A NOVEL PROCESS MINING MODEL FOR TELECLAIM INSURANCE PROCESS

Ganesha K[#1], Gagana J[#2], Namratha A C[#3]
[# 1, 2, 3] Department of Computer Science, Amrita School of Arts and Sciences, Mysuru Campus,
Amrita Vishwa Vidyapeetham, Amrita University, India
[1] asasganesha@gmail.com, [2] gaganajaykumar@gmail.com, [3]namrathaac93@gmail.com

Abstract—In this paper, we propose a process mining model for teleclaim insurance process. The major problem faced by every insurance organization is to manage enormous amount of data, which were generated for every business activities. Managing teleclaim data is a complex task, which requires process model to know the control flow of the event logs. Researchers today use ProM tool as an extensible framework that supports a wide variety of process mining techniques which makes use of α – algorithm. α – algorithm generates a process model. But, the generated process model will not be specific for every cases instead it is the generalized model. So, the logs which will not suit the compliance or the process model will not be considered its neglected which is the main drawback of α-algorithm. To overcome this problem, we propose a Teleclaim Model algorithm. The proposed algorithm generates process models and traces for teleclaim dataset, in which processes flow within their respective models, so that it will not eliminate the logs which do not fit into given compliance. The fitness for each model is obtained by replaying the event logs on the process models to analyze its behavior. The proposed process model is useful for insurance organizations to improve their business process for their clients. Fitness for the proposed models can be used as a base of the insurance company to decide whether the claim is valid or not.

Keywords - Process mining (PM), ProM, α-algorithm, Teleclaim Model (TCM), Petrinets.

I. INTRODUCTION

Process mining is a technique that allows for the analysis of business processes based on event logs. The fundamental idea is to extract process model from event logs recorded by an information system. Process mining aims to improve business processes by bringing in techniques and tools for discovering process, control, data, organizational and social structures from event logs [1]. The main aim of process mining is to extract knowledge about various processes from its process execution logs [2, 3].Process mining bridges the gap between data-centric analysis techniques such as machine learning and traditional model-based process analysis and data mining [6].

A new profession for future is Data science, because the organizations which are unable to utilize data in a smart way cannot survive. It is not sufficient to focus on data storage and data analysis. The actual relationship between data to process analysis is the genuine need of data scientist [23].

Process mining looks at finding the actual confrontation with event data and process models. This technology was very recently made available, but it can be applied to any type of operational processes applications such as: analyzing the process of treatment for patients in hospitals, improving customer service processes in a multinational, being aware of the browsing behavior of customers by means of booking site, analyzing the causes for failures in a luggage handling system, to confine the forged insurance claimant. All of these applications have one thing in common i.e., dynamic behavior which needs to be related to process models. Process mining is the linking chain between both model-based process analysis and data-oriented analysis techniques [7].

One of the major needs for today's business organization is to know the need of process mining. Companies need to learn more about how their processes operate in the real world [8]. Process mining technique attempts to reconstruct a complete process models by extracting the non-trivial and useful real time process information from event logs and exact flow of the processes can be monitored[4, 9].Every insurance company needs to put on a competitive advantage, and to stream line their processes. Since each process contains multiple events, it is also difficult to keep track of events.

Process mining is used broadly in the areas like hospital management, banks, insurance, industrial applications etc, because it helps in process analysis, process design and process enhancement which improves Business process engineering and business process modeling. In this paper we propose a process mining model for teleclaim insurance processes. Teleclaim insurance processes deals with each business process by handling the inbound phone calls, whereby different types of insurance claims are lodged over the phone. The process is supported by two separate call centers operating for two different organizational entities (Brisbane and Sydney). Both centers are similar in terms of incoming call volume and average total call handling time, but both of the organizations differs in timestamp, that they take to complete the process and deploying the agents where there exists time complexity. After the initial steps in the call centre, the remainder of the process is handled by the back-office of the insurance company. Teleclaim is a synthetic event log without noise. Researchers today use ProM tool as an extensible framework that supports a wide variety of process mining techniques which makes use of α - algorithm. To solve the compliance drawback by generating the process model for each case present in the teleclaim insurance dataset, we propose TCM algorithm to overcome this problem.

The rest of the paper is organized as follows: in section II, we give a brief of related work about process mining and its algorithm. Section III, introduces our proposed algorithm and gives the mathematical analysis of the algorithm. Section IV presents the experimental results. Finally, we provide concluding remarks in Section V.

II. BACKGROUND

In this section existing work on process mining is consolidated. Process mining is useful for at least two reasons. First of all, it could be used as a technique to find out how people and procedures really work. In every system the executed process will be logged but does not enforce the specific way of working. In such an environment process mining could be used to gain insight in actual process [10].

Process mining techniques allow for various types of analysis based on so-called event logs. For example, using process mining one can reconstruct a process model from a log generated by some information system. In the last ten years researchers around the world have been working on such techniques [1, 6, 7]. In [8] Cook and Wolf present process discovery as a tool to support the design of software processes, because it is a hard, expensive, and error prone activity, especially for big and complex processes. In [9] Aalst et al, present an algorithm that mine models having three properties in mind: completeness, minimalist and irredundant.

The effectiveness of that α-algorithm was formally proved for a class of process models, the WF-Nets (*Workflow Net*), which are Petri nets that require: (i) a single Start place, (ii) a single End place, and (iii) every node must be on some path from Start to End. However, such an algorithm has severe limitations, for example, the inability to deal with short loops. Noise in the event log is closely related to anomaly detection. Some process mining methods deal with the mining of noisy logs [1, 9, 10, 11, 12], yet their approaches are limited to the frequency evaluation of dependency relation between two activities. For example, infrequent dependency relations between two activities may not be modeled in the resulting process model.

A more sophisticated and promising approach, called genetic mining, was proposed in [14]. This algorithm is based on genetic algorithms, which search for a solution (an individual) that satisfies selection criteria, called fitness function. All previously mentioned process mining methods are mainly concerned with the modeling of normal behavior, yet some of them also deal with noisy logs. Then, in order to fill this gap, recent researches have been addressing the problem of identifying anomalous trace in logs of PAISs [3, 4, 5, 14].

In [5], Aalst and Medeiros present anomaly detection methods, which are supported by α-algorithm. A drawback of this work is that it demands a known 'normal' log, but a known 'normal' log may not be available in applications domains that demand flexible support. In [15] Yang et al, present a framework to detect fraud and abuse in health insurance systems. In this work clinical pathways are used to construct a detection model, whose features are based on frequent control-flow patterns inferred from two datasets, one with fraudulent instances and other with normal instances.

In [3] and [4], Bezerra and Wainer present three different approaches to detect anomalous traces: sampling, threshold, and iterative approaches. Nevertheless, as pointed out by the authors, the methods presented in [3, 4] have serious practical limitations, directly resulting from the adopted process mining algorithm, which cannot deal with larger logs. In [23] many soft computing approaches like (encompassing Evolutionary, Fuzzy and Neural Network techniques) are used in process mining.

Many mining algorithm failed to work on unstructured logs, so to work on spaghetti structure process mining techniques are used. Process Mining is able to fill that gap, providing revolutionary means for the analysis and monitoring of real-life processes. Extensive research in this area allows us to extract information about business process from event logs by using ProM [24], which is a platform independent tool with an extensible framework to supports various process mining techniques.

III. PROPOSED METHOD

Existing work in ProM uses α-algorithm for teleclaim business process event logs, which generates a process model. But the generated process model will not be specific for each cases instead it is the generalized model. So, the logs which will not suit the compliance or the process model will not be considered its neglected which is the main drawback of α-algorithm.

To overcome this drawback in this paper, we propose a TCM (Teleclaim Model) algorithm. TCM generates the process model for each case present in the teleclaim event log. Algorithm 1 presents the steps involved in the

TCM. TCM algorithm, describes the collection of traces 'T', σ represents the sequences of cases in which 'n' is the number of case present in σ. Later checks the length of the selected case meaning the number of traces present in the case. Further checks whether the case starts and end with trace 'a' and 'k'. Detailed description of the whole algorithm is explained in section IV.

Algorithm 1: Teleclaim Model

Step1: $T_l = \{t \in T \mid \exists_{\sigma \in L}\ t \in \sigma\}$ Where $\sigma = \{L_1......L_{n-1}\}$,

Where n = sequence of length

T = set of transitions.(i.e, $t_a, t_b, t_c, t_d, t_e, t_f, t_g, t_h, t_i, t_j, t_k$)

Let us consider $L_1 = \{t_a, t_b, t_d, t_f, t_g, t_k\}$

Step2: Check the length of L_1

Step3: Case should begin with the event t_a, so check whether the event begin with t_a.

$T_{a(start)} = \{t_a \in T \mid \exists_{\sigma \in L}\ t_a = \text{first}(\sigma)\ \wedge\ goto\ T_{k(end)}\ else\ goto\ I\}$

Step4: Everycase should end with the event t_k.

$T_{k(end)} = \{t_k \in T \mid \exists_{\sigma \in L}\ t_k = last(\sigma) \wedge\ goto\ T_p\ else\ goto\ I\}$.

Step5: $T_p = \{t_b, t_c \in T \mid \exists_{\sigma \in L}$ if $(t_a \to t_b)$ $goto\ T_b\ else$ if $(t_a \to t_c)$ $goto\ T_c$ $else\ goto\ I\}$

Step6: $T_b = \{t_b \in T \mid \exists_{\sigma \in L}\ token = t_b \wedge\ if(t_b \to t_d)\ goto\ T_d\ else\ if(t_b \to t_k)\ goto\ T_{k(end)}$ $else\ goto\ I\}$

Step7: $T_c = \{t_c \in T \mid \exists_{\sigma \in L}\ token = t_c \wedge\ if(t_c \to t_e)\ goto\ T_e\ elseif(t_c \to t_k)\ goto\ T_{k(end)}$ $else\ goto\ I\}$

Step8: $T_d = \{t_d \in T \mid \exists_{\sigma \in L}\ token = t_d \wedge\ if(t_d \to t_f)\ goto\ T_f\ else\ goto\ I\}$

Step9: $T_e = \{t_e \in T \mid \exists_{\sigma \in L}\ token = t_e \wedge\ if(t_e \to t_f)\ goto\ T_f\ else\ goto\ I\}$

Step10: $T_f = \{t_f \in T \mid \exists_{\sigma \in L}\ if(T(t_d \# t_e) \wedge (t_d \to t_f))$ $then$ token $= t_f$ $else\ if(T(t_d \# t_e) \wedge (t_e \to t_f))$ $then$ token $= t_f$ $\}$ $\{if(t_f \to t_g)\ goto\ T_g\ else\ goto T_{k(end)}\}$

Step11: $T_g = \{t_g \in T \mid \exists_{\sigma \in L}\ token = t_g,\ if(t_g \to t_k)\ goto\ T_{k(end)}\ else$ $if(t_g \to t_h)\ goto T_h\ else\ goto\ T_i\}$

Step12: $T_h = \{t_h \in T \mid \exists_{\sigma \in L}\ token = t_h,\ if(t_h \to t_i)\ goto T_i\ else\ goto\ T_j\}$

Step13: $T_i = \{t_i \in T \mid \exists_{\sigma \in L}\ token = t_i,\ if(t_i \to t_h)\ goto T_h\ else\ if(t_i \to t_j)\ goto T_j$ $else\ T_{k(end)}\}$

Step14: $T_j = \{t_j \in T \mid \exists_{\sigma \in L}\ token = t_j,\ if(t_j \to t_i)\ goto T_i\ else\ goto T_{k(end)}\}$

Step15: Result *of traces of a given case* R=L_1 *or* $< t_a, t_b, t_d, t_f, t_g, t_k >$

Step16: $I = invalid\ case.$

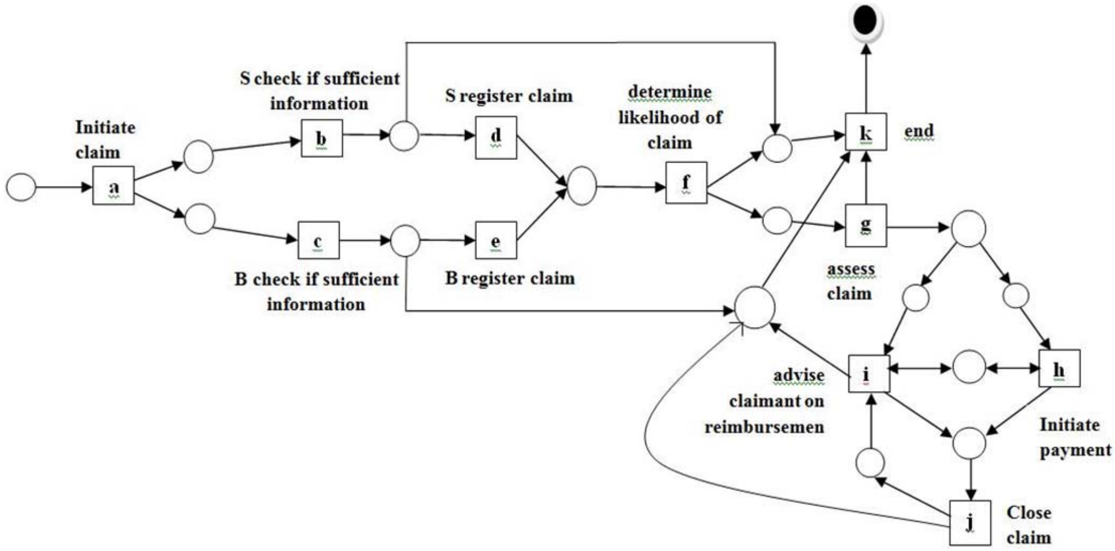

Fig.1. Petri net for the teleclaim event logs

- Initiate claim=a
- S check if sufficient information is available =b
- B check if sufficient information is available=c
- S register claim =d
- B register claim =e
- determine likelihood of claim=f
- assess claim=g
- initiate payment=h
- advise claimant on reimbursement=i
- Close claim=j
- end=k

IV. EXPERIMENTAL RESULTS

For measuring the effectiveness of performance of event logs, each logs fitness is measured by the below Equation 1 [24].

Equation 1:

$$fitness = \frac{1}{2}\left(1-\frac{m}{c}\right)+\frac{1}{2}\left(1-\frac{r}{p}\right)$$

p = produced tokens

c = consumed tokens

m = missing tokens

r = remaining tokens

By replaying the event logs on the generated process model the fitness can be calculated. When all the traces is replayed on the model, and obtain a fitness value is equal to 1 then model is considered to be effective. If the fitness value varies between 0 and 1 then model cannot be effective.

In this paper work implementation is based on Teleclaim Insurance Dataset which contains activities performed by two different organizations (Sydney and Brisbane), the visualiser also represents from which particular organization the activities is performed.Fig.2.represent the teleclaim business workflow in two different organizations (Sydney and Brisbane). The log contains 46138 events related to 3512 case.

case	event	startTime	completeTime	org:resource	location	outcome
0	B check if sufficient information is available	01-01-70 5:30	01-01-70 5:30	Call Centre Agent Brisbane	Brisbane	
0	B register claim	01-01-70 5:30	01-01-70 5:30	Call Centre Agent Brisbane	Brisbane	
0	determine likelihood of claim	01-01-70 6:17	01-01-70 6:17	Claims handler	Brisbane	
0	end	01-01-70 6:17	01-01-70 6:17	Claims handler	Brisbane	not liable
1	S check if sufficient information is available	01-01-70 5:30	01-01-70 5:30	Call Centre Agent Sydney	Sydney	
1	S register claim	01-01-70 5:30	01-01-70 5:30	Call Centre Agent Sydney	Sydney	
1	determine likelihood of claim	01-01-70 5:31	01-01-70 5:31	Claims handler	Sydney	
1	assess claim	01-01-70 5:32	01-01-70 5:32	Claims handler	Sydney	
1	advise claimant on reimbursement	01-01-70 5:34	01-01-70 5:34	Claims handler	Sydney	
1	initiate payment	01-01-70 5:37	01-01-70 5:37	Claims handler	Sydney	
1	close claim	01-01-70 5:38	01-01-70 5:38	Claims handler	Sydney	
1	end	01-01-70 5:38	01-01-70 5:38	Claims handler	Sydney	processed
10	B check if sufficient information is available	01-01-70 5:30	01-01-70 5:30	Call Centre Agent Brisbane	Brisbane	
10	B register claim	01-01-70 5:31	01-01-70 5:31	Call Centre Agent Brisbane	Brisbane	
10	determine likelihood of claim	01-01-70 5:44	01-01-70 5:44	Claims handler	Brisbane	
10	assess claim	01-01-70 5:45	01-01-70 5:45	Claims handler	Brisbane	
10	initiate payment	01-01-70 5:47	01-01-70 5:47	Claims handler	Brisbane	
10	advise claimant on reimbursement	01-01-70 5:48	01-01-70 5:48	Claims handler	Brisbane	
10	close claim	01-01-70 5:49	01-01-70 5:49	Claims handler	Brisbane	
10	end	01-01-70 5:50	01-01-70 5:50	Claims handler	Brisbane	processed

Fig.2.Teleclaim event logs

Consider the workflow log as shown in the Table1. The log contains the information about the 6 valid cases and 3 invalid cases; totally there are 9 cases (i.e., workflow instances) in the log. Each case in the log produces unique results (like not liable, rejected, processed, insufficient information and invalid). We deduce a process model as shown in the Fig. 1. ; The model is represented in terms of a petrinet. The petrinet starts with the execution of trace 'a' and ends with the execution of trace 'k'; traces are represented by transitions. Initially trace 'a' is executed and next either trace 'b' (S check if sufficient information) or trace 'c' is executed; not both at a same time because trace 'b' and trace 'c' are independent of each other (XOR-Split, i.e.,($a \rightarrow b$) and ($a \rightarrow c$) and ($b \# c$)); these traces have been added for routing purposes only and not present in the workflow log.

Case1: a,b,d,f,k–let off.

Case2: a,b,d,f,g,k – Rejected.

Case3: a,b,k - Lack of Information.

Case4: a,b,d,f,g,h,i,j,k – Processed.

Case5: a,b,d,f,g,h,i,j,i,k – Processed.

Case6: a,b,d,f,g,i,h,j,k – Processed.

<u>Invalid Cases:</u>

Case7: a,b,d,f

Case8: b,d,f,k

Case9: b,d,f

A. Experimented Cases

Case1 – Let off Cases

In Case1, let us consider the case L=(a,b,d,f,k) and checks for the length of L and checks whether case starts with trace t_a and ends with trace t_k. if this condition becomes true then token moves to transition 'a' (i.e., $T_{a(start)}$) and then checks whether t_a follows t_b or t_c , since as we mentioned t_b and t_c are independent of each other (i.e., $t_b \# t_c$) t_a should follow either t_b or t_c. According to the case L t_a follows t_b so token moves to transition 'b' (i.e., T_b) and checks whether t_b follows t_d or t_k (i.e.,($t_b \rightarrow t_d$) or ($t_b \rightarrow t_k$) XOR-Split), if there is insufficient information then token moves to transition 'k' (i.e., $T_{k(end)}$) and terminates the process; but according to the case L t_b follows t_d (i.e.,($t_b \rightarrow t_d$)), so token moves to transition 'd'(i.e., T_d). Currently token is at t_d and checks whether t_d follows t_f(i.e.,($t_d \rightarrow t_f$)) and we need to notice is t_d must follow t_f ,because this particular process follows a casualty rule (i,e,. ($t_d \rightarrow t_f$) and not ($t_f \rightarrow t_d$)) and there is no other deviations it can attain; so token moves to transition 'f'(i.e.,T_f). At transition T_f , it checks, whether transitions t_d and t_e is independent of each other (i.e ($t_d \# t_e$) they does not follow each other) and t_f is followed by t_d (i.e$t_f \leftarrow t_d$) else checks t_f is followed by t_e (i.e$t_f \leftarrow t_e$) if any one of the above condition found to be true token stay at t_f .And checks for one more condition whether t_f follows t_g(i.e., ($t_f \rightarrow t_g$)); according to the case L condition ($t_f \rightarrow t_g$) fails so token moves to transition $T_{k(end)}$ and terminate the process. And we obtain the process model as shown in the Fig. 3.

Output:

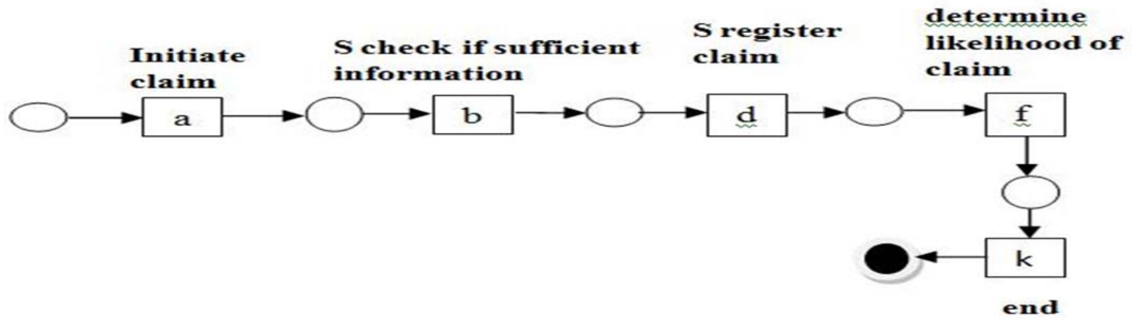

Fig. 3. Not liable process work flow

Case 2 – Rejected Cases

In Case 2, let us consider the case L=(a,b,d,f,g,k) and checks for the length of L and checks whether case starts with trace t_a and ends with trace t_k. if this condition becomes true then token moves to transition 'a' (i.e., $T_{a(start)}$) and then checks whether t_a follows t_b or t_c , since as we mentioned t_b and t_c are independent of each other (i.e., t_b# t_c) t_a should follow either t_b or t_c. According to the case L t_a follows t_b so token moves to transition 'b' (i.e., T_b) and checks whether t_b follows t_d or t_k (i.e.,($t_b \rightarrow t_d$) or ($t_b \rightarrow t_k$) XOR-Split), if there is insufficient information then token moves to transition 'k' (i.e., $T_{k(end)}$) and terminates the process; but according to the case L t_b follows t_d (i.e.,($t_b \rightarrow t_d$)), so token moves to transition 'd'(i.e., T_d). Currently token is at t_d and checks whethert$_d$ follows t_f (i.e.,($t_d \rightarrow t_f$)) and we need to notice is t_d must follow t_f ,because this particular process follows a casualty rule (i.e., ($t_d \rightarrow t_f$) and not ($t_f \rightarrow t_d$)) and there is no other deviations it can attain; so token moves to transition 'f'(i.e., T_f). At transition T_f , it checks, whether transitions t_d and t_e is independent of each other (i.e.,(t_d # t_e) they does not follow each other) and t_f is followed by t_d (i.e$t_f \leftarrow t_d$) else checks t_f is followed by t_e (i.e.,($t_f \leftarrow t_e$) if any one of the above condition found to be true token stay at t_f .And checks for one more condition whether t_f follows t_g(i.e., ($t_f \rightarrow t_g$)); according to the case L condition ($t_f \rightarrow t_g$) becomes valid so token moves to transition T_g. At transition T_g it checks three conditions, whether transition t_g follows t_k (i.e., $t_g \rightarrow t_k$) else checks t_g follows t_h (i.e., $t_g \rightarrow t_h$) else checks t_g is followed by t_i (i.e., $t_g \rightarrow t_i$); according to case L condition($t_g \rightarrow t_k$) becomes valid so token moves to transition $T_{k(end)}$ and terminate the process. And we obtain the process model as shown in the Fig.4.

Output:

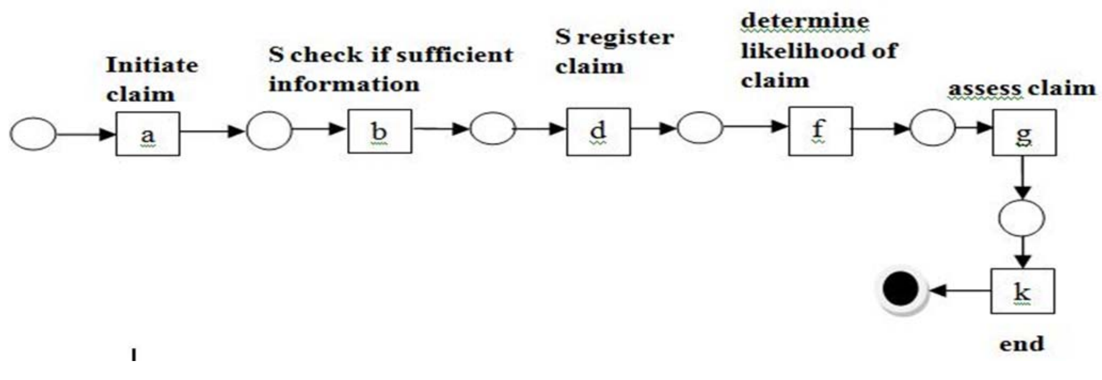

Fig. 4.Rejected process work flow.

Case3 –Lack of Information Cases

In Case 3, let us consider the case L=(a,b,k) and checks for the length of L and checks whether case starts with trace t_a and ends with trace t_k. if this condition becomes true then token moves to transition 'a' (i.e., $T_{a(start)}$) and then checks whether t_a follows t_b or t_c , since as we mentioned t_b and t_c are independent of each other (i.e., t_b # t_c) t_a should follow either t_b or t_c. According to the case L t_a follows t_b so token moves to transition 'b' (i.e., T_b) and checks whether t_b follows t_d or t_k (i.e.,($t_b \rightarrow t_d$) or ($t_b \rightarrow t_k$) XOR-Split), according to case L condition ($t_b \rightarrow t_k$)becomes valid so token moves to transition $T_{k(end)}$ and terminate the process. And we obtain the process model as shown in the Fig.5.

Output:

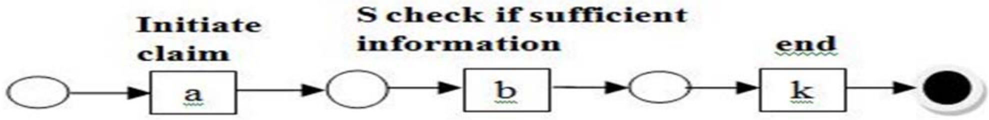

Fig. 5. Insufficient information process work flow.

Case4, Case 5 and Case 6 – Complete Cases

In Case 4, let us consider the case L=(a,b,d,f,g,h,i,j,k) and checks for the length of L and checks whether case starts with trace t_a and ends with trace t_k. if this condition becomes true then token moves to transition ‘a’ (i.e., $T_{a(start)}$) and then checks whether t_a follows t_b or t_c , since as we mentioned t_b and t_c are independent of each other (i.e., t_b# t_c) t_a should follow either t_b or t_c. According to the case L t_a follows t_b, so token moves to transition ‘b’ (i.e., T_b) and checks whether t_b follows t_d or t_k (i.e.,($t_b \rightarrow t_d$) or ($t_b \rightarrow t_k$) XOR-Split), if there is insufficient information then token moves to transition ‘k’ (i.e., $T_{k(end)}$) and terminates the process; but according to the case L t_b follows t_d (i.e.,($t_b \rightarrow t_d$)), so token moves to transition ‘d’(i.e., T_d). Currently token is at t_d and checks whether t_d follows t_f (i.e.,($t_d \rightarrow t_f$)) and we need to notice is t_d must follow t_f ,because this particular process follows a causality rule (i.e., ($t_d \rightarrow t_f$) and not ($t_f \rightarrow t_d$)) and there is no other deviations it can attain; so token moves to transition ‘f’(i.e., T_f). At transition T_f , it checks, whether transitions t_d and t_e is independent of each other (i.e., (t_d # t_e) they does not follow each other) and t_f is followed by t_d (i.e$t_f \leftarrow t_d$) else checks t_f is followed by t_e (i.e$t_f \leftarrow t_e$) if any one of the above condition found to be true token stay at t_f .And checks for one more condition whether t_f follows t_g(i.e., ($t_f \rightarrow t_g$)); according to the case L condition ($t_f \rightarrow t_g$) becomes valid so token moves to transition T_g. At transition T_g it checks three conditions, whether transition t_g follows t_k (i.e., $t_g \rightarrow t_k$) else checks t_g follows t_h (i.e., $t_g \rightarrow t_h$) else checks t_g is followed by t_i (i.e., $t_g \rightarrow t_i$); according to case L condition (i.e., $t_g \rightarrow t_h$) becomes valid so token moves to transition T_h. At transition T_h it checks two conditions, whether transition t_h follows t_i (i.e., $t_h \rightarrow t_i$) else checks t_h follows t_j (i.e., $t_h \rightarrow t_j$), according to case L condition ($t_h \rightarrow t_i$) is true so token moves T_i. At transition T_i it checks three conditions, whether transition t_i follows t_h(i.e., $t_i \rightarrow t_h$) because processes h and i are followed by each other (i.e., (h||i)), else checks t_h follows t_j (i.e., $t_i \rightarrow t_j$) else checks t_i it follows t_k (i.e., $t_i \rightarrow t_k$), according to case L condition ($t_i \rightarrow t_j$) holds good, so token moves to transition T_j. At transition T_j it checks two conditions, whether transition t_j follows t_i (i.e., $t_j \rightarrow t_i$) because processes i and j are followed by each other (j||i),else checks t_j follows t_k (i.e., $t_j \rightarrow t_k$), according to case L condition ($t_j \rightarrow t_k$) holds good, so token moves to transition $T_{k(end)}$ and terminate the process. And we obtain the process model as shown in the Fig. 6.

Output:

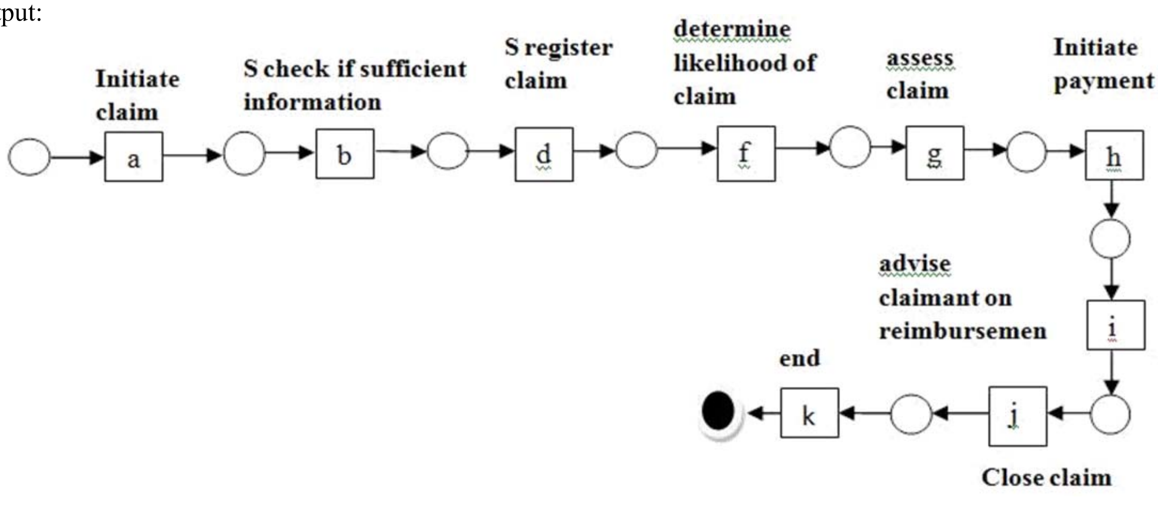

Fig. 6.Processed process work flow.

In Case 5

Let us consider the case L=(a,b,d,f,g,h,i,j,i,k) and checks for the length of L and checks whether case starts with trace t_a and ends with trace t_k. if this condition becomes true then token moves to transition ‘a’ (i.e., $T_{a(start)}$) and then checks whether t_a follows t_b or t_c , since as we mentioned t_b and t_c are independent of each other (i.e., t_b # t_c) t_a should follow either t_b or t_c. According to the case L t_a follows t_b so token moves to transition ‘b’ (i.e., T_b) and checks whether t_b follows t_d or t_k (i.e.,($t_b \rightarrow t_d$) or ($t_b \rightarrow t_k$) XOR-Split), if there is insufficient information then token moves to transition 'k' (i.e., $T_{k(end)}$) and terminates the process; but according to the case L t_b follows t_d(i.e.,($t_b \rightarrow t_d$)), so token moves to transition 'd'(i.e., T_d). Currently token is at t_d and checks whether t_d follows t_f (i.e.($t_d \rightarrow t_f$)) and we need to notice is t_d must follow t_f ,because this particular process follows a casualty rule (i.e., ($t_d \rightarrow t_f$) and not ($t_f \rightarrow t_d$)) and there is no other deviations it can attain; so token moves to transition 'f'(i.e., T_f). At transition T_f , it checks, whether transitions t_d and t_e is independent of each other (i.e., (t_d # t_e) they does not follow each other) and t_f is followed by t_d (i.e$t_f \leftarrow t_d$) else checks t_f is followed by t_e (i.e., $t_f \leftarrow t_e$) if any one of the above condition found to be true token stay at t_f .And checks for one more condition whether t_f follows t_g(i.e., ($t_f \rightarrow t_g$)); according to the case L condition ($t_f \rightarrow t_g$) becomes valid so token moves to transition T_g. At transition T_g it checks three conditions, whether transition t_g follows t_k (i.e., $t_g \rightarrow t_k$) else checks t_g follows t_h (i.e., $t_g \rightarrow t_h$) else checks t_g is followed by t_i (i.e., $t_g \rightarrow t_i$); according to case L condition ($t_g \rightarrow t_h$) becomes valid so token moves to transition T_h. At transition T_h it checks two conditions, whether transition t_h follows t_i (i.e., $t_h \rightarrow t_i$) else checks t_h follows t_j (i.e., $t_h \rightarrow t_j$), according to case L condition ($t_h \rightarrow t_i$) is true so token moves to transition T_i. At transition T_i it checks three conditions, whether transition t_i follows t_h(i.e., $t_i \rightarrow t_h$) because processes h and i are followed by each other (i.e., (h||i)), else checks t_h follows t_j (i.e., $t_i \rightarrow t_j$) else checks t_i follows t_k (i.e., $t_i \rightarrow t_k$), according to case L condition ($t_i \rightarrow t_j$) holds good, so token moves to transition T_j. At transition T_j it checks two conditions, whether transition t_j follows t_i (i.e., $t_j \rightarrow t_i$) because processes i and j are followed by each other (j||i),else checks t_j follows t_k (i.e., $t_j \rightarrow t_k$), according to case L condition ($t_j \rightarrow t_i$) holds good, so token moves T_i and it is the place where looping takes place. Currently token is at T_i, at transition T_i it checks three conditions, whether transition t_i follows t_h(i.e., $t_i \rightarrow t_h$) because processes h and i are followed by each other (i.e., (h||i)), else checks t_h follows t_j (i.e., $t_i \rightarrow t_j$) else checks t_ii follows t_k (i.e$t_i \rightarrow t_k$), according to case L condition ($t_i \rightarrow t_k$) holds good, so token moves to transition $T_{k(end)}$ and terminate the process. And we obtain the process model as shown in the Fig. 7.

Output:

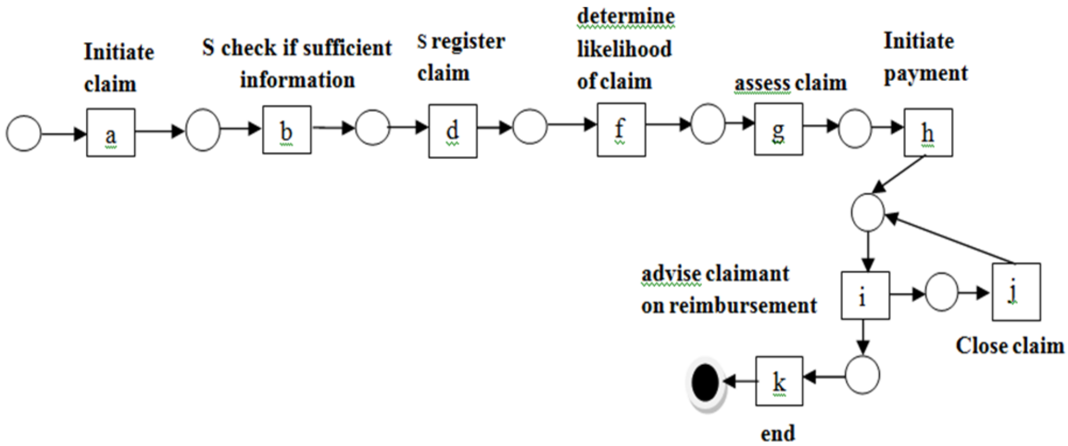

Fig. 7. Processed work flow with loops between two processes.

In Case 6

Let us consider the case L=(a,b,d,f,g,i,h,j,,k) and checks for the length of L and checks whether case starts with trace t_a and ends with trace t_k. if this condition becomes true then token moves to transition 'a' (i.e., $T_{a(start)}$)and then checks whether t_a follows t_b or t_c , since as we mentioned t_b and t_c are independent of each other (i.e., t_b# t_c) t_a should follow either $t_b$or t_c. According to the case L t_a follows t_b, so token moves to transition 'b' (i.e., T_b) and checks whether t_b follows t_d or t_k (i.e.,($t_b \rightarrow t_d$) or ($t_b \rightarrow t_k$) XOR-Split), if there is insufficient information then token moves to transition 'k' (i.e., $T_{k(end)}$) and terminates the process; but according to the case L t_b follows t_d (i.e.,($t_b \rightarrow t_d$)), so token moves to transition 'd'(i.e., T_d). Currently token is at t_d and checks whether t_d follows t_f(i.e.,($t_d \rightarrow t_f$)) and we need to notice is t_d must follow t_f ,because this particular process follows a casualty rule (i.e., ($t_d \rightarrow t_f$) and not ($t_f \rightarrow t_d$)) and there is no other deviations it can attain; so token moves to transition 'f'(i.e.,T_f). At transition T_f , it checks, whether transitions t_d and t_e is independent of each other (i.e., (t_d # t_e) they does not follow each other) and t_f is followed by t_d (i.e., $t_f \leftarrow t_d$) else checks t_f is followed by t_e (i.e., $t_f \leftarrow t_e$) if any one of the above condition found to be true token stay at t_f .And checks for one more condition whether t_f follows t_g(i.e., ($t_f \rightarrow t_g$)); according to the case L condition ($t_f \rightarrow t_g$) becomes valid so token moves to transition T_g. At transition T_g it checks three conditions, whether transition t_g follows t_k (i.e., $t_g \rightarrow t_k$) else checks t_g follows t_h (i.e., $t_g \rightarrow t_h$) else checks t_g is followed by t_i (i.e., $t_g \rightarrow t_i$); according to case L condition ($t_g \rightarrow t_i$) holds good, so token move to transition T_i. At transition T_i it checks whether transition t_i follows t_h (i.e., $t_i \rightarrow t_h$) because (h||i),else checks t_h follows t_j (i.e., $t_i \rightarrow t_j$) else checks t_i follows t_k (i.e., $t_i \rightarrow t_k$), according to case L condition ($t_i \rightarrow t_h$) holds good, so token moves transition T_h. At transition T_h it checks two conditions, whether transition t_h follows t_i (i.e., $t_h \rightarrow t_i$) else checks t_h follows t_j (i.e., $t_h \rightarrow t_j$), according to case L condition ($t_h \rightarrow t_j$) is true so token moves to transition T_j. At transition T_j it checks two conditions, whether transition t_j follows t_i (i.e., $t_j \rightarrow t_i$) because processes i and j are followed by each other (j||i),else checks t_j follows t_k (i.e., $t_j \rightarrow t_k$), according to case L condition ($t_j \rightarrow t_k$) holds good, so token moves to transition $T_{k(end)}$ and terminate the process. And we obtain the process model as shown in the Fig. 8.

Output:

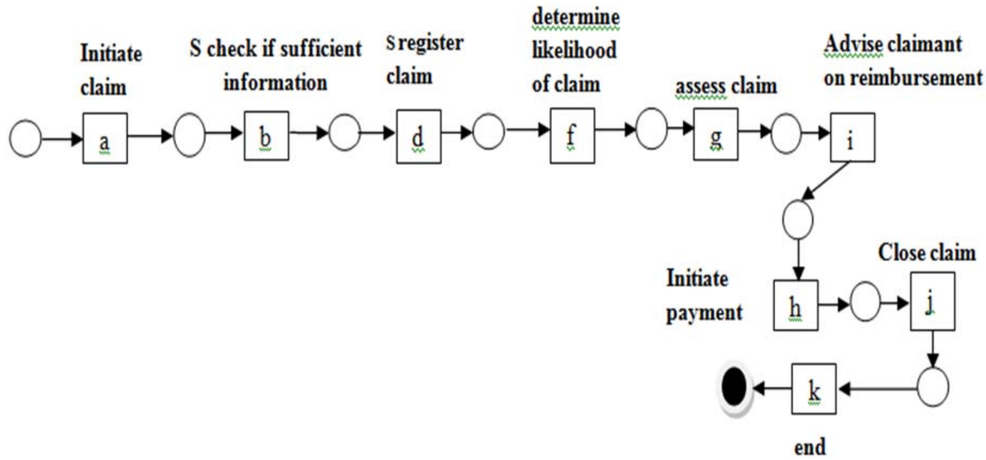

fig.8. Processed work flow using reimbursement step earlier than initiate payment process.

Case7, Case 8 and Case 9 –Invalid Cases

In Case 7

Let us consider the case L=(a,b,d,f) and checks the length of L and checks whether case starts with trace t_a and ends with trace t_k, since case does not end with t_k case will be considered as a invalid process.

In Case 8

Let us consider the case L=(b,d,f,k) and checks the length of L and checks whether case starts with t_a ends with trace t_k, since case does not starts with t_a case will be considered as a invalid process.

In Case9

Let us consider the case L=(b,d,f) and checks the length of L and checks whether case starts with t_a ends with trace t_k, since case does not starts and end with t_a and t_k case will be considered as an invalid process. To evaluate the effectiveness of the generated process model fitness formula is used (Equation 1) [24].

For example: Consider a case (a,b,k) which is replayed on the process model (Fig. 10.) obtained fitness value equal to 1 using the Equation 1, where p=3, c=3, m=0 and r=0.

$$f = \frac{1}{2}\left(1 - \frac{0}{3}\right) + \frac{1}{2}\left(1 - \frac{0}{3}\right)$$

$$\therefore fitness = 1$$

B. Organizational entity 1:

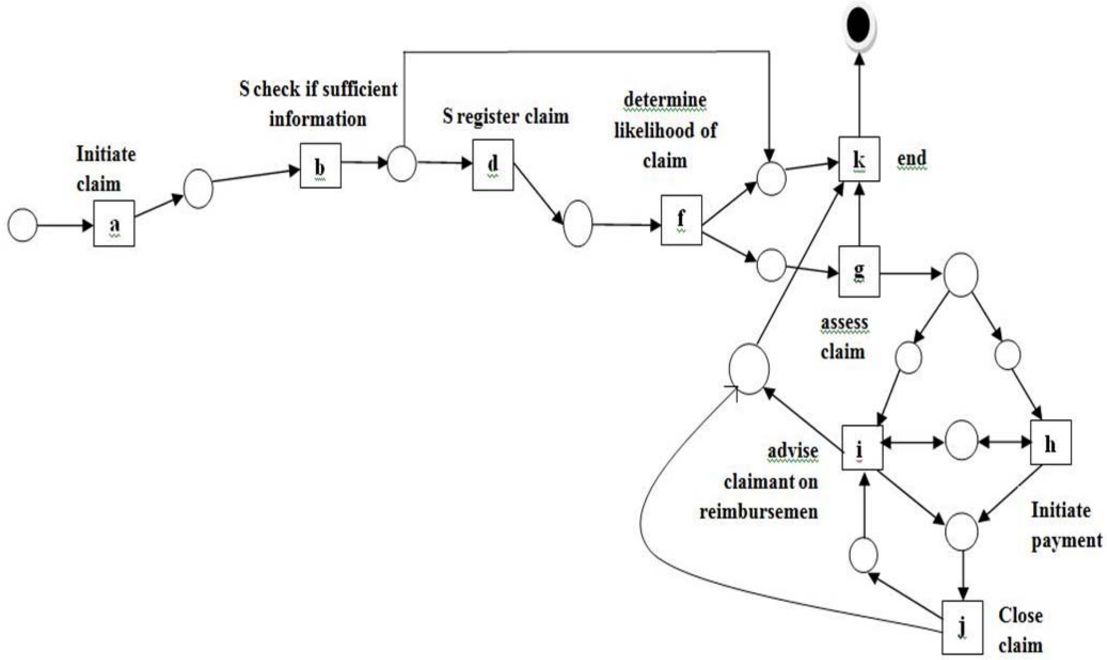

Fig. 9. Organizational entity (Sydney) Model 1

Table I. Fitness value for above traces in model 1

Traces	Fitness
a,b,d,f,g,h,i,j,k	1
a,b,d,f,g,k	1
a,b,d,f,k	1
a,b,k	1
a,b,d,f,g,h,i,j,i,k	1
a,b,d,f,g,i,h,j,k	1

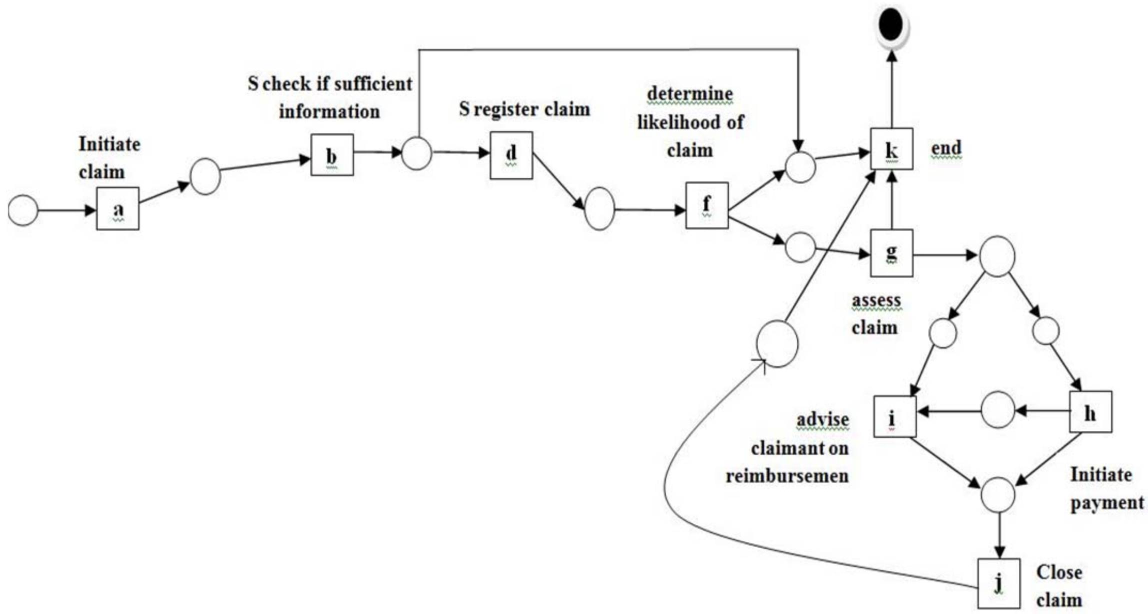

Fig.10. Organizational entity (Sydney) Model 2

Table II. Fitness value for above traces in model 2

Traces	Fitness
a,b,k	1
a,b,d,f ,k	1
a,b,d,f,g,k	1
a,b,d,f,g,h,i,j,k	1
a,b,d,f,g,h,i,j,i,k	0.875
a,b,d,f,g,i,h,j,k	0.875

For example: Consider a case (a,b,d,f,g,h,i,j,i,k) which is replayed on the process model (Fig. 10.) we obtain the fitness value equal to 0.875, where p=8, c=8, m=1 and r=1.

$$f = \frac{1}{2}\left(1-\frac{1}{8}\right)+\frac{1}{2}\left(1-\frac{1}{8}\right)$$
$$\therefore fitness = 0.875$$

C. *Organizational entity 2:*

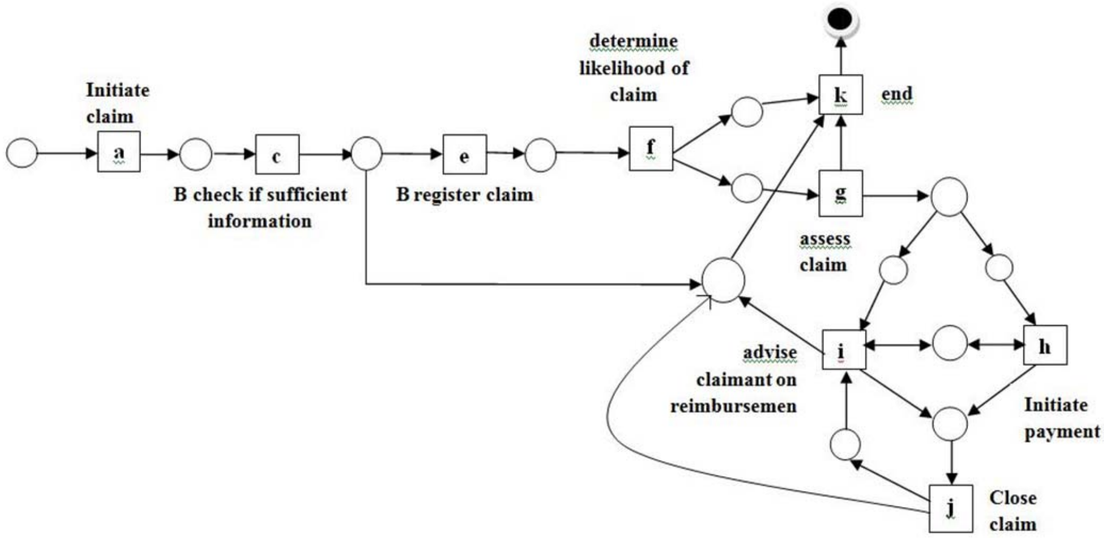

Fig. 11. Organizational entity (Brisbane) Model 3

Table4. Fitness value for above traces in Model 3

Traces	Fitness
a,c,e,f,g,h,i,j,k	1
a,c,e,f,g,k	1
a,c,e,f,k	1
a,c,k	1
a,c,e,f,g,h,i,j,i,k	1
a,c,e,f,g,i,h,j,k	1

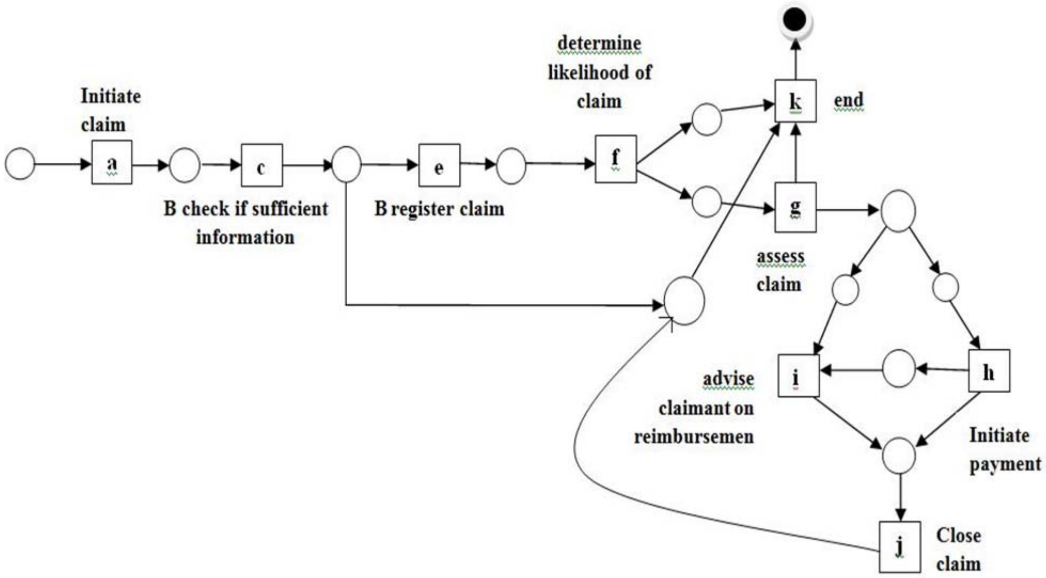

Fig. 12. Organization entity (Brisbane) model4.

Table5. Fitness value of Model 4

Traces	Fitness
a,c,k	1
a,c,e,f ,k	1
a,c,e,f,g,k	1
a,c,e,f,g,h,i,j,k	1
a,c,e,f,g,h,i,j,i,k	0.875
a,c,e,f,g,i,h,j,k	0.875

V. CONCLUSION

This research work deals with event logs of teleclaim, proposing unique models for different valid cases present in teleclaim workflow log. The presented novel approach generates an effective process model by replaying the traces on the process model. The event logs which do not fit into compliance will be considered as invalid cases. The resultant process model is useful for any insurance organization to improve their business process for their clients. Fitness for the proposed models can be used as a base of the insurance company to decide whether the claim is valid or not. The Future work can be extended by constructing an Event Driven Process Chains (EPC's) and also check the conformance using footprints from event logs and generated EPC model, to calculate the fitness, precision and generalization in a single metric.

REFERENCES

[1] van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L.,Schimm, G., Weijters, A.J.M.M, Workflow mining: A survey of issues and approaches. Data & Knowledge Engineering, Volume-47(2), November 2003, pp. 237–267.

[2] van der Aalst, W.M.P., Weijters, A.J.M.M, Process mining: a research agenda. Computers in Industry, Volume-53(3), pp. 231–244, 2004.

[3] Bezerra, F., Wainer, J, Anomaly detection algorithms in logs of process computing awaresystems, In: SAC Proceedings of the 2008 ACM symposium on Applied New York, NY, USA, pp.951952, 2008.

[4] Bezerra, F., Wainer, J, Anomaly detection algorithms in business process logs, In: ICEIS 2008: Proceedings of the Tenth International Conference on Enterprise Information Systems, Volume AIDSS., Barcelona, Spain, pp.11–18, 2008.

[5] van der Aalst, W.M.P., de Medeiros, A.K.A, Process mining and security: Detecting anomalous process executions and checking process conformance, Electronic Notes in Theoretical Computer Science, pp.3–21, 2005.

[6] de Medeiros, A.K.A., van der Aalst, W.M.P.,Weijters, A, Workflow mining:Current status and future directions, In Meersman, R., Tari, Z.,

[7] Schmidt, D., eds.:On The Move to Meaningful Internet Systems,Volume 2888 of LNCS, 2003.

[8] Cook, J.E., Wolf, A.L.: "Discovering models of software processes from event-based data. ACM Trans",Software Engineering Methodology, Volume- 7(3) ,1998, pp. 215–249.

[9] van der Aalst, W.M.P., Weijters, T., Maruster, "Workflow mining:Discovering process models from eventlogs". IEEE Transactions on Knowledge and Data Engineering, pp.1128–1142, 2004.

[10] Agrawal, R., Gunopulos, D., Leymann, F.: "Mining process models from workflow logs". In: EDBT '98:Proceedings of the 6th International Conference on Extending Database Technology, London, UK, Springer-Verlag,1998, pp.469–483.

[11] Cook, J.E., Du, Z., Liu, C., Wolf, A.L.: "Discovering models of behaviour for Concurrent workflows".Computers in Industry, Volume-53(3) ,2004,pp. 297–319.

[12] Pinter, S.S., Golani, M.: "Discovering workflow models from activities' life spans".Computers in Industry, Volume-53(3) ,2004,pp. 283–296.
[13] Herbst, J., Karagiannis, D.: "Workflow mining with in wolves". Computers in Industry, Volume-53(3), 2004, pp. 245–264.
[14] de Medeiros, A.K.A., Weijters, A.J.M.M., van der Aalst, W.M.P.: "Genetic process mining: A basic approach and its challenges". In: Business Process Management Workshops, Volume 3812 of Lecture Notes in Computer Science, Nancy, France, September 2006, pp.203–215, ISBN 978-3-540-32595-6.
[15] Yang, W.S., Hwang, S.Y.: "A process-mining framework for the detection of healthcare fraud and abuse". Expert Systems with Applications, Volume 31(1), pp. 56–68, July 2006.
[16] Van Dongen B., de Medeiros, A., Verbeek, H., Weijters, A., van der Aalst, W.: "The prom framework: A new era in process mining tool support". In: Applications and Theory of Petri Nets. Volume 3536 of Lecture Notes in Computer Science, 2005.
[17] MadderiSivalingamSaravanan, Rama S ree .R.J, "Application of process mining in insurance: A Case study for UTI ", International Journal of Advanced Computer and Mathematical Sciences ISSN 2230-9624, Vol 2, Issue 3, 2011, pp.141-150
[18] P. V. Kumaraguru and Dr. S. P. Rajagopalan, "Model Discovery from Motor Claim Process Using Process Mining Technique", International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013.
[19] H.M.W. Verbeek, W.M.P. van der Aalst. "Decomposed Process Mining: The ILP Case." Business Process Management Workshops, Volume 202, pp.264–76,(April 12, 2015).
[20] Leemans, M., & van der Aalst, W.M. (2015, September). Process mining in software systems: Discovering real-life business transactions and process models from distributed systems. In Model Driven Engineering languages and Systems (MODELS), 2015 ACM/IEEE 18thInternational Conference on (pp. 44-53). IEEE
[21] Dixit, P. M., Buijs, J. C. A. M., van der Aalst, W. M. P., Hompes, B. F. A., &Buurman, J. Enhancing Process Mining Results using Domain Knowledge.
[22] Calvanese, D., Montali, M., Syamsiyah, A., & van der Aalst, W. M. Ontology-Driven Extraction of Event Logs from Relational Databases.
[23] Turner C J, Tiwari A, Olaiya R, XuY: "Business Process Mining: From Theory to Practice". Business Process Management Journal, Volume 18, Issue 3, pp.493-512, 2012.
[24] http://www.processmining.org

AUTHORS

Ganesha K: Received his MCA degree from ICFAI University, Tripura. He is currently serving as Lecturer in Computer Science Department, Amrita Vishwa Vidyapeetham University, Mysuru Campus, Mysuru. His area of research includes Process Mining, Database Systems and Data Mining.

Gagana J: She is currently pursuing her MCA degree in Amrita Vishwa Vidyapeetham University, Mysuru Campus, Mysuru. Her area of research includes Process Mining and Data Mining.

Namratha A C: She is currently pursuing her MCA degree in Amrita Vishwa Vidyapeetham University, Mysuru Campus, Mysuru. Her area of research includes Process Mining and Data Mining.