International Journal of Automation and Computing 15(1), February 2018, 115-124
DOI: 10.1007/s11633-017-1092-9

# An Approach to Locating Delayed Activities in Software Processes

Yun-Zhi Jin[1] Hua Zhou[1,2,3] Hong-Ji Yang[4] Si-Jing Zhang[5] Ji-Dong 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, Corsham, SN13 0BZ, UK
[5]Department of Computer Science and Technology, University of Bedfordshire, LU1 3JU, UK
[6]State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, Nanjing 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 resource occupation probabilities with respect to 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 random fields, metrics.

## 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 (international organisation for standardization)/IEC (international electrotechnical commission) 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 also stated in [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 in [4], "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. The motivation for our research is to help software processes to be more successful in terms of time cost. According to Wu et al.[5], the problem of delay often unavoidably exists in every aspect of engineering systems, such as chemical reactor systems, cardiovascular-respiratory control systems and networked control systems[6]. Specifically, this problem is no exception in software process. In most of the cases, some activities that own bigger resource occupation probabilities with respect to 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 Dr. Petri[7] in Germany in 1962 which 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[8] introduced workflow management as an application domain for Petri-nets, proposed state-of-the-art results with respect

Research Article
Manuscript received June 10, 2016; accepted May 2, 2017; published online September 21, 2017
This work was supported by National Natural Science Foundation of China (No. 61462091), High-tech Industrial Development Program of Yunnan Province (No. 1956, in 2012), New Academic Researcher Award for Doctoral Candidates of Yunnan Province of China (No. ynu201414), Natural Science Youth Foundation of Yunnan Province of China (No. 2014FD006), and the Postgraduates Science Foundation of Yunnan University (No. ynuy201424).
Recommended by Associate Editor Xun Chen

to the verification of workflows, and highlighted some Petri-net-based workflow tools. Likewise, Van der Aalst and Ter Hofstede[9] 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[10] put forward a Petri-net-based algebra, used to model control flows, as a necessary constituent of reliable web service composition process. We can also see Ge et al.[11] 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[12] discussed the isomorphism between the behaviour of Petri-nets with exponentially distributed transition rates and the Markov processes. Furthermore, Barbot and Kwiatkowska[13] 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.[14] 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 suitable for time performance evaluation of system. For instance, Marsan et al.[15] proposed a class of generalised stochastic Petri nets (GSPNs) for the performance evaluation of multiprocessor systems. Furthermore, relying on the SPN, Lei et al.[16] analyzed the performance of device-to-device (D2D) communications with dynamic interference. Similarly, Dong et al.[17] employed an approach to predicting the performance of web service composition. Shan et al.[18] constructed a formalized model of vehicular 1 553 B bus system and then analysed the performance of the vehicular 1 553 B 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 certain measures such as increasing resources, excavating and executing some parallel 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, based on SPN, Jiao[19] presented an example in economics field to locate the core opinion leader, which can be reflected by the values of probabilities of the places with tokens. Inspired by this, we in this paper locate the 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 recent years, the most closely related work to our research is that of [20], they proposed an approach to tailoring software processes from software processes lines and information about the characteristics of projects. In their work, the prioritization algorithm based on analytic hierarchy process (AHP) method is used to select the most suitable process elements (e.g., activities) to meet the requirements of tailoring and guide the process engineer to choose the best elements to be added in the tailored process. By means of this algorithm, the numerical probability of each alternative elements is calculated and the higher the probability, the better chances the alternative will be selected and added in the tailored software process line. Different from their work, our research aims to locate the delayed activities by its values of probabilities of the places with tokens based on SPN in software processes before improving them.

Motivated by the above observation, 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 places 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 [21], 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)[22−24] model.

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

Suppose that $T = \{t_1, t_2, \cdots, t_n\}, t_i \in T$, $\lambda(t_i) = \lambda_i$ are nonnegative real values, which represent the occur-

rence rates of the transition $t_i$ (meeting the conditions of an occurrence). When the transition $t_i$ is fired, the corresponding time delay $d_i$ is random variable satisfying $d_i(\tau) = \mathrm{e}^{-\lambda_i \tau}$, where $\tau$ is the related time. Therefore, the average time delay $\overline{d_i}$ of transition $t_i$ is determined by $\overline{d_i} = \int_0^\infty \mathrm{e}^{-\lambda_i \tau} \mathrm{d}\tau = \frac{1}{\lambda_i}$.

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

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

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

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 $\Sigma$) in the Markov chain. In general, $Q$ is a $r$-dimensional vector $\prod = [\pi_1, \pi_2, \cdots, \pi_r](r = |R(M_0)|)$, where $\pi_i$ denote the steady probabilities of marking $M_i$. Here, the $r$-dimensional vector $\prod$ satisfy the following equations:

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

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 satisfying some special conditions can be worked out.

Let $B$ be 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 $\prod = [\ \pi_1, \pi_2, \cdots, \pi_r\ ]$ as

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

where $\pi_i$ is the steady probability 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 worth 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 successful system analysis by 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. Furthermore, 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 do not use notations of Begin and End for convenience, especially when as a TFD is considered cyclic.

## 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, i.e., 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[25, 26]: We have a set of states $S = \{s_1, s_2, \cdots, s_r\}$, 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 probability $p_{ij}$ is called transition probability, 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

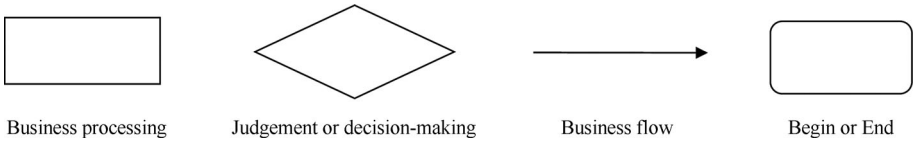



Fig. 1 Fundamental notations of TFD

probability distribution, defined on $S$, 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 $P$ with $p_{ij}$ to completely specify the Markov chain.

Howard[27] 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. Furthermore, 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, in control theory[28], and in many other 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 Judge and Swanson[29]. Moreover, Jiang et al.[30] formulated saliency detection via absorbing Markov chain on an image graph model.

# 3 Main idea

In this paper, we adopt two research methods (i.e., quantitative analysis method and validation method). On one hand, we design an algorithm to perform a quantitative analysis for calculating probabilities of the places with tokens. On the other hand, the research is validated by means of a case study. 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 $\frac{1}{\lambda_i}$ is the average execution time. For each activity, we calculate the 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 short, 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, 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.

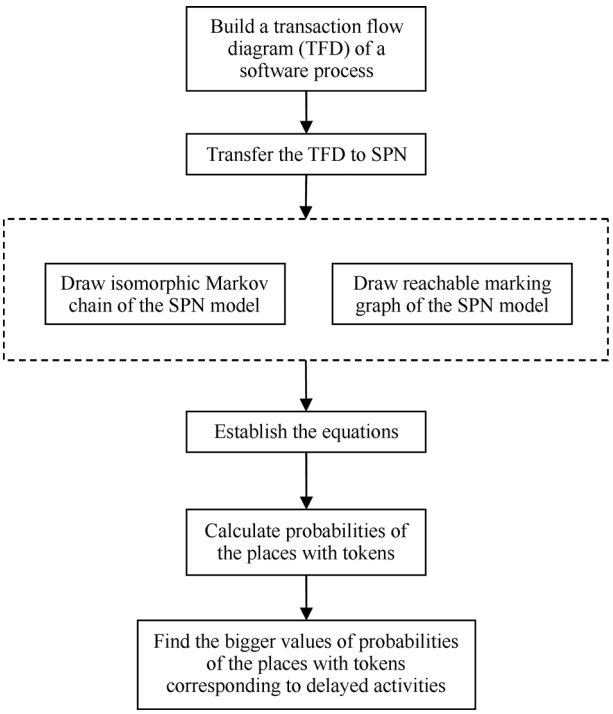



Fig. 2 Framework of locating delayed activities

This approach is based on a series of classical methods, say, stochastic Petri-nets, transaction flow diagram and Markov chain. The first step of the approach is to build a transaction flow diagram of a software process and then transfer it to SPN. Additionally, we draw isomorphic Markov chain and reachable marking graph of the SPN model. Moreover, we establish (3). Finally, by calculating the probability transfer matrix of the Markov chain (MC) (the algorithm and its correctness is given in Section 3.3), 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 with $\mu_i$ tokens in each place correspond to the delayed

activities in the software process. Here

$$\mu_i = \begin{cases} 0, & \text{if place } P_i \text{ has no token in every marking} \\ 1, & \text{otherwise.} \end{cases}$$

Assume that the Markov chain with $r$ markings ($M_1$, $M_2, \cdots, M_r$) and reachable marking graph of SPN with $m$ places ($P_1, P_2, \cdots, P_m$) have been achieved, then the probabilities of the places with tokens $P(\mu_i = 1)$ can be calculated by 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 variables is 1. Then, we show that the solution of (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 [31] 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:

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

By Lemma 1, one can achieve $r$ homogeneous equations

$$\Big(\sum_v \lambda_v\Big) x_i - \Big(\sum_u \lambda_u x_u\Big) = 0 \ (i = 1, 2, \cdots, r) \quad (4)$$

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

Equations (4) can be denoted as

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

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

Noting that the probability distribution of discrete stochastic variables satisfies

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

and that the states of continuous time Markov processes are discrete, it follows that

$$\sum_i = 1^r x_i = 1. \quad (6)$$

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 $\prod = [\pi_1, \pi_2, \cdots, \pi_r]$ is the unique solution[32] 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 with $\mu_i$ 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

With the booming of commercial off-the-shelf (COTS), the increasing number of software components with open standard interfaces are available in the commercial market. In order to reduce costs and shorten development time, a number of enterprises prefer to purchase COTS software components rather than design them by themselves.

In this paper, we improve a real case of the COTS purchase process in an enterprise and then use the improved case of a COTS purchase process 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.

### 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 $\lambda_i$ be the firing rate parameters of each transition. We say that the transition was fired in the average execution time $\frac{1}{\lambda_i}$, it means that, after the average execution time $\frac{1}{\lambda_i}$, the import place indicates that the enterprise obtains or masters the purchase phase information, while the export place indicates that the enterprise loses or unable to grasp the purchase phase information. In this way, each

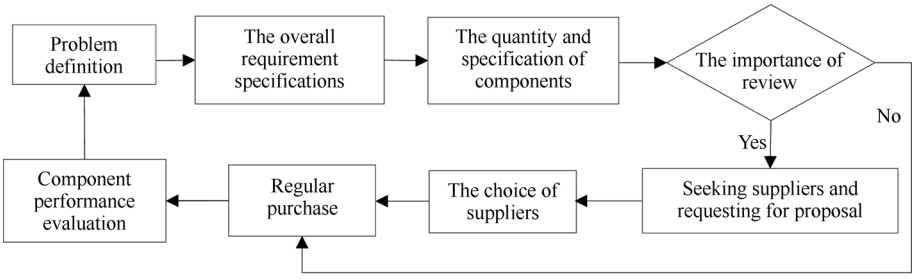



Fig. 3 TFD of COTS purchase process in an enterprise

Table 1 Concrete meaning corresponding to each variable in Fig. 4

| Place | Meaning | Transition | Meaning |
|---|---|---|---|
| $P_1$ | Problem information set | $T_1$ | To analyse the problem information |
| $P_2$ | Requirement specification information set | $T_2$ | To understand the related requirements |
| $P_3$ | Number and specification of components information set | $T_3$ | To assess the components needed to be purchased |
| $P_4$ | List of suppliers and proposal information set | $T_4$ | To filter namelist of suppliers |
| $P_5$ | Supplier evaluation information set | $T_5$ | To evaluate the suppliers |
| $P_6$ | Regular purchase information set | $T_6$ | To evaluate 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 |

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 for 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 with $\mu_i$ 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. Table 1 shows the concrete meaning corresponding to each variable in Fig. 4.

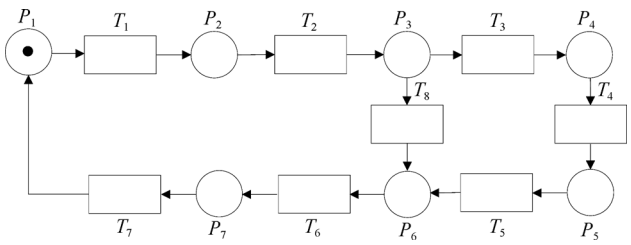



Fig. 4 SPN model of COTS purchase process in an enterprise

## 4.3 Isomorphic Markov chain and the reachable marking graph

In Fig. 4, the firing rate of transitions ($T_1$, $T_2$, $T_3$, $T_4$, $T_5$, $T_6$, $T_7$, $T_8$) obeys 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: $\frac{1}{\lambda_1}=1$, $\frac{1}{\lambda_2}=\frac{1}{2}$, $\frac{1}{\lambda_3}=\frac{1}{2}$, $\frac{1}{\lambda_4}=\frac{1}{3}$, $\frac{1}{\lambda_5}=1$, $\frac{1}{\lambda_6}=\frac{1}{3}$, $\frac{1}{\lambda_7}=\frac{1}{2}$, $\frac{1}{\lambda_8}=1$, 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.

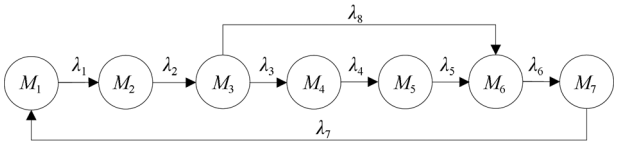



Fig. 5 Isomorphic Markov chain

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 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 state 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

$$H \times Q = 0 \tag{7}$$

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

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

We plug $\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 2$, $\lambda_4 = 3$, $\lambda_5 = 1$, $\lambda_6 = 3$, $\lambda_7 = 2$, $\lambda_8 = 1$ back into (7), then the steady state probability of each state marking can be computed by the simultaneous linear equations

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

By solving (8), we achieve the following steady-state marking probabilities in Table 3.

Table 3 Steady-state marking probabilities

| $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ |
|---|---|---|---|---|---|---|
| 0.281 3 | 0.140 6 | 0.093 8 | 0.062 5 | 0.187 5 | 0.093 8 | 0.140 6 |

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 with $\mu_i$ tokens in each place for any marking. The precise token probability density functions are calculated as follows in Tables 4(a) and 4(b).

Table 4 (a) Token probability density functions

| $P(\mu_1{=}0)$ | $P(\mu_2{=}0)$ | $P(\mu_3{=}0)$ | $P(\mu_4{=}0)$ | $P(\mu_5{=}0)$ | $P(\mu_6{=}0)$ | $P(\mu_7{=}0)$ |
|---|---|---|---|---|---|---|
| 0.718 7 | 0.859 4 | 0.906 2 | 0.937 5 | 0.812 5 | 0.906 2 | 0.859 4 |

Table 4 (b) Token probability density functions

| $P(\mu_1{=}1)$ | $P(\mu_2{=}1)$ | $P(\mu_3{=}1)$ | $P(\mu_4{=}1)$ | $P(\mu_5{=}1)$ | $P(\mu_6{=}1)$ | $P(\mu_7{=}1)$ |
|---|---|---|---|---|---|---|
| 0.281 3 | 0.140 6 | 0.093 8 | 0.062 5 | 0.187 5 | 0.093 8 | 0.140 6 |

## 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 enterprise has strong supplier information collection abilities, and the purchasing and bidding work is highly efficient in the filtering namelist of suppliers stage.

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

3) The value $P(\mu_1 = 1)$ is maximum, which indicates that the occurrence probability of problem information set in the COTS purchase process is maximum. Hence, the analyzing problem information stage is delayed activity.

The probability values place $P(\mu_1 = 1)$ and $P(\mu_5 = 1)$ are larger than the others, it implies that the enterprise has less ability in analyzing the problem information and evaluating 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 analyzing the problem information and evaluating 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 analyzing problems information and assessment information. Besides, hiring some experts to analyze the problems and mine parallel activities are helpful as well.

The above discussion indicates that, only when probabilities of the places with tokens corresponding to the delayed

activities are efficiently calculated, we can take some useful measures to improve those activities by the proposed approach. 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 explaining how to locate the delayed activities in software processes by relying 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.

Due to limited space in this paper, we have considered only the SPN in which the firing rate of transitions obeys negative exponential distribution. Besides, this approach requires those values of the average execution time of activities which may be judged from the previous experience by software process domain experts as input parameters in our proposed algorithm. Even so, the approach might be helpful for software developers to locate the delayed activities in the 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, on 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, and 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 more effective approaches to locate the delayed activities in software processes.

**Algorithm 1.** Calculation of probabilities of the places with tokens

**Input:** The average execution time parameters $(\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \cdots, \frac{1}{\lambda_n})$

**Output:** The values of steady state probability with $\mu_i$ tokens in each place for any marking

1) Begin
2) Dim $Q = [q_{ij}]_{r\times r} \leftarrow 0$, $\prod = [\pi_1,\ \pi_2,\ \cdots,\ \pi_r] \leftarrow 0$, $P(M_i) \leftarrow 0$,$P(\mu_i = 1) \leftarrow 0$, $s \leftarrow 0$, $i \leftarrow 0$, $j \leftarrow 0$, $r \leftarrow 0$, $m \leftarrow 0$
   /* Establish probability transfer Matrix $Q = [q_{ij}]_{r\times r}$ as shown in (1), (2) */
3) For $i$ from 1 to $r$
4)  For $j$ from 1 to $r$
5)  If $i \neq j$ and $M_i[t_k > M_j\ (\exists\, t_k \in T)$, then
6)  $q_{ij} \leftarrow \lambda_k$
7)  Elseif $i = j$, then
8)  For $h$ from 1 to $r$
9)  If $M_i[t_k > M_h (\exists\, t_k \in T)$, then
10)  $s \leftarrow s$-$\lambda_k$
11)  End if
12)  End for
13)  $q_{ij} \leftarrow s$
14)  Else
15)  $q_{ij} \leftarrow 0$
16)  End if
17)  End for
18) End for
19) Read $\lambda_1, \lambda_2, \cdots, \lambda_n$
   /* Input values of parameters $\lambda_1, \lambda_2, \cdots, \lambda_n$ */
20) Solve the equations $\begin{cases} \prod Q = 0 \\ \sum\limits_{i=1}^{r} \pi_i = 1 \end{cases}$ with the help of Matlab to obtain $\pi_i$, namely, the state marking probability $P(M_i)$, and then output them.
   /* Deduce the steady state probability with $\mu_i$ tokens in each place for any marking */
21) For $i$ from 1 to $r$
22)  $P(M_i) \leftarrow \pi_i$
23)  Print $P(M_i)$
24) End for
25) For $i$ from 1 to $m$  /* Loop of place $P_m$ */
26)  For $j$ from 1 to $r$  /* Loop of marking $M_r$ */
27)  If place $P_i$ has one token in marking $M_j$, then
28)  $P(\mu_i = 1) \leftarrow P(\mu_i = 1) + P(M_j)$
29)  End if
30)  End for
31) End for
32) Dim $a[m+1] \leftarrow \mathbf{0}, b[m+1] \leftarrow \mathbf{0}, c \leftarrow 0, d \leftarrow 0$
33) For $i$ from 1 to $m$
34)  $a[i] \leftarrow P(\mu_i = 1)$  /* Save values of probabilities */
35)  $b[i] \leftarrow i$  /* Save numerical order */
36) End for
   /* Order the values of steady state probability with $\mu_i$ tokens in each place for any marking from largest to smallest */
37) For $j$ from 1 to $m$
38)  For $i$ from $m$ to $j$
39)  If $a[i] > a[i-1]$, then
40)  $c \leftarrow a[i-1]$
41)  $d \leftarrow (i-1)$
42)  $a[i-1] \leftarrow a[i]$
43)  $b[i-1] \leftarrow b[i]$
44)  $a[i] \leftarrow c$
45)  $b[i] \leftarrow d$
46)  End if
47)  End for
48) End for
   /* Output the values of steady state probability with $\mu_i$ tokens in each place for any marking from largest to smallest */
49) For $i$ from 1 to $m$
50)  Print $P(\mu_{b[i]} = 1)$: $a[i]$

51) End for
52) End

## References


[1] L. J. Osterweil. Software processes are software too. In *Proceedings of the 9th International Conference on Software Engineering*, IEEE 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 Heidelberg, Germany: Springer-Verlag, pp. 9, 2009.

[4] R. S. Pressman. *Software Engineering: A Practitioner's Approach*, New York, USA: McGraw Hill, 2000.

[5] X. J. Wu, X. L. Wu, X. Y. Luo. Adaptive neural network dynamic surface control for a class of nonlinear systems with uncertain time delays. *International Journal of Automation and Computing*, vol. 13, no. 4, pp. 409–416, 2016.

[6] Y. Ge, Y. Li. SCHMM-based compensation for the random delays in networked control systems. *International Journal of Automation and Computing*, vol. 13, no. 6, pp. 643–652, 2016.

[7] C. A. Petri. Kommunikation mit Automaten, Ph. D. dissertation, University of Bonn, Germany, 1962.

[8] W. M. P. 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.

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

[10] R. Hamadi, B. Benatallah. A Petri-net-based model for web service composition. In *Proceedings of the 14th Australasian Database Conference*, Adelaide, Australia, pp. 191–200, 2003.

[11] J. D. Ge, H. Hu, Q. Gu, J. Lu. Modeling multi-view software process with object Petri nets. In *Proceedings of International Conference on Software Engineering Advances*, Tahiti, France, 2006.

[12] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, vol. c-31, no. 9, pp. 913–917, 1982.

[13] B. Barbot, M. Kwiatkowska. On quantitative modelling and verification of DNA walker circuits using stochastic Petri nets. *Application and Theory of Petri Nets and Concurrency*, R. Devillers, A. Valmari, Eds., Cham: Springer, pp. 1–32, 2015.

[14] 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)

[15] M. A. Marsan, G. Conte, 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.

[16] L. Lei, Y. K. Zhang, X. M. Shen, C. Lin, Z. D. 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.

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

[18] G. J. Shan, G. J. Wang, Y. Q. Dai, Y. Z. Wang. Performance analysis of the vehicular 1553B bus system using stochastic Petri net. In *Proceedings of International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, Chengdu, China, pp. 405–408, 2013.

[19] L. Jiao. The research based on the transfer in organizational buying process to seeking for core opinion leader. *Economic Research Guide*, no. 21, pp. 178–181, 2010. (in Chinese)

[20] W. G. Lorenz, M. B. Brasil, L. M. Fontoura, G. V. Pereira. Activity-based software process lines tailoring. *International Journal of Software Engineering and Knowledge Engineering*, vol. 24, no. 9, pp. 1357, 2014.

[21] M. K. Molloy. On the Integration of Delay and Throughput Measures in Distributed Processing Models, Ph. D. dissertation, University of California, USA, 1981.

[22] C. Lin. *Introduction to Stochastic Petri-nets and System Performance*, 2nd ed., Beijing, China: Tsinghua University Press, 2005. (in Chinese)

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

[24] C. Y. Yuan. *The Principle and Application of Petri Nets*, Beijing, China: Publishing House of Electronics Industry, 2005. (in Chinese)

[25] J. G. Kemeny, H. Mirkil, J. L. Snell, G. L. Thompson. *Finite Mathematical Structures*. New York, USA: Prentice-Hall, 1959.

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

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

[28] A. Gosavi, A. Parulekar. Solving Markov decision processes with downside risk adjustment. *International Journal of Automation and Computing*, vol. 13, no. 3, pp. 235–245, 2016.

[29] G. G. Judge, 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.

[30] B. W. Jiang, L. H. Zhang, H. C. Lu, C. Yang, M. H. Yang. Saliency detection via absorbing Markov chain. In *Proceedings of IEEE International Conference on Computer Vision*, Sydney, Australia, pp. 1665–1672, 2013.

[31] F. P. Kelly. *Reversibility and Stochastic Networks*, New York, USA: Wiley Press, 1979.

[32] P. J. B. King, I. Mitrani. Numerical methods for infinite Markov processes. In *Proceedings of International Symposium on Computer performance Modelling, measurement and evaluation*, Toronto, Ontario, Canada, pp. 277–282, 1980.

**Yun-Zhi Jin** received the M. Sc. degree in system analysis and integration from Yunnan University, China in 2013. Currently, he is a Ph. D. degree candidate in the Research Center of Cloud Computing of Yunnan Province, Yunnan University, China.

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 the B. Sc. and M. Sc. degrees in computer from the Jilin University, China in 1987 and 1990, respectively, and received 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.

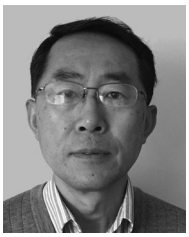
His research interests include software engineering, system analysis and integration, web and distributed computing.

E-mail: hzhou@ynu.edu.cn (Corresponding author)
ORCID iD: 0000-0001-9381-0827

**Hong-Ji Yang** received the B. Sc. and M. Sc. degrees in computer from the Jilin University, China in 1982 and 1985, respectively China, and received 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. He has become IEEE Computer Society Golden Core member since 2010, also, he is a member of Engineering and Physical Sciences Research Council Peer Review College since 2003. He is the Editor-in-Chief of *International Journal of Creative Computing*.

His research interests include software engineering, creative computing, web and distributed computing.

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



**Si-Jing Zhang** received the B. Sc. and M. Sc. degrees, both in computer science, from Jilin University, China in 1982 and 1988, respectively. He received the 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 Communication 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 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, quality of service (QoS) provision, vehicular ad hoc networks, and wireless networks for real-time industrial applications.

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

**Ji-Dong Ge** received the Ph. D. degree in computer science from Institute of Computer Software at Computer Software Department of Nanjing University, China in 2007. Currently, he is an associate professor in Software Institute, Nanjing University, China.

His research interests include software engineering, workflow, process mining, Petri nets, distributed computing, cloud computing, big data, services computing, software architecture, inheritance of behaviour, formal methods, software process, formal verification, model checking, unified modeling language, mobile agents.

E-mail: gjd@nju.edu.cn