inheritance of Behavior, Formal Methods, Software Process, Formal Verification, Model Checking, UML, Mobile Agents, etc.

E-mail: gjd@nju.edu.cn