https://doi.org/10.17576/apjitm-2017-0602-08

http://www.ftsm.ukm.my/apjitm
Asia-Pacific Journal of Information Technology and Multimedia
Jurnal Teknologi Maklumat dan Multimedia Asia-Pasifik
Vol. 6 No. 2, December 2017: 77 - 90
e-ISSN: 2289-2192

EVALUATION AND EXTRACTING FACTUAL SOFTWARE ARCHITECTURE OF DISTRIBUTED SYSTEM BY PROCESS MINING TECHNIQUES

MAHDI SAHLABADI
AMIRHOSSEIN SAHLABADI
RAVIE CHANDREN MUNIYANDI
ZARINA SHUKUR

ABSTRACT

The factual software architectures that are actually implemented of distributed systems do not conform the planned software architectures (Beck 2010). It happens due to the complexity of distributed systems. This problem begets two main challenges; First, how to extract the factual software architectures with the proper techniques and second, how to compare the planned software architecture with the extracted factual architecture. This study aims to use process mining to discover factual software architecture from codes and represents software architecture model in Petri Net to evaluate model by the linear temporal logic and process mining. In this paper, the applicability of process mining techniques, implemented in the ProM6.7 framework is shown to extract and evaluate factual software architectures. Furthermore, capabilities of Hierarchical Colored Petri Net implemented in CPN4.0 are exploited to model and simulate software architectures. The proposed approach has been conducted on a case study to indicate applicability of the approach in the distributed data base system. The final result of the case study indicates process mining is able to extract factual software architectures and also to check its conformance.
Keyword: distributed software architecture, process mining, color petri net

INTRODUCTION

BACKGROUND AND MOTIVATION

A distributed system normally has various components that are communicating through the network. Perhaps one of the significance of distributed system is integration of a large number of legacy applications distributed over the network. A distributed system can be developed by various programming languages and established on various distinctive operating systems. Sometimes, using a distributed system is more economical than a centralized system. Utilizing of existing low power systems is financially justifiable rather than purchasing either supercomputers or mainframes (Qureshi 2005). Naturally, a distributed software design is an untamable problem (Feldgen, M. , Clua, O 2012). Understanding the essence of pervasive software based on system requirement is a great concern in the distributed system in which software components are spread over several nodes. This will lead to increase the complexity of the system. As a result, it is essential to organize these systems properly (Feldgen, M. , Clua, O 2012).

Software architectures for distributed systems are bulky and complex because of the transparency (ability of presenting a distributed system to the users as if it is a single machine) of the distributed system. Transparency conflicts with the other requirements. So the middleware is used as a highly flexible solution as it can facilitate integration of various components over the distributed system (Qureshi 2005). Integration of components is an error prone process for designing. Executable models of software designs offer this opportunity to simulate and test the design logic prior to implementation of the software (Gomaa 2008). Software designers frequently sketch different designs, in order to explore the design's problems, particularly during the early stages of software design. This overdesigning imposes prematurely the conformity and precision in case of the formal design tool usage (Mangano, N., Dempsey, M. and Lopez, N., Van der Hoek, Andre 2011). Additonaly, in last two decades, the distributed software system has been shifted from "data-aware" information systems to "process- aware" information system (PAIS) (Dumas, et al. 2005). In addition, Service Oriented Architecture (SOA) is viewed as a collection of services to support the business process (van der Aalst 2004), (Georgakopoulos 1995), (Muehlen 2004). Being aware of all processes and services is mandatory to have an effective SOA design, while there are enormous processes tangled in a distributed system.

Process mining makes a great improvement in application domains' processes. Information systems store huge number of events. However, there is still a problem to gain a useful value from these event data. Process mining facilitates the system to extract the process, which is related information from the event data. It automatically discovers a process model from the recorded event (W. M. Aalst 2011).

Colored Petri Nets (CP-nets or CPNs) is a graphical language for concurrent system modeling and evaluation of their features. It is a modeling language based on the discrete event and capability of Petri net. This language has been enriched by the high level programming language. Petri nets present a model in graphical notation that is emphasizing concurrency and synchronization of the model (Kurt 2009).

Process mining discovers processes from the event log. Processes can be presented in CPNs as it describes processes accurately. CPNs also can be used for simulation in order to synthesize data for process mining. Complex designs can be sketched by CPNs tools in order to simulate the event log for process mining. Then, process mining tools discover and evaluate processes to test CPNs model. CPNs and process mining combination eases the designing of software architecture in early stages of the software development. Moreover, implemented software can be logged according to process mining criteria during the run time. So, the factual software architecture can be extracted and evaluated against the designed software architecture which has been introduced in early stages of the software development. This approach is applicable particularly in contexts of distributed system.

PROBLEM STATEMENT

Seamless interconnection and organization of multiple software and components are emerging a challenge, which is taking over coding of individual modules (Dumas, et al. 2005). Consequently, designing of service-oriented, adaptive software, and evolutionary software architectures are also challenging (Gomaa 2008). Software systems have not any tangible representation, which allows us to perceive directly the realization of the large-scale abstractions (software architecture). It is difficult to identify components and their interactions (factual software architecture) in distributed software. In order to reduce this challenge, Software Architecture is treated independently to have an accurate understanding of the requirements (functional and non-function) (Gardazi, S.U. and

Shahid, A. 2009). Using the new paradigms of "from programming to assembling", and "from data orientation to process orientation", software architecture becomes more independent from the platform (Dumas, et al. 2005).

There are many approaches to assess extracted software architectures free from its platform. Unified Modeling Language (UML) 2.0 is being used mostly in software architecture modeling and assessment area (Jensen and Aalst 2009) (Rodriguez-Priego 2010), but it suffers from lack of semantic and presenting complex interaction. So it uses formal aids like Petri net as a complementary method (Kirsten Berkenkötter 2008), (ShangGuan We 2009) (Virbitskaite 2007) (Baek Jorgensen 2004). Petri Net has the proper visualization beside its semantic (W. M. Aalst 2011) (Kurt Jensen 2009), which helps stockholders to understand software architecture more conveniently.

Process mining in practice indicates that the ideal models applied to configure system are not really reflected in the real life model (Kurt Jensen 2009). Although many tools and techniques have been used in the real-life processes in process mining, still there is a problem to extract the proper models from the log files (van Dongen 2003) (Reijers 2007) (ShangGuan We 2009).

The challenge is how to collect the appropriate event data in order to apply for process mining. Firstly, data may be spread over various resources. Secondly, event data is normally uncompleted. Thirdly it may be contaminated by outliers. Lastly, event data might not be stored in the same level of granularity (v. d. Aalst 2012). In practice, process mining relies on event logs of highly complex processes to discover the models or check the conformance of the existing models (Kurt 2009) (v. d. Aalst 2012).

This research aims to extract software architecture models from the log files of distributed systems by process mining techniques and presents software architecture in Petri net. This happens in order to extract factual software architectures from the system. Process mining deals with the log files of system and overcomes to the challenge of large-scale abstraction. The model which is presented in Petri net is verifiable and executable. There are some trivial jobs in between which connect the steps of methodology.

PETRI NET AND PROCESS MINING IN SOFTWARE DESIGN

The executable model of software architecture with mathematical proof techniques can validate software architecture design prior to implementation. The executable model may synthesize textual or semi-formal specification to simulate software architecture model. Also, a visualized model of software architecture indicates that the system has been explored deeply (Heft 2004).There is a need of static models (object frames) beside dynamic models (activity frames) to simulate non-deterministic behaviors of a software system (Jeffrey J.P. Tsai 2009). Complexity and interleaving of software system may lead to state explosion problem (Sharafi 2007). Recently, Petri nets are used to formalize UML interactions because they describe the true concurrency and interleaving behaviors within software interactions. The UML notation meets the user's needs flexible enough to heed their expectations, but this flexibility rooted from semantic informality that can be interpreted differently (Gomaa 2008). So it is difficult to analyze and verify automatically. UML 2.0 is more precise than UML 1.x., but UML 2.0 still encounters with problems in terms of informality and lack of tools to analyze and validate models automatically (Thouraya Bouabana 2013). Normally, executable models in the area of software architecture can be presented by Petri net, queuing network and stochastic process algebra (S. a. Emadi 2008). Rodriguez introduces references concept map that is insightfull for software modeling. it suggests Petri net as the powerful modeling technique (Rodriguez-Priego 2010). Medvidovic compares

between 10 Architectural Description Languages (ADLs) includes; UniCon ACME, Wright Aesop, MetaH, C2, Darwin, Rapide, SADL, Weavs .The study states that the Hierarchal Colored Petri Net possesses all the capabilities in order to model the software architecture (Medvidovic 2000). On the other hand, Use Case is often described in an informal language. Although it is readable, still there are needs to be precise in specification. There are several researches to formalize Use Cases (v. d. Aalst 2012), such as BNF-like grammar which is formally described them. (Gervasi 2009) Due to this end, there are several types of UML(Fuzzy UML) F-UML transforms F-UML to formal model (Fuzzy Petri Net), and then presents feedbacks to software model elements. It shows the ability of handling uncertainty in information systems modeling by using of the standard language (Haroonabadi 2008). CPN tool offers some techniques for probability in transition of states. Generally, Petri nets are being applied in software requirement elicitation, software modeling and upholding UML in order to execute and verify diagrams. Typically, in recent distributed systems, software and hardware are intertwined tightly. Traditional development is exposed to new challenges for the system component integration (Staines 2010). Formal process of algebraic language like (CCS, π – calculus, CSP, ACP) is frequently used in academic researches like Petri nets. The trend follows clarity of specification of the process regardless of any special attention to the particular analysis technique (Kurt Jensen 2009).

There are some researches (Michel dos Santod Soares 2008), (Simonetta Balsamo 2012), (Zee 2011), (Wensong Hu 2011), (Staines 2010), (Khadka 2008), (S. a. Emadi 2008), (Emadi and Shams 2008), (Heft 2004), (Gervasi 2009), (Thouraya Bouabana 2013), (Pettit 2004), (ShangGuan We 2009), (Meier 2011) that implicitly or explicitly state there is a need of formal approach for current UML modeling. In addition, they suggest Petri nets in order to make UML more formalized.

PROCESS MINING ADOPTION

Complexity and dynamism of reality is higher than what models represent. Sometimes, models are abstracted from details and some of their aspects are not relevant to the purpose of the model (Kurt Jensen 2009). What process mining does is to connect between the real process model and their data. Basically, information systems record the event data in an unstructured way. The data should be collected from various tables or sub-systems' exchanging messages. Accordingly, it is necessary to extract and filter the event data from the existing data. This is mandatory part of process mining (W. M. Aalst 2011).

Some researches (Hua Duan 2009), (W.M.P. van der Aalst a 2011), (Kaymak 2012), (Lemos 2011) apply process mining in the field of software management. These studies show that process mining is applicable in various area of software engineering. Besides, there is not any research has been done in the current proposed area so far.

PROPOSED SOLUTION

According to reviews of the last research in software architecture evaluation (Gorton 2009), categorization of the software architecture discovery method is a very difficult job. Consequently, the proposed approach has adopted from activities of the late software architecture evaluation method (Lindvall 2003), which is a standard method.

ATAM, ALMA, PASA are well-known software architecture review techniques (Gorton 2009). However, they suffer from lack of agility (Kijas 2013). They are not Ad-hoc (Babar 2014)

and cannot be applied on the late stage of software architecture implementation easily (Lindvall 2003).

IDENTIFYING ARCHITECTURE ELEMENTS FROM ACTUAL SYSTEM

Software architecture perception is very crucial for maintenance and evolves of system. Typically, software architecture is depicted and shared in documents and diagrams in which they are created manually. Sometimes, software architecture models do not comply the actual implemented architecture. The factual architecture is an expected architecture which is attempted to be illustrated in the software architecture model. However, the implemented software contains the architecture implicitly via code structures and its dependencies (Beck 2010). The comparison of early software architecture, which is planned, with the implemented architecture in the latter point can indicate architectural drifts. Besides, evaluating the implemented software architecture versus the documented architecture reveals architectural violations (Beck 2010) (Dumas, et al. 2005). In this step, software components and modules are identified by the system document review.

IDENTIFY EVENTS

Process mining produces business process models from the log file, which is the list of the sequence of activities. These activities associate with an event. Therefore, we should identify the events prior to start with process mining in a system.

Assume E is the event universe, i.e., the set of all possible event identifiers. Events is featured by different attributes, e.g., an event has a timestamp, correspond to an activity, is executed by special doer. Let AN be a set of attribute names. For any event $e \in E$ and name $n \in AN$:$\#_n(e)$ is the value of attribute n for event e . If event e does not have an attribute named n , then $\#_n(e) = \perp$ (null value).

For a better understanding, the standard attributes are presumed below:

- #activity(e) is the activity associated to event e .
- #time(e) is the timestamp of event e .
- #resource(e) is the resource associated to event e .
- #trans(e) is the transaction type associated to event e , examples are schedule, start, complete, and suspend.

(Classifier) For any event $e \in E$, $\underline{e}$ is the name of the event.

If events are identified by their activity name, then $\underline{e}$ = #activity (e). Also it is possible that

$\underline{e}$ = #resource(e).

Assume A is a set of activity names. A basic trace σ is a sequence of activities, i.e., $\sigma \in A^*$.A simple event log L is a multi-set of traces over A.

All cases in L are converted into sequences of (activity) names using the classifier. A case $c \in L$ is an identifier from the case universe C. $(c^{\wedge}) = \#trace(c) = \langle e_1,e_2,...,e_n \rangle \in E^*$ is the sequence of events executed for c (#trace(c) can be any possible subset of all events' set). $(\underline{c}^{\wedge}) = \langle \underline{e}_1,\underline{e}_2,...,\underline{e}_n \rangle$maps these events onto (activity) names using the classifier.

The α-algorithm gets workflow log as an input and returns Petri net as an output. Bear in mind, that the time record confirms that the tasks are totally logged in order. According to such a logging process, definition introduces four ordering sub-relations among tasks:

$$> w, \rightarrow w, \#w, ||w \tag{1}$$

Definition 1 (Log-based ordering relations) Let W is a workflow log over T (W is a scoped part of log T. It means W can be any subset of T). W∈ (T*), Let a, b∈T (a and b are any different events which belong to T) (W. M. Aalst 2011):

(1) $a >_W b$ if and only if there is a trace $s = t_1 t_2 \cdots t_{n-1} t_n$ and i ∈ {1,…,i-1} such that s ∈ W and $a = t_i, b = t_{i+1}$,

(2) $a \rightarrow_W b$ if and only if $a >_W b$, and $b \not>_W a$

(3) $a \#_W b$ if and only if $a \not>_W b$, and $b \not>_W a$

(4) $a \|_W b$ if and only if $a >_W b$ and $b >_W a$.

FIGURE 1. Relationship between events

The algorithm indicates the relationship between two events can be stated in four different types: 1. Ordered, 2. Succession 3. Not ordered and 4. Parallel.

MAKING LOG FILES

The data need to be transformed to a log format. A database D, any tuple d ∈ D has the form of d =<c,t,s>, whereas d.c identifies an unique component, and d.s indicates the location of the component at time d.t. The time is assumed as a discrete variable, which is expressed by the discrete value with defined granularity. Moreover, the event data unit may happen anytime, apart from any regular intervals found in time series. Additionally, space is modeled as a semantic locations' collection, either a physical spaces such as special node or station, or cyber locations such as web addresses and domains. Spatiotemporal database D as indicated in graph G (GV, GE), where: GV is the set of nodes/components in G, GE is the set of edges/links in G according to spatiotemporal co-occurrences between components (Lauw 2005) (Sahlabadi 2014).

In this study, the new tuple is proposed with two additional fields of m and r. Moreover, d.r represents the component role in an interaction. Also, r ∈R*, indicates that the component can have the multiple role. The d.m is an event type which depends on the system and m ∈M*,
R ∈ {$sender, receiver, initiator$}
m is the event type, and m ∈M, (Sahlabadi 2014)

MERGING LOG FILES

As the log file would be disterbuted over the differnet nodes, there is a need of merging alogrithm to collect all logs to a single log file.

Input: A #Thread(e)

Output: Merged traces

1. e = #trace(#Thread(e)))→ t_n
2. foreach loge file contain (#Thread(e)) do
3. extract t in which ((#Thread(e) ∈ t) Λ (tail(t_n) #time < head(t)#time))
4. $t \rightarrow t_{n+1}$
5. append(t_n, t_{n+1})
6. end

The above algorithm should be done on logs and data in order to have a final model. Dealing with various data stored over the distributed system and merging them together, is crucial along with scoping related data in terms of time and space. Append function adds the subsequent event which has been stored in different log files to the event happened prior to it. The case study would vary in such areas and there is not any standard way to support the unique method.

PLAY-IN, PLAY-OUT, AND REPLAY

It is worthy to mention that process mining is ambitious to provide reasonable ties between a process model and the "reality" obtained from an event data. The terms of Play-in, Play-out, and Replay are coined for sake of above-mentioned fact. Play-out is one of the basic usages of process models. Petri nets generate many possible behaviors. The technique of "playing the token game" executes repeatedly and produces traces by means of Petri nets. Play-out either analyzes models or enacts business processes. Normally, Play-out engines are simulation tools, which run the experiments. Actually, the simulation runs a model repeatedly in order to effectively collect confidence intervals and statistics. Moreover, the exhaustive state-space's analysis is a rampant method named model checking that can be executed as Play-out methods. In opposite side of the Play-out method, Play-in method stands. Sample of behaviors are collected to produce the proper model that is generalizing the specific behavior. Play-in method is also known as an inference. Play-in techniques discover event logs by various discovery approaches. The α-algorithm is one of the basic Play-in approaches. Event log and its relevant process models are both inputted to Replay. The event log is replayed based on the process model (W. M. Aalst 2011). Simulation also could prove the model validity. Figure 2 has shown the simulation originated from the reality and compared between them. Using this model, it is possible to identify the model's correctness.

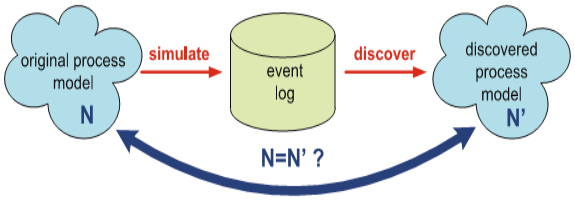

FIGURE 2. Simulation for validation (W. M. Aalst 2011)

EXAMPLE OF ANALYSIS

Considering system policy ,security strategy and rules , there is an example in a system that data should be stored in various remote nodes. Besides,the petri net model with the performance

considertion (Time out palce) is evaluted. The main requierments of software architecture are normally stated in natural languags or UML diagrams, they both can be converted in to Petri nes (Emadi and Shams 2008) (S. a. Emadi 2008). Figure 3 has shown the Petri net model of software architecture based on main requiermens of system. In this figure, the policy is as follow; Data should be stored three times in different nodes.

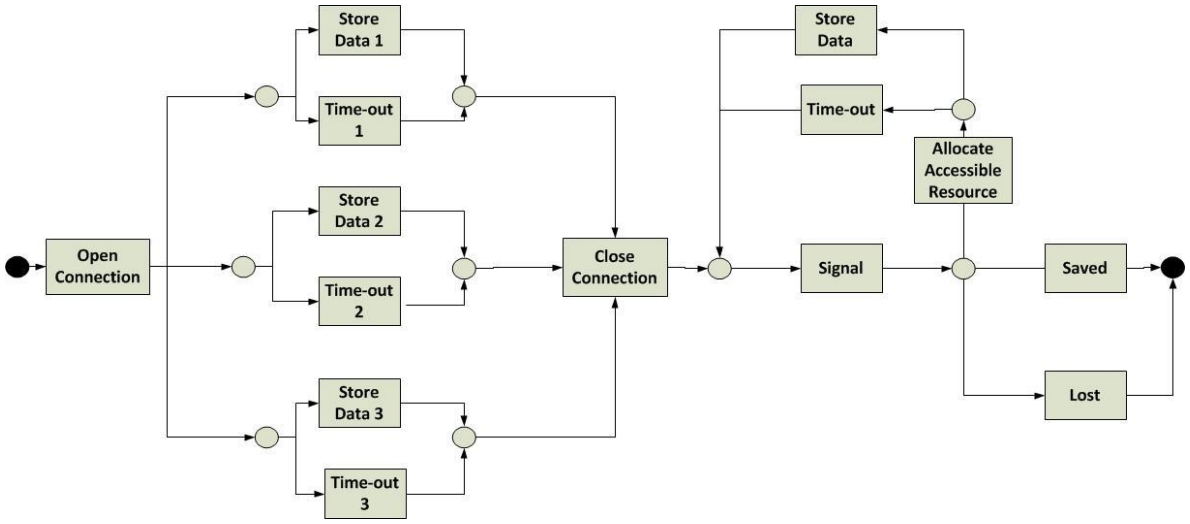

FIGURE 3. Shows model of secure transaction

In this research, log files have been synthetized by Color Petri Nets (CPN) tool (K. a. Jensen 2009). CPN Tool 4.0 simulates timed Hierarchal Colored Petri nets. Figure 4 shows the Petri net model synthetizes raw data. The start node produces ten tokens to run the model ten times.

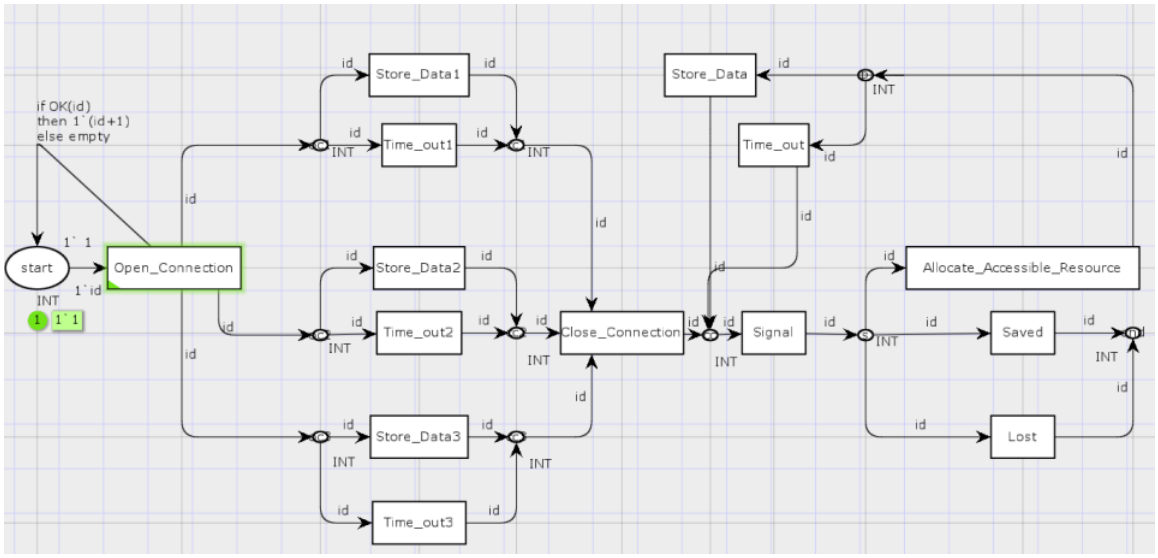

FIGURE 4. Shows HCPN model of simulation

The data which has been produced by CPN should be converted to Extensible Event Stream (XES)/ Mining Extensible Markup Language (MXML) format which is understandable for Prom. Figure 5 indicates the raw data has been provided by simulation of CPN tool. The raw data has been modified regarding to Making / Merging Log Files steps. The final log file consists of 10 process instances and 82 events.

Step	ProcessId	Event	CaseId	Timestamp
1	0	Open_Connection	1	##########
2	0	Time_out1	1	##########
3	0	Open_Connection	2	##########
4	0	Store_Data3	1	##########
5	0	Store_Data3	2	##########
6	0	Open_Connection	3	##########
7	0	Time_out3	3	##########
8	0	Store_Data1	2	##########
9	0	Store_Data2	3	##########
10	0	Time_out1	3	##########
11	0	Close_Connection	3	##########
12	0	Open_Connection	4	##########
13	0	Store_Data3	4	##########
14	0	Open_Connection	5	##########
15	0	Time_out2	5	##########
16	0	Store_Data3	5	##########
17	0	Store_Data1	4	##########
18	0	Open_Connection	6	##########
19	0	Store_Data1	6	##########
20	0	Time_out1	5	##########
21	0	Store_Data3	6	##########
22	0	Close_Connection	5	##########
23	0	Signal	5	##########

FIGURE 5. Initilal log file

Then, as data is clean, Alpha algorithm is a good choice to extract the model. Prom 6.7 has been used in order to convert data to the log file. Alpha algorithm has been applied by means of alpha-plugin of Prom. Synthesized data is used in order to ensure about efficiency of the approach. As it is shown in the figure 6, the resulted model presented in Petri Net model is similar to figure 3. It indicates that the extracted factual software architecture conforms the planned architecture.

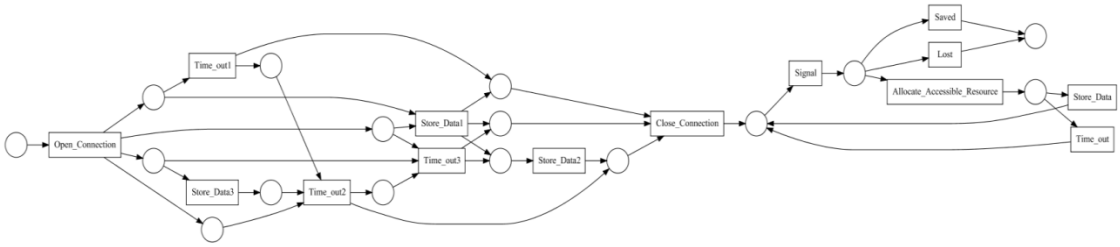

FIGURE 6. Sofwtare architcture in Petri net model exctracted by Alpha algorithm

In this study, the model is checked by Linear Temporal Logic (LTL),which is kind of a temporal logic. LTL has additional operators. The primitive logical operators which are temporal operators are; always ([]), eventually (<>), until (⊔), wait until (W), and next time (O).

LTL checker plugin of Prom 6.7 loads the LTL procedures. As can be seen below in the snippet of LTL code, there are four main procedures: 1. Saved_Xor_Lost() which ensures the traces should be ended either with Saved or Lost node, 2. Data_Should_Store_three_times() which checks the data should be stored three times to assume it as a saved case, 3. No_same_resource() that ensures the data has been stored in three different resources, 4. Following_action_Signal() ensures that if the data is not saved three times, it should be saved again.

```
1. procedure Saved_Xor_Lost ( ) :=
2. (<>(activity == "saved") <-> !(<>(activity == "lost")));

3. procedure  following_action_Signal() :=
4. []( (activity == " Signal" -> _O( ((activity == "saved" \/
5. activity == "Lost") \/ activity == "AllocateAccessibleResource") )));

6. execute( r : resource, a : activity ) :=
7. ( (activity == a /\ resource == r ) ) ;

8. procedure no_same_resource() :=
9. forall[r:resoure |
10. (((!(execute(r,"StoreData 1")) \/ !(execute(r," StoreData 2"))) /\
11. (!(execute(r," StoreData 1")) \/ !(execute(r," StoreData 3")))) /\
12. (!(execute(r," StoreData 2")) \/ !(execute(r," StoreData 3")))) ];

13. saved(a : activity ) :=
14. ( (activity == a /\ return == "saved" ) ) ;

15. procedure Data_Should_Store_three_times( ) :=
16. (((accept("StoreData 1") /\ accept("StoreData 2")) /\
17. accept("StoreData 3"))
18. ( activity == "saved" ) );
19. formula started_before_finished_after( start _tim e:tim esta mp,
20. end_time:timestamp) :=
21. (<>( timestamp < start_time ) /\ <>(    timestamp > end_time ));
```

Result of LTL after executions indicates that the log file does not have any non-compliant trace for Saved_Xor_Lost() and No_same_resource(), while non-compliant trace is 8 and 9 for Data_Should_Store_three_times(), and 1,3,7,8 and 10 for following_action_Signal().Figure 7 shows the non-compliant traces for following_action_Signal in Prom6.7

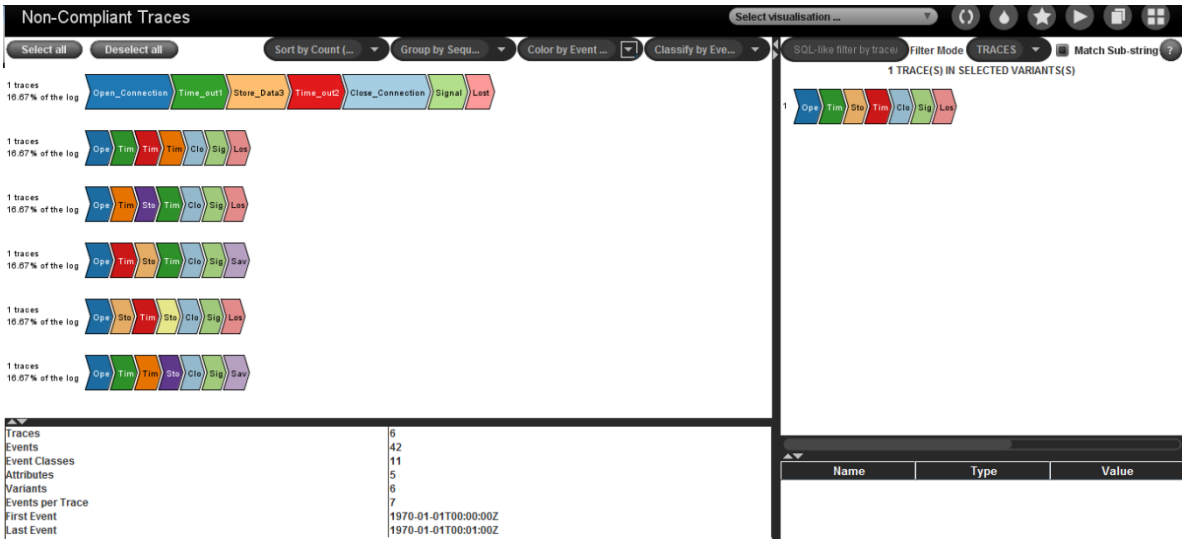

FIGURE 7. shows non-compliant traces of following_action_Signal().

CONCLUSION

Process mining techniques help designers to collect the transform log files in order to process a model using Petri Net presentation. Process mining reduces the documentation overwork. It relies on the factual and detailed system process rather than the abstract or briefed model. It also makes the software architecture maintenance easier. Petri Net has the great capability to represents a model with functional and non-functional details.

The approach extracts the real software architecture and evaluates the software architecture with its drift from the planned software architectures which has been designed in early stage of software development. It is also simple, graphical to understand by other stockholders and executable to simulate abstract indicating model behaviors in details. This approach saves time and budget as it can be applied during the project, unlike other approaches that only be applied once before starting of the project. The approach can be considered as an agile software architecture method.

The idea of using process mining in software architecture design is introduced for the first time in this research. It can also be extended to the software requirement engineering. In addition, using this approach has the revolutionary effects. The researchers planned to drag further research on the software architecture maintenance of enterprises in which more methods and facts will be explored.

REFERENCE

Aalst, van der. "Service Mining: Using Process Mining to Discover, Check, and Improve Service Behavior." Service Computing, IEEE Transactions on, 2012: 1-11.

Aalst, Wil M.P. van der. Process Mining Discovery, Conformance and Enhancement of Business Processes. Springer-Verlag Berlin Heidelberg , 2011.

Babar, Muhammad Ali and , and Brown, Alan W. and , and Mistrik, Ivan. "Chapter 7 - Continuous Software Architecture Analysis." In Agile Software Architecture, 161-188. Boston: Morgan Kaufmann, 2014.

Baek Jorgensen, J. and Bossen, C. "Executable use cases: requirements for a pervasive health care system." Software, IEEE, 2004: 34-41.

Beck, Fabian and Diehl, Stephan. "Visual comparison of software architectures." Proceedings of the 5th international symposium on Software visualization, 2010: 183--192.

Dumas, M., van der Aalst, W.M.P., and ter Hofstede. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Process-Aware Information Systems: Bridging People and Software through Process Technology. Wiley &Sons, Chichester, 2005.

Emadi, S, and F Shams. "Computer and Electrical Engineering, 2009. ICCEE '09. Second International Conference on." Mapping Annotated Use Case and Sequence Diagrams to a Petri Net Notation for Performance Evaluation (Mapping Annotated Use Case and Sequence Diagrams to a Petri Net Notation for Performance Evaluation), 2008: 67-81.

Emadi, S. and Shams, F. "From UML component diagram to an executable model based on Petri Nets." Information Technology, 2008. ITSim 2008. International Symposium on, 2008: 1-8.

Feldgen, M. , Clua, O. "Promoting design skills in distributed systems." Frontiers in Education Conference (FIE), 2012: 1-6.

Gardazi, S.U. and Shahid, A. "Survey of software architecture description and usage in software industry of Pakistan." Emerging Technologies. ICET . International Conference on, 2009: 395-402.

Georgakopoulos, D., Hornick, M., Sheth, A. "An Overview of Workflow Manage-ment: From Process Modeling to Workflow Automation Infrastructure." Distributed and Parallel Databases 3, 1995: 119–153.

Gervasi, Osvaldo ,Taniar, David, Murgante, Beniamino and Laganà, Antonio , Mun, Youngsong , Gavrilova, MarinaL. "Verification of Use Case with Petri Nets in Requirement Analysis." Computational Science and Its Applications – ICCSA, 2009: 29-42.

Gomaa, H. "Advances in Software Design Methods for Concurrent, Real-Time and Distributed Applications." Software Engineering Advances, . ICSEA '08. The Third International Conference on, 2008: 451-456.

Gorton, M. A. Babar and I. "Software Architecture Review: The State of Practice." Computer 42, no. 7 (2009): 26-32.

Haroonabadi, A. and Teshnehlab, M. and Movaghar, A. "A Novel Method for Modeling and Evaluation of Uncertain Information Systems." Information Technology, ICIT '08. International Conference on, 2008: 238-243.

Heft, von Prof. Dr. Robert Gold. "Petri Nets in Software Engineering." 5 aus der ReiheArbeitsberichte - Working Paper, 2004: 1612-6483.

Hua Duan, Qingtian Zeng ,Huaiqing Wang , Sherry X. Sun. "Classification and evaluation of timed running schemas for workflow based on process mining." Journal of Systems and Software, 2009: 400 - 410.

Jeffrey J.P. Tsai, Alan Liu. "Experience on knowledge-based software engineering: A logic-based requirements language and its industrial applications." Journal of Systems and Software, 2009: 1578-1587.

Jensen, Kurt and Kristensen, Lars M. "CPN ML Programming." In Coloured Petri Nets: Modelling and Validation of Concurrent Systems, 43-77. Berlin: Springer Berlin Heidelberg, 2009.

Jensen, Kurt, and WilM.P Aalst. "Model-Based Software Engineering and Process-Aware Information Systems." Transactions on Petri Nets and Other Models of Concurrency II, 2009: 27-45.

Kaymak, U. , Mans, R. ,van de Steeg, T., Dierks, M. "On process mining in health care." Systems, Man, and Cybernetics (SMC), IEEE International Conference on, 2012: 1859-1864.

Khadka, Binsan and Mikolajczak, Boleslaw. "Incorporating Object-Orientedness in Transformations from Live Sequence Charts to Colored Petri Nets." Proceedings of the Fifth International Conference on Information Technology: New Generations, 2008: 1179 - 1183.

Kijas, Andrzej Zalewski and Szymon. "Beyond ATAM: Early architecture evaluation method for large-scale distributed systems." Journal of Systems and Software 86, no. 3 (2013): 683 - 697.

Kirsten Berkenkötter. "Reliable UML Models and Profiles." Electronic Notes in Theoretical Computer Science, 2008: 203 - 220.

Kurt Jensen, Aalst, Wil M. "Process-Aware Information Systems: Lessons to Be Learned from Process Mining." Transactions on Petri Nets and Other Models of Concurrency II, 2009: 1-26.

Kurt, lars. "Introduction to Modeling and Validattion." In Coloured petri Nets ,Modeling and Validation of Concurrent System, 8-10. New York: Springer, 2009.

Lauw, Hady,Lim, Ee-Peng,Pang, HweeHwa,Tan, Teck-Tim. "Social Network Discovery by Mining Spatio-Temporal Events." Computational & Mathematical Organization Theory (Kluwer Academic Publishers) 11, no. 2 (7 2005): 97-118.

Lemos, A.M. ,Sabino, C.C.,Lima, R.M.F. ,Oliveira. "Using process mining in software development process management: A case study." Systems, Man, and Cybernetics (SMC), IEEE International Conference on, 2011: 1181-1186.

Lindvall, Mikael and Tvedt, Roseanne Tesoriero and Costa, Patricia. "An Empirically-Based Process for Software Architecture Evaluation." Empirical Softw. Engg 8, no. 1 (2003): 83-108.

Mangano, N., Dempsey, M. and Lopez, N., Van der Hoek, Andre. "A demonstration of a distributed software design sketching tool." Software Engineering (ICSE), 33rd International Conference on, 2011: 1028-1030.

Medvidovic, N.,Taylor, R.N. "A classification and comparison framework for software architecture description languages." Software Engineering, IEEE Transactions on, 2000: 70-93.

Meier, P., Kounev, S., Koziolek, H. "Automated Transformation of Component-Based Software Architecture Models to Queueing Petri Nets." Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE 19th International Symposium on, 2011: 339-348.

Michael E. Shin, Hassan Gomaa. "Software requirements and architecture modeling for evolving non-secure applications into secure applications." Science of Computer Programming, 2007: 60-70.

Michel dos Santod Soares, Stephan Julia , Jos Varncken. "Real-time scheduling of batch systems using Petri nets and linear logic." Journal of Systems and Software, 2008: 1983 - 1996.

Muehlen, zur. Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems. Berlin: Logos, 2004.

Pettit, R.G., Gomaa, H. "Modeling behavioral patterns of concurrent software architectures using Petri nets." Software Architecture,. WICSA Proceedings. Fourth Working IEEE/IFIP Conference on, 2004: 57-66.

Qureshi, Zahid H. Formal Modelling and Analysis of Mission-Critical Software in. Sidny: Australian Defence Force, 2005.

Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A.K., Song, M., Verbeek, H.M.W. "Business Process Mining: An Application." Information Systems, 2007: 713–732.

Rodriguez-Priego, Emilio and García-Izquierdo, FranciscoJ. and Rubio, ÁngelLuis. "Modeling Issues: a Survival Guide for a Non-expert Modeler." Model Driven Engineering Languages and Systems, 2010: 361-375.

Sahlabadi, M., R.C. Muniyandi and Z. Shukur. " Detecting abnormal behavior in social network websites by using a process mining technique. ." Comput. Sci 10 (2014): 393-402.

ShangGuan We, Cai Bai-gen and Wang Jian,Wang Yan , Gou Chen-xi. "Research of System Modeling and Verification Method Combine with UML Formalization Analysis and Colored Petri Ne." Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on, 2009: 488-491.

Sharafi, Mehran and Shams Aliee, Fereidoon and Movaghar, Ali. "A Review on Specifying Software Architectures Using Extended Automata-Based Models." International Symposium on Fundamentals of Software Engineering, 2007: 423-431.

Simonetta Balsamo, Peter G. Harrison,Andrea Marin. "Methodological construction of product-form stochastic Petri nets for performance evaluation." Journal of Systems and Software, 2012: 1520 - 1539.

Staines, A. Spiteri. "Supporting Requirements Engineering with Different Petri Net Classes." International Journal of Computers, Issue 1 vol 2, 2010,: 215-222.

Thouraya Bouabana, Tebibel and Stuart H. Rubin. "An interleaving semantics for UML 2 interactions using Petri nets." Information Sciences, 2013: 276 - 293.

van der Aalst, W.M.P., van Hee, K.M. "Workflow Management: Models, Methods,system." MIT Press, 2004: 45-47.

van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G.,Weijters, A.J.M.M. "Workflow Mining: A Survey of Issues and Approaches." Data and Knowledge Engineering, 2003: 237–267.

Virbitskaite, Irina and Voronkov, Andrei. "Sarstedt, Stefan and Guttmann, Walter." Perspectives of Systems Informatics, 2007: 349-362.

W.M.P. van der Aalst a, M.H. Schonenberg, M. Song. "Time prediction based on process mining." information System, 2011: 450-475.

Wensong Hu, Yang a, Ke Zuo Xingui. "Based Aspect-oriented Petri Nets in Software Engineering." Physics Procedia, 2011: 646-650.

Zee, Durk-Jouke van der. "Building insightful simulation models using Petri Nets — A structured approach." Decision Support Systems, 2011: 53-64.

Mahdi Sahlabadi
Amirhossein Sahlabadi
Ravie Chandren Muniyandi
Zarina Shukur
Fakulti Teknologi dan Sains Maklumat
Universiti Kebangsaan Malaysia
Bangi, Selangor, Malaysia
ahlabadi2002@gmail.com, amirsahl1991@gmail.com, ravie@ukm.edu.my,
zarinashukur@ukm.edu.my

Received: 28 February 2017
Accepted: 4 October 2017
Published: 4 December 2017