International Journal of Computer Applications Technology and Research
Volume 7–Issue 05, 193-199, 2018, ISSN:-2319–8656

Availability Assessment of Software Systems Architecture Using Formal Models

Mahbubeh Sadeghian
Komail Institute of Higher Education
Kordkooy, Iran

Homayun Motameni
Department of Computer Engineering
Islamic Azad University, Sari
Iran

Abstract: There has been a significant effort to analyze, design and implement the information systems to process the information and data, and solve various problems. On the one hand, complexity of the contemporary systems, and eye-catching increase in the variety and volume of information has led to great number of the components and elements, and more complex structure and organization of the information systems. On the other hand, it is necessary to develop the systems which meet all of the stakeholders' functional and non-functional requirements. Considering the fact that evaluation and assessment of the aforementioned requirements - prior to the design and implementation phases - will consume less time and reduce costs, the best time to measure the evaluable behavior of the system is when its software architecture is provided. One of the ways to evaluate the architecture of software is creation of an executable model of architecture.
The present research used availability assessment and took repair, maintenance and accident time parameters into consideration. Failures of software and hardware components have been considered in the architecture of software systems. To describe the architecture easily, the authors used Unified Modeling Language (UML). However, due to the informality of UML, they utilized Colored Petri Nets (CPN) for assessment too. Eventually, the researchers evaluated a CPN-based executable model of architecture through CPN-Tools.

Keywords: Software Architecture; Unified Modeling Language; Colored Petri Nets; Availability.

1. INTRODUCTION

Due to the recent increase in the complexity, volume, number of users, use, and amount of investment, it is necessary to design and develop software system in a way which simultaneously reduces the investment cost and meets the developers and users' needs. The intrinsic complexity of the software system is caused by factors like complexity of the subject matter and manufacturing process, flexibility, and difficult non-standard behaviour of complex system [1]. Although the complexity cannot be omitted, it must be controlled by using some methods. In addition, checking the specifications of a system - prior to design and implementation phases - will save time and lower costs. Therefore, the best time to measure the evaluable behaviour of a system is when the architecture of the system is provided. Software architecture, which focuses on components and connectors as the most important elements and components, creates a relationship between structure and behaviour of components. A set of components with their own properties interact with each other through the connectors. The connectors have their own specifications too. These will form software system architecture within a specific configuration [2]. To attain this goal scientists and developers use modelling. This paves the way for creation and changing the software easily and cost-effectively. A model approximately presents properties of a real system. Therefore, by utilising a model it is possible to evaluate a system prior to its implementation. Sometimes it is economically impossible to build a system without considering the design and elementary output. [3]. Models, through executable representation of the properties of the architecture, display run-time behaviour of systems. These behaviours make the evaluation of many quality attributes possible. There are various models to display executable architecture including Petri Nets, queuing network, stochastic process algebra and etc. Also, some architecture description languages can demonstrate an executable architecture.
Availability is one of non-functional requirements that affect software development. The reason is that availability problems can cause significant changes in all of the stages of software life cycle. This is particularly observable in the initial stages of the development.
Here, the availability is among the factors which greatly influence the customers' selection and is in direct relationship with the failure and faults of a system and its consequences. In fact, failure occurs when a system is not able to deliver a service based on its characteristics and attributes. Several works focused on non-functional requirements, especially availability in the initial stages of software architecture and development. However, these methods have limited application in real and complicated examples. Also, none of these methods evaluates the requirements entirely. To assess this metric, the authors proposed method includes CPN.
The second part of the article includes the fundamental concepts of software architecture, UML, CPN, and availability. The third part contains selected review of previous studies. The fourth part embraces the detailed description of the proposed method. The fifth part represents a case study for evaluation and modelling of the proposed method. This is done to demonstrate the correctness of the method.

2. BASIC CONCEPTS

Regarding their type of application and performance, concepts are defined differently. This method was followed in the present article.

2.1 Software Architecture

In fact, software architecture expresses a structure of a set of solutions for a problem. Decomposition and breaking

the space of a problem into smaller sections paves the way for designation of general characteristics of each part, and finding solution for a set of parts with common specifications.
Therefore, designing of software structure is a crucial and necessary stage in the development of large and complicated software. As shown in Figure 1, requirements are the main input of software architecture design. Requirements include the users and stakeholders' desired properties and specifications of a system [4].

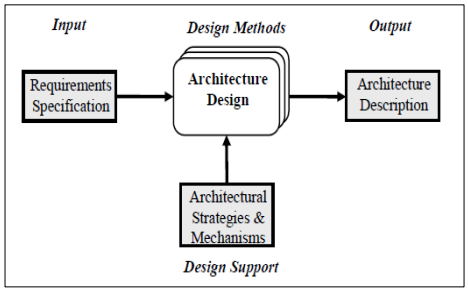

Figure 1. Software Architecture View

Software architecture includes the initial decisions for designing a system. These are the fundamentals of system design, implementation, deployment, and maintenance. Software architecture is the first evaluable element in the process of software development. So, the first step to design a system which meets the quality requirement is producing the software structure.

2.2 Unified Modelling Language

Presenting a model of a system paves the way for studying a system from intended aspects. To develop information systems, various methods have been proposed. They can be widely categorised into structured and object-oriented. The lack of UML in the structured methods causes deficiencies in the readability and efficiency of them. This problem was resolved in the object oriented methods through unified modelling language. UML is a powerful language which supports all of the object oriented concepts. Therefore, it is simultaneously used by the object oriented methodologies and databases. This language has introduced a powerful set of modelling elements, charts and Pre-defined structures to describe the structural characteristics and behavioural of software architecture [5]. Due to the nature of the requirements, uncertainty in information system is inevitable. To develop systems, object oriented methodologies express their intended concepts through UML.

2.3 Coloured Petri Nets

Coloured Petri Nets have been introduced as a developed model of Petri Nets. In addition to places, transitions and tokens, these networks introduced concepts of statement, guard and colour. CPN use simple abilities of petri nets and programming languages. (Figure 2)

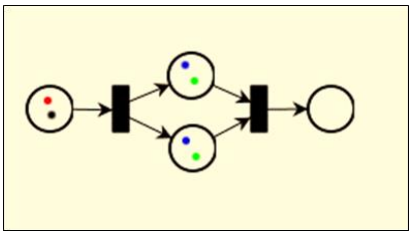
Figure 2. A Model of Coloured Petri Nets'

Tokens carry data values in these networks. Coloured Petri nets provide more accurate models of complex Asynchronous processing systems. In these networks, unlike petri nets, tokens are distinguishable because each token has an attribute called colour [6]. To put it in another way, they offer a clear graphical view of the system with a mathematical approach and indicate communication patterns, control patterns and information flows. These networks present a framework for analysis, validation and performance evaluation. Petri nets are based on graph. Informally, a directed graph consists of two elements: place and transition. These networks are based on state not event. This makes the explicit modelling of each case possible. Petri nets represent models of structural and behavioural aspects of a discrete event system. They also provide a framework for analysis, validation, performance evaluation and reliability [7]. The reason behind using CPN instead of simple petri nets is that in the former, tokens can have different values in a place. So if the tokens are in the same place, it would be possible to distinguish them from each other [6].

2.4 Availability

In fact, the concept of availability is developed for repairable systems that require continuous work. At each random moment, these systems are whether working or out of order. In the latter case, they are trying to recover and restart functioning in the minimum time. Generally, a system has got two possible states: operating or down. Availability is defined as the probability that one system works desirably at any point of time *t* and builds a cycle as “operational” and “downtime” states. In other words availability is a compound of reliability and maintainability parameters [8].
Availability is one of the non-functional requirements. It is computed in part of system that is in process and shows proportion of time that the system is running, executing, and alive. This quality attribute shows the extent to which the system is able to resolve its problems and return to its natural state.

3. PREVIOUS STUDIES

In recent years, various methods have been proposed to assess the availability of the software systems. These methods have been discussed in various studies. Some of these methods have been studied in this paper. Reference [8] discussed availability assessment based on safety in design stage. This method was used in the present research for availability assessment and will be explained in the following sections. Here, we presumed the access to the sequence diagrams of the system. Because it is impossible to perform availability assessment through UML diagrams, we need to do a mapping of sequence diagrams on petri nets. After production of corresponding petri nets

via UML sequence diagrams, it was implemented in CPN Tools.

In [9] a model, which is based on measuring the level of system, is presented through an example of the safety protection system. Different types of failures are discussed too. In [10] the relationship between software availability measurement and the number of restorations with imperfect debugging has been studied. This paper proposed a software availability model considering the number of restoration actions. Through this model, quantitative measurement of software availability assessment is achievable. This is based on the number of restoration actions.

Also different methods and availability models which can be used in the industry are discussed in [11], [12], [13] and [14].

In [15] UML activity diagram was converted into CPN through an automatic method. This paper indicates that as the result of semantic difference between CPN and UML, direct transforming from UML to CPN is done manually. It also shows transforming method that can automatically convert UML activity diagram into CPN. To overcome the semantic difference between activity diagram and CPN, an intermediate XML formatted model was used. The present article demonstrates the conversion of UML 2.0 diagram into CPN through construction of an intermediate model. This is utilised to evaluate the performance of software.

[16] Presents a method of evaluation through using discrete time Markov model. This method computes the reliability of software architecture based on the reliability of each component and the probability of transitions. In order to evaluate quantitatively, [17] proposed a method which is based on dynamic Bayesian network. To evaluate performance quantitatively, [18] developed an approach based on discrete time Markov model. This approach computes the performance of software architecture considering the consumed time in each component and number of the times that they meet each other in the process of running a programme. [19] Presents a method for quantitative assessment of the security. This method has its basis in discrete time Markov model and calculates the security of software architecture based on the vulnerability of components and the frequency that they meet each other in the process of running a programme.

[20] Represents a way to create a formal model of the UML diagrams. This research includes a methodology to create a formal model of the system with the purpose of analysis and behavioural modelling of UML. In fact UML state diagrams, which illustrate a sequence of states of an object during its lifecycle, and UML collaboration diagrams are converted into object petri nets. Then they assess verification of UML behavioural properties through presented formal model to detect and reveal behaviours based on concurrency such as deadlock. Moreover, [21] provided a way to assess the authenticity of behaviour and validation of UML sequence diagrams. This method use source and destination of messages in sequence diagram and these diagrams are mapped into PROME Language. Then SPIN tool was used for simulation.

4. PROPOSED METHOD

In this paper the authors used CPN and CPN Tools for modelling and availability assessment. Most of the previous studies used Markov model but in this paper CPN were used. In other words, compared to Markov model, describing a problem via a petri net is less complicated. This is also closer to the designer's understanding of the system description. Petri nets can be applied to overcome the deficiencies of the Markov model. In Markov model, small changes in design of the system lead to many changes in Markov chain structure. Therefore, development of the systems that use Markov model is difficult. However, it is easily possible to develop the systems which are modelled by petri nets. The following section describes the method.

4.1 Assessment of Quality Properties via CPN

Figure (3) shows the process of evaluation of quality properties through values attributed to the tokens. Here, the value of *x* on transition indicates the value of a quality property. There are two probabilities for firing; firing *t1* transition and firing *t2* transition. After the t1 transition firing, the value of *x* is updated to *f(x)* which means the value of a quality property has changed to the value of *f(x)* in one stage of progress.

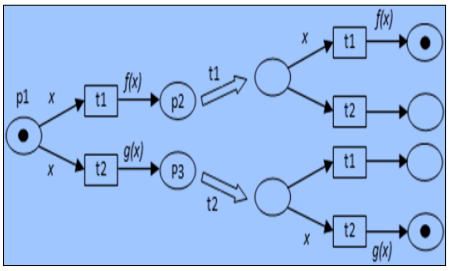

Figure 3. Evaluation of the Quality Properties in CPN [6]

4.2 Availability Assessment

To compute resources availability we have mean time between resource failure and it's modification in a time interval. We can obtain considered metric based on the availability of the resources. Availability is divided into intrinsic and operational types (Figure 4). The former is a function of mean time to repair (MTTR) and mean time between failures (MTBF) [8].

$$A_{in} = \mu/(\mu + \lambda) \quad (1)$$

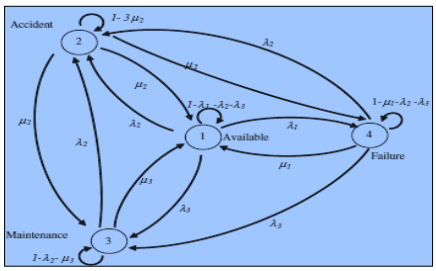

Figure 4. General Model of Availability

In equation (1), $\mu = 1/\text{MTTR}$ and $\lambda = 1/\text{MTBF}$ represent repair rate and failure rate respectively. Operational availability equals the time period attributable to administrative procedures.

$$A_{op} = \text{MTBF}/(\text{MTBF} + \text{MTTR} + \text{TAP}) \quad (2)$$

TAP is dependent on the policies of enterprise maintenance organization and procedures like constraints of ISO-900x. These are not available in the design phase. However, safety-based operational availability, which is the authors' proposed idea includes:

MTTR : mean time to repair
MTTM : mean time to preventive maintenance
MTTP : mean time to prepare the system for operation after an accident or incident.
Failure Rate : $\lambda_1 = 1/\text{MTBF}$
Accident Rate : $\lambda_2 = 1/\text{MTBA}$,
Preventive Maintenance rate : $\lambda_3 = 1/\text{MTBM}$, Repair Rate : $\mu_1 = 1/\text{MTTR}$,
Prepare Rate : $\mu_2 = 1/\text{MTTP}$ and
Rate of Repair and Maintenance : $\mu_3 = 1/\text{MTTM}$. [8]

4.3 UML Conversion Algorithm to CPN

The authors used UML language and sequence diagrams to describe and present behaviour of architecture respectively. The aim of this step is converting sender and receiver of messages into petri nets. A sequence diagram in performance evaluation shows how a client from a particular type moves between services centres. It is possible to consider a colour for a token in petri nets for every set of scenario which is assigned to a customer. Later, all of the message senders and receivers, and message between them must be converted into petri nets. This diagram emphasizes on the communication pattern between the components, (i.e. the interaction between components) and is plotted due to the messages sending time. This paper used the converting method on messages sender and receiver object and variety of messages in sequence diagram such as sequence, selection, concurrency and iteration. In this case each of the messages sender and receiver components will be transformed to place-transition-place [22].

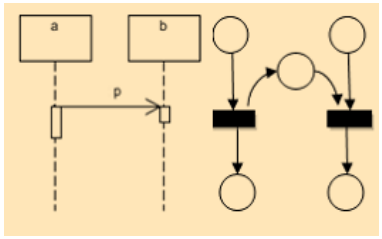

Figure 5. Asynchronous Messages and its Equivalent Petri Net

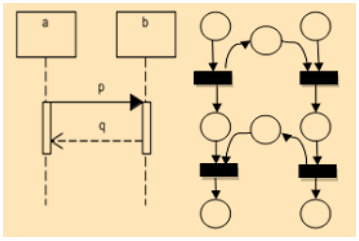

Figure 6. Synchronous Messages and its Equivalent Petri Net

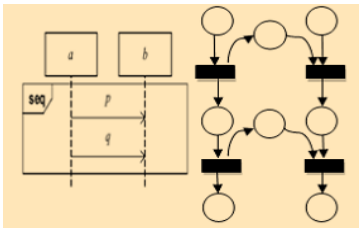

Figure 7. Sequence Structure and its Equivalent Petri Net

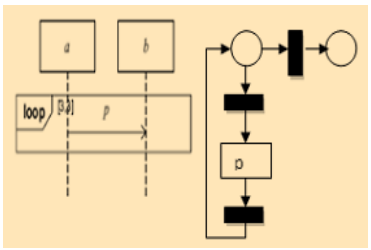

Figure 8. Iteration Structure and its Equivalent Petri Net

4.4 Creation of an Executable Model

The first step to create an executable model is using of hierarchical structure of coloured petri net. So, for CPN a transition will replace compound parts. Next, a subpage will be defined for every transition. In addition, a presented algorithm was used to transform sequence diagram to CPN. The followings are the steps to create an executable model of availability assessment through sequence diagram:
First, we act hierarchically to create an executable model via CPN. Here, we substitute a compound part in a sequence diagram for a replacing transition and define a subpage for every replacing transition.
Second, we use the presented algorithm for transforming sequence diagram to CPN.
Third, to implement the transition in CPN Tools, we present the properties and requirements of the sequence diagram in Standard ML.
Fourth, to evaluate whether the plotted sequence diagram represents the stated properties correctly, all of them must be checked with the consideration of all possible executing states in the sequence diagram. For example, figure (9) shows CPN for sequence diagram of withdrawal from ATM in CPN Tools.

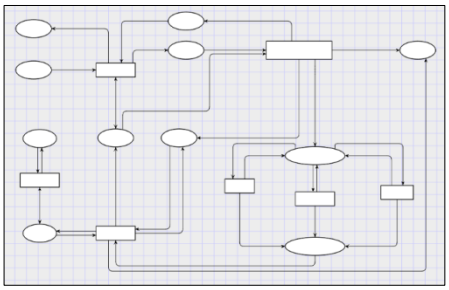

Figure 9. Executable Model of Withdrawal from ATM

After creation of an executable model via a petri net, it is viable to run this model in CPN Tools Software. Based on the obtained information, the behaviour of architecture products will be approved or rejected. In the latter case, the product must be modified.

5. CASE STUDY

To prove the applicability and validation of the model, a case study is evaluated. In case studies, we look for small comprehensive examples which simply represent real world samples. Due to its simplicity, the example of withdrawal from an ATM is easily understood. The following section includes simulation of an executable model in CPN Tools, availability assessment of architecture, and the obtained results. Figure 10 presents a general view of the presented model. This is based on

the defined data. Every unit of this system has the potential to fail. This demonstrates the vitality of the availability metric for the system. Sequence diagram for ATM withdrawal is shown in figure 11. Sequence diagrams are useful designing tools because they provide dynamic view of the system behaviour.

The modelling process is done in a way that specifies the number of requests. Then, the system will be run. In addition, it is possible to stop the operation at any given time and observe the amount of availability which is shown in percentage. Figure 12 presents part of the petri net model equivalent to sequence diagram in CPN Tool.

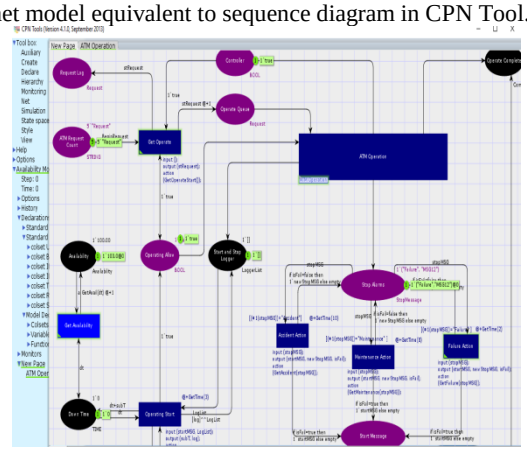

Figure 10. General View of the Presented Model

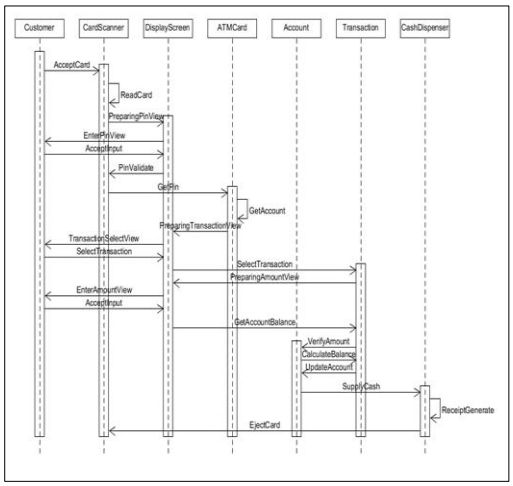

Figure 11. Sequence Diagram of ATM Withdrawal

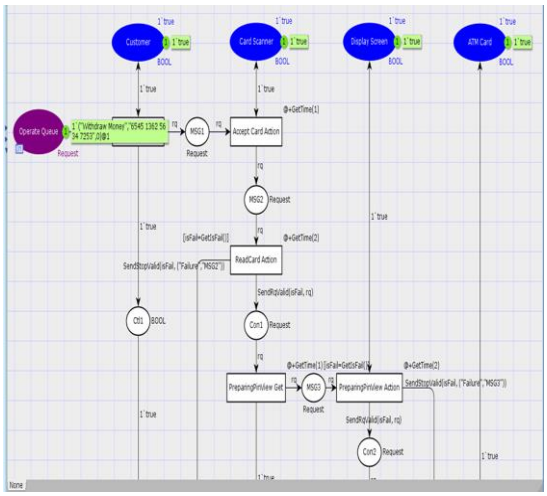

Figure 12. Part of Petri Net Model equivalent to ATM Sequence Diagram

Figure 13 embodies the results of the proposed method with 10 requests. In addition, this model has the potential to include fluctuations in number of requests regarding the needs.

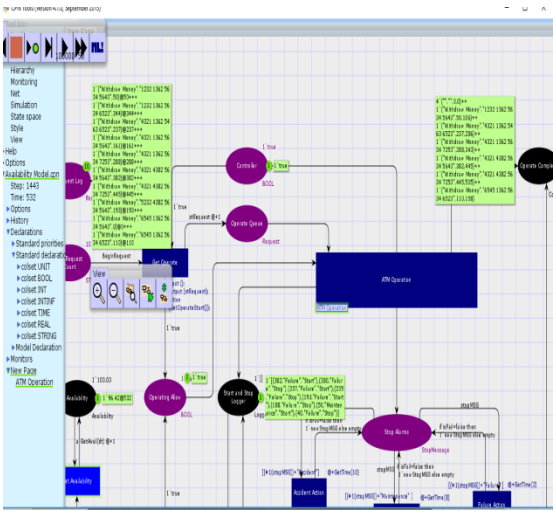

Figure 13. Output of Model for 10 Requests

Table 1 shows the executed model with 10 requests.

Table 1. Output of model with 10 requests

Type of operation	Card number	Start time	Request completion time
Withdraw money	1232 1362 5634 5643	50	106
Withdraw money	1232 1362 5634 6523	344	---
Withdraw money	4321 1362 5463 6523	237	286
Withdraw money	4321 1362 5634 5643	161	---
Withdraw money	4321 1362 5634 7253	288	343
Withdraw money	4321 4382 5634 5643	382	445
Withdraw money	4321 4382 5634 7253	445	505
Withdraw money	5232 4382 5634 5643	193	---
Withdraw money	6545 1362 5634 5643	0	---
Withdraw money	6545 1362 5634 6523	110	158

Considering the results in a single run with ten requests, some of requests were not completed .In other words, four requests did not reach the end.

Table 2. Output of the Executed Model with 10 Requests and 4 Failures

Time	Event	Reason of event
40	failure	Stop
50	Maintenance	Start

188	Failure	Stop
193	Failure	Start
235	Failure	Stop
237	Failure	Start
380	Failure	Stop
382	Failure	Start

Results of "start and stop logger" are shown in table (2). Figure 14 illustrates the amount of availability of the system using presented method. Diagram of requests and completion time are presented in figure 15. As it is observable, four requests were not completed.

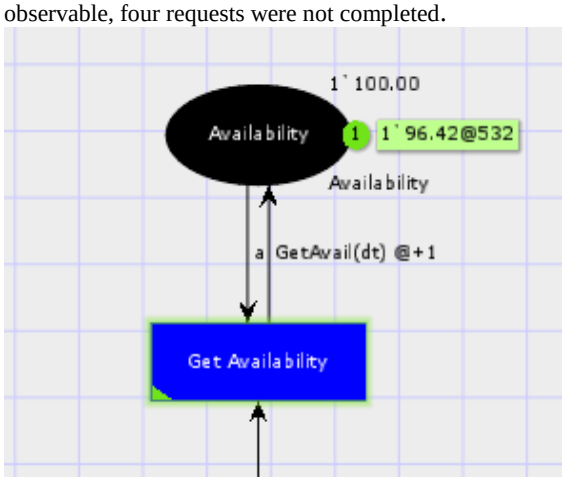

Figure 14. Amount of System Availability

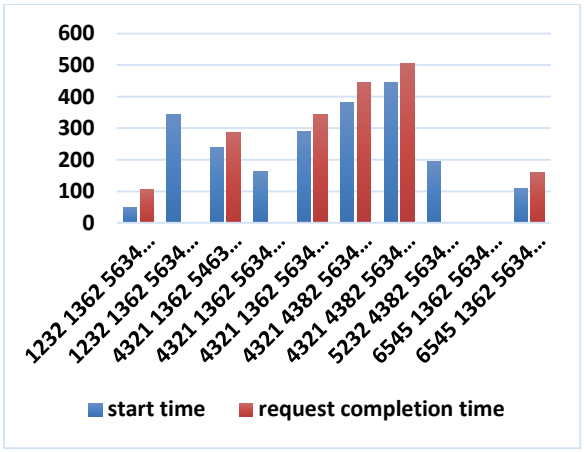

Figure 15. Diagram of 10 Requests and their Completion

6. CONCLUSION

The present article focused on the availability assessment in level of software systems architecture and took the features of the software and hardware platform into consideration. Moreover, a CPN-based executable model was represented. These networks have strong mathematical background. The main objective of this paper is to provide an executable model to evaluate the availability of the architecture of software systems with input from the maintenance time and the time of the accident. In this paper, the authors used sequence diagram to describe behaviour of software systems architecture. Eventually, a CPN-based executable model was achieved via proposed method. In fact, this study provided a way to create an executable model which takes availability with input from the times of maintenance and accident into consideration. The results of the simulation of this method enable us to identify the deficiencies in the planning phase and modify our products. This saves time and reduces costs to a large extent.

REFERENCES

[1] Cortellessa V., Marco A.D. & Inverardi P., (2004), "Three performance models at work: A software designer perspective", Electronic notes in theoretical computer science, vol. 97, pp. 219–239.

[2] Clements P. C., Kazman R. & Klein M. H., (2001), "Evaluating software architecture methods and case studies", Addison-Wesley Professional.

[3] Galster M., Moussavi M. & Eberlein A., (2006), "Transition from requirements to architecture: A review and future perspective", Proceedings of the seventh ACIS international conference on software engineering, artificial intelligence, networking, IEEE, pp. 9-16.

[4] Thiel S., (2005), "A framework to improve the architecture quality of software-intensive systems". Approved dissertation of, The university of Duisburg-Essen, for obtaining the academic degree

[5] Cortellessa V. & Pompei A. (2007), "Towards a UML profile for QoS: A contribution in the reliability domain", WOSP '04 proceeding of the 4th international workshop on software and performance, vol. 29 pp. 197-206.

[6] Fukuzawa K. & Saeki M., (2002), "Evaluating software architectures by coloured PetriNets", Proceedings of the 14th international conference on software engineering and knowledge engineering, pp. 263-270

[7] Jensen K. (1997). "Coloured Petri Nets: Basic concepts, Analysis methods and practical use", EATCS monographs on theoretical computer science, vol. 2, no .2, pp70-120.

[8] Remy H. & Amadou C. (2014), "Safety-based availability assessment at design stage", Computers & Industrial Engineering, vol 70, pp. 107–115.

[9] Hecht M., Tang D., Hecht H. & Brill R. W. (1997) "Quantitative reliability and availability assessment for critical systems including software", Computer assurance, COMPASS '97, Are we making progress towards computer assurance? Proceedings of the 12th annual conference on.

[10] Tokuno K. & Yamada S. (2003), "Relationship between software availability measurement and the number of restorations with imperfect debugging", computers and mathematics with applications, vol. 46, no. 7, pp. 1155-1163.

[11] Ahmed Q., Khan F., Ahmed S. (2014), "Improving safety and availability of complex systems using a risk-based failure assessment approach", Journal of Loss Prevention in the Process Industries, vol. 32 pp. 218-229.

[12] Choi H. & Chang D., (2016), "Reliability and availability assessment of seabed storage tanks using fault tree analysis", Ocean Engineering, vol. 120, pp. 1–14.

[13] Zio E., Baraldi P. & Patelli E., (2006), "Assessment of the availability of an offshore installation by Monte Carlo simulation", International journal of pressure vessels and piping, vol. 83, no. 4, pp. 312–320.

[14] Zitrou A., Bedford T. & Walls L. (2016), "A model for availability growth with application to new generation offshore wind farms", Reliability engineering and system safety, vol. 152, pp. 83–94.

[15] Zhu L. & Wang Y., (2012), "From UML activity diagrams to CPN: An automatic transforming method", IEEE, 7th international conference on computing and convergence Technology (ICCCT).

[16] Cheung R. C. (1980), "A user-oriented software reliability model", IEEE Transactions on Software Engineering, vol. 6, pp. 118-125.

[17] Roshandel R., Medvidovic N. & Golubchik L. (2007), "A Bayesian model for predicting reliability of software systems at the architectural level", Software architectures, Components and applications, Springer-Verlag, pp. 108-126.

[18] Gokhale S. S., Wong W. E., Trivedi K. S. & Horgan J. R., (2004), "An analytical approach to architecture-based software performance and reliability prediction", Performance evaluation, Elsevier science publishers B. V. Amsterdam, The Netherlands, vol. 58, no. 4, pp. 391-412.

[19] Sharma V.S. & Trivedi K.S. (2009), "Quantifying software performance, reliability and security: An architecture-based approach", Journal of Systems and Software, vol. 80, no. 4, pp. 493-509.

[20] Yahia E., Aubry A. & Paneto H., (2012). "Formal measures for semantic interoperability assessment in cooperative enterprise information systems", Computers in Industry, Vol. 63, no. 5, pp. 443-457.

[21] Laxman P. B., (2013), "Validation of UML models for interactive systems with CPN and SPIN", MTech thesis.

[22] Cheung L., Roshandel R., Medvidovic N. & Golubchik L., (2008). "Early Prediction of Software Component Reliability", ICSE 08, Proceedings of the 30th international conference on Software engineering, Leipzig, Germany, pp. 111-120.