JOURNAL OF MANAGEMENT AND ACCOUNTING STUDIES 2020(04)

Available online at http://journals.researchub.org

Evaluating the availability of a mobile payment model in e-commerce using the colored Petri net

Abdolghader Pourali*

*Department of computer, Abadan Branch, Islamic Azad University, Abadan, Iran

ARTICLE INFO

Article history:

Received 10 May 2020

Received in revised form 13 Aug 2020

Accepted 20 Sept 2020

Keywords:

Colored Petri Net,

Availability,

Mobile Electronic Payment,

Integrated Modeling Languag

ABSTRACT

Availability is an important feature of a mobile payment system in the banking industry. It should be also capable of providing services to the users throughout the week without interruption. In general, a payment system in mobile commerce should have high reliability and availability to enable users and stakeholders to do business transactions, buying and selling and pay for financial transactions with a high degree of reliability. Since an architecture or a payment model in e-commerce only displays the way of interaction and collaboration between users and mortgagors and provides no assessment of the performance, reliability and availability of the system to the stakeholders, thus, in this study, we managed to provide an accurate assessment of the availability of a mobile payment model at the software architecture level by employing a colored petri dish using the official models. The methodology is as follows: First, we converted the studied model into an integrated modeling language (UML). However, the semi-official nature of this language has made it unable to assess all non-obligatory needs. To achieve this goal, an applicable model must be created from these graphs. Finally, by converting the applicable model into the official language and using the Petri Net networks and "CPN Tools" tool, an accurate assessment of the availability level would be provided.

1. Introduction

With the increasing spread of information technology and the urgent need for software, especially in sensitive industries such as the banking system, safety, aerospace and medical industries, recognizes as major and vital industries, the availability of each of these systems has undoubtedly drawn the attention of the experts in this area as a sensitive subject. Accessibility or availability is introduced as one of the most important indicators of non-obligatory computer system requirements in addition to other qualitative features such as reliability, efficiency, performance, etc. for quality evaluation (Gorton, 2006). It examines the failure rate of the system and its associated implications. System failure occurs when the system fails to provide its default services properly. Nowadays, with the advent of technology and information technology, software architecture researchers are also trying and working hard to develop better software systems and have gained some achievements in this regard. In fact, it would better that the creation of high-reliability software systems occurs by modeling architectural patterns in the early phases of the software development lifecycle. Software architecture is a great indicator of quality properties. Choosing the patterns to be modeled and how to model them, and ultimately, their evaluation are important decisions that the researchers of software architecture have focused on them. In fact, by software modeling, the developer can design the software before allocating additional resources and return to the pre-design stage, i.e., determining the requirements, and ensure the generation of a reliable system (Pourali, 2017). In addition to describing the software and fragmenting it into components, software architecture styles have a major impact on the qualitative features of the designed software. Therefore, choosing an appropriate architectural style for the system appears to be highly essential (Bass et al., 2010). As wrong decisions, in addition to preventing the achievement of desirable qualitative attributes, will lead to the loss of huge cost and time used for the design. The best time to measure the measurable behavior of a system is when the software architecture of that system has been created. The software architecture can be assessed as the first measurable behavior of the system, i.e., non-tasking needs such as security, availability, usability, changeability, reliability, statics, and efficiency (Clements et al., 2002; Avgeriou1 et al., 2004). In this research, we tried to choose a model of a mobile payment system to assess the availability level before implementation (Pourali, 2014). To this end, the study style should be applicable. Thus, the UML charts were used. In this study, we used sequential diagrams to illustrate the architectural behavior. In the next step, since the UML graph was not applicable, by marginalizing the considered metric (availability) through labeling the stereotypes, the trend was continued. Finally, the study style chart was converted to the applicable model. After drawing the Petri model with the help of cpntools software, the availability level was calculated.

**Corresponding author:* Abdolghader.pourali@gmail.com

DOI: https://doi.org/10.24200/jmas.vol8iss04pp1-12

2. Definition and concepts of availability

Before providing a definition for availability, we begin by defining the reliability (Avizienis et al., 2001; BS EN 60300-2, 2004). As availability is one of the attributes of reliability. Reliability is the ability of the system to avoid failures that are greater than the acceptable degree or severity and the acceptable occurrence period time. Availability is one of the main features of a reliable system (BS EN 60300-4, 2008; Dubrova, 2005). In other words, the availability is the system's operational probability at the time of need. That is, availability is associated with a fraction of the system time that is active and running (Avizienis et al., 2001; Tanenbaum & Van Steen, 2007). This definition can also be expressed by relation (1) (Pourali, 2017).

Availability = MTTF / (MTTF + MTTR) (1)

Min Time-to-Failure is the average error time and Min Time-to-Repair is the average correction time. Availability has a close relationship with reliability. Thus, higher reliability in a system will lead to its more availability, and vice versa (Tanenbaum & Van Steen, 2007). Reliability refers to the ability of a system to continue operations over time. The parameters involved in relation (1) both depend on the architecture. The average error time increases mainly due to the creation of a high-tolerant architecture. However, the error tolerance is achieved by repeating important processing elements and connections (joints) in the architecture. The next parameter is the mean time of correction; with better architecture, the average failure time rises. Also, the addition of processing elements affects the architecture when a crisis occurs (Pourali, 2017).

3. Unified Modeling Language (UML)

The UML is a standard general-purpose modeling language in software engineering developed by the Object Management Group. Using UML, almost any application that may be run on any combination of hardware, operating system, programming language, and network, can be modeled (http://www.omg.org/members/spec/UML/2.5/). UML is a third-generation modeling language and an open method for describing features, graphical display, building and documenting of the components of a developing software system. UML is used to understand, design, browse, configure, store, and control the information of software systems. This language can be used for all software development methods, all stages of the software life cycle, all applications areas and any media.

3.1. Sequence diagram

It shows how objects communicate with each other in the form of sequential messages. The sequence diagram of messages is a language that models the interaction between components with processes and the interaction between samples and the environment. In this scenario-based model, the relationship between the samples, i.e., sending and receiving messages and local events as well as the arrangement between them are described. This language, while defining the partial behavior of the system, imposes some limitations on the data values transmitted and the time of events. On the other hand, the message sequence diagram has a graphic display. The lifetime of each sample is shown in the vertical dimension of this graph and the message appears as a horizontal arrow between the transmitter and the receiver (Allen et al., 1997).

4. Petri networks

Petri Networks were introduced by Carl Adam Petri in 1962 and is used as a formal visual approach for describing and modeling dynamic, controlling, distributing and concurrent systems and communication systems. Petri networks and its extensions have been widely used for modeling, describing and architecture of systems (Bobbio et al., 2006; Fukuzawa & Saeki, 2002; Dehkordi et al., 2013) and systems safety analysis (Ramos et al., 2009; Tian & Yan, 2013). Also, using Petri nets based on graph theory, the systems with asynchronous, indeterminate, and random structures can be analyzed. It is possible to analyze The Petri network is based on the graph, and one can informally argue that a two-part oriented graph is composed of two elements of location and transition. A formal description of Petri network is as follows.

Definition 1: Network 1 is a triple of N = (P, T, F) in which:

P and T are respectively separated sets of places and transitions.

$F \subset (P \times T) \cup (T \times P)$ is a set of edges.

In Figure 1, a network example can be seen, in which:

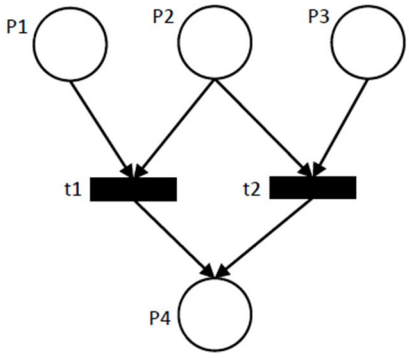


Figure 1. A network

P = {P1, P2, P3, P4}, T = {t1, t2}

F = {(P1, t1), (P2, t1), (P2, t2), (P3, t3), (t1, P4), (t2, P4)}

Definition 2: A petri network is a network in the form of PN = (N, M, W), where:

N = (P, T, F) is a network.

M: $P \rightarrow N$, which is known as the Petri net marking and shows the number of tokens in each location. M0 represents the initial marking.

W: $F \rightarrow N$l is a function that shows the weight of each edge.

Figure 2 shows a Petri net sample, in which m0 = (1, 1, 0, 0).

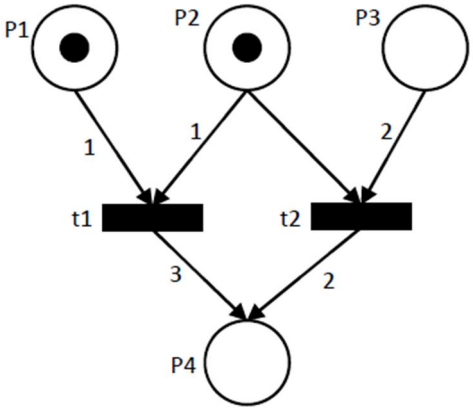


Figure 2. A Petri net

5. Related works

In 2005, Martinello presented a model for calculating **a**vailability, which aim is to evaluate the availability of Web services users (Martinello, 2005). This architecture focuses solely on software layers. Although it has used a three-layer hierarchical failure in its availability calculation model, but, all of these layers are designed from a software dimension perspective and do not pay attention to the availability requirements in lower layers.

In 2017, Sadeghi et al. provided a multi-tiered computational model for the availability of technology services. In this research, adhering to an architectural-based viewpoint, it was tried to connect all triple layers of an IT service, including the business, software and infrastructure layers in an integrated interconnected framework. Then, the general availability of a process in the business layer was accurately calculated according to the relationships modeled between all three layers and provided to the user (Valavi et al., 2017).

In (Dantas et al., 2015), the researchers provided a heterogeneous hierarchical model for calculating the availability of cloud-based services. The aim of this study was to provide certain indicators to the audience to use them for making decision on investment in creating a specific cloud platform by using the eucalyptus product compared to rent a public cloud service such as cloud services of Amazon Company.

Macedo et al. in 2014 (Macedo et al., 2014) and Zhu in 2012 (Zhu, 2012) provided a model for calculating the availability of large network systems based on the Markov model. They utilized hierarchical models for this purpose. First, they obtained the reliability of the general structure of the network with models such as the tree of failure or block diagrams. Then, they designed an appropriate Markov model on various states of the network according to the structure and calculated the availability.

In 2014, Menon et al. evaluated the failure (breakdown) methods of electronic products to estimate the time of failure. Johanna et al. (Menon et al., 2014) calculated the reliability of wireless sensor network hardware in 2011 (Johanna et al., 2011). They provided two possible standard methods based on the reliability prediction and speed testing of the wireless sensor networks wireless hardware.

In 2011, Milanovic et al. provided a toolbox for automatic generation of availability model to calculate availability based on their previous research (Milanovic & Milic, 2011). These tools are only used to calculate the availability of the network communication layer and do not consider all real-world conditions such as the software that should be available to run the process or service and all internal and external resources or addition of units that can exist in each component. On the other hand, in intricate networks such as the Internet, a single model like the reliability block diagram or the tree of failure, etc. cannot be used alone, and the hierarchical models and a combination of several models should be used to reduce the computational complexities.

In 2013, Dittrch et al. provided a framework for automatically creating reliability block diagrams from the users' perspectives in the service-oriented architectures (Dittrich & Rezende, 2013). They initially present a description of the desired service, a network topologic model and a pair of client and the service provider in web-based service software. Accordingly, they identify the relevant ATC components and prepare the service availability model from the user's perspective. Then, based on the provided availability model, they calculate the sustainable mode availability from different perspectives. Their model is not a layered model and merely computes the availability at the network level.

In 2013, Franke et al. presented an architectural framework to analyze the availability of the organizational IT service (Franke et al., 2014). Their architectural framework is used for quantitative and qualitative modeling and assessment of services availability. In their model, they have focused on census of inaccessibility factors for a service, which is practically difficult and depends on the expertise and experience of the calculator. Thus, exact and similar answers cannot be achieved under identical conditions. In 2013, Macedo et al. (Macedo et al., 2013) and Silva et al. (Silva et al., 2013) provided a tool to investigate the availability of objects Internet equipment based on the tree of failure. In (Hoque et al., 2015), the availability and reliability of satellite communications have been calculated based on the Erlang channel capacity. They used the PRISM formal description language to implement and simulate their proposed method. In contrast, in (Gonzalez et al., 2015), a random reliability model has been provided to show the most important core network availability indicators in a mobile telecom infrastructure to the telecom operators. The core network is referred to the central network and the main infrastructure of the general management of a telecom operator. In (Pourali, 2017), the availability assessment at software architecture on the object-oriented style is provided by using the colored Petri networks. Also, in (Pourali et al., 2014), a reliability assessment of a payment model in the mobile e-commerce is presented by using the colored Petri networks.

6. Availability assessment

Case Study: In Figure (1), a secured payment model for mobile e-commerce is presented (Pourali, 2014), in which important metrics such as data integrity, denial negation, identity-recognition and confidentiality are met. Now, we want to provide an accurate assessment of the availability and response time of this system using the UML language and the Petri network and then by using the cpntools software. The steps are as follows.

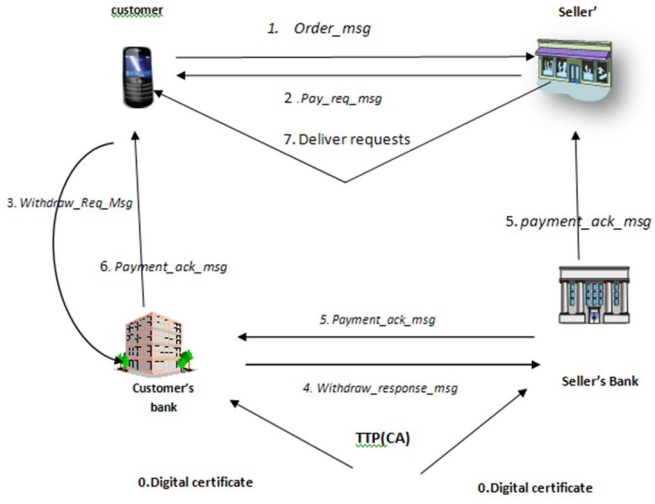


Figure 3. The proposed model for assessing availability

Step 1: Obtaining the sequence diagram from Figure 1

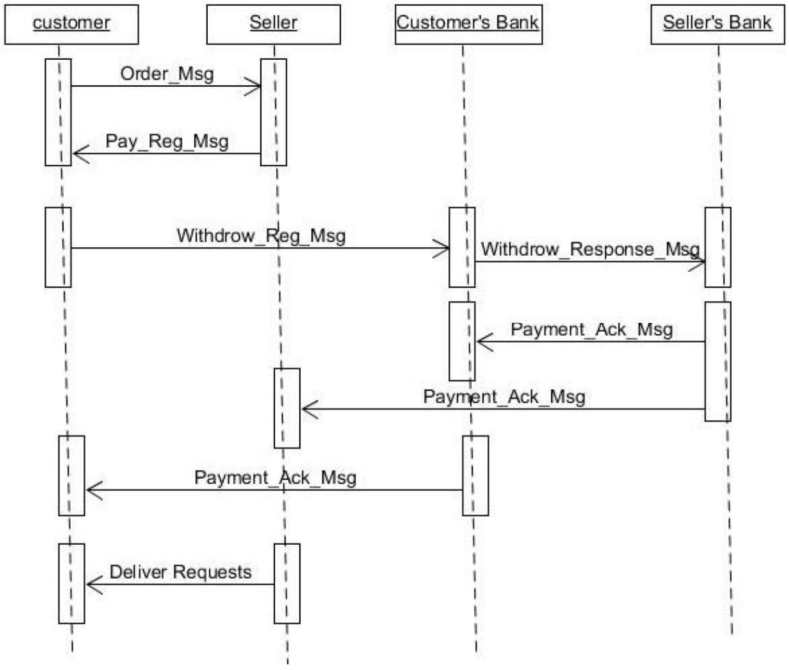


Figure 4. The sequence diagram obtained from the architecture (Figure 1)

This diagram shows the relationships between components or processes and the interaction between samples and the environment. This chart is scenario-based. It describes sending and receiving of messages and the order between them. Here, we used this diagram to describe the architectural behavior to use the algorithm of converting the UML diagrams to the colored Petri in the next step.

Step 2: Achieving the sequence diagram from the proposed architecture along with the stereotyping

Given that the UML diagrams alone are not functional, to achieve this, we need to have an executable model of these charts. Here, using the stereotypes, the evaluated metrics were labeled to the sequence diagram showing the behavior of the architecture. Then, by presenting the algorithm to convert the sequence diagram to the colored Petri net, we drawn a formal applicable model to have an accurate assessment in the CPN tools. The clichés listed below for availability and the response time labeled were used. To functionalize this, we needed to label the required stereotypes to the sequence diagram for calculating the above metrics. The stereotypes used in this method are as follows.

6.1. Sequence diagram and annotation of accessibility features and response time

The stereotypes of << PAclosedload >> and << PAopenload >> are used to specify the work load. The << PAdemand >> label (tag) is used to calculate the request time to execute a specific task, which is in the stereotype << PAstep >>. The cliche << REcomponent >> shows the basic class and the labels of this stereotype. The REcompfailprob tag shows the probability of failure of each component alone, and REbp indicates the number of recalling a component or the so-called busy period. In the nesting mode of components, the REcompfailprob tag of the external components is a function of REcompfailprob of more internal components.

Table1. The stereotype << PAclosedload >> and its tags

Stereotype	Base Class	Tags
<< PAclosedload >>	Message Stimulus Action State Subactivity State Action Action Execution Operation Method Reception	PAresepTime PApriority PApopulation PAextDelay

Tag	Type	Multiplicity	Domain Attribute Name
PAresepTime	**PAperValue**	**[0..*]**	**Workload:: resepTime**
PApriority	**Integer**	**[0..1]**	**Workload:: priority**
PApopulation	**Integer**	**[0..1]**	**Close Workload:: population**

PAextDelay	**PAperValue**	**[0..1]**	**Close Workload::extenalDelay**

Table 2. The stereotype << REcomponent >> and its tags

Stereotype	Base Class	Tags
<<REcomponent>>	Classifier ClassifierRole Component Instance	REcompfailprob REbp

Tag	Type	Multiplicity
Recompfailprob	Real (0.1)	[0..1]
Rebp	Integer	[0..*]

The stereotype << REconnector >> shows the basic class and the labels of this stereotype. The << REconnfaiprob >> tag indicates the failure probability of a connector and REnummsg shows the number of calls for an interface, or in other words, the number of messages.

Table 3. Stereotype << REconnector >> and its tags

stereotype	Base class	tags
<<RE connector>>	Message Stimulus AssociationRole	RE connfailprob REnummsg PAmodifiTime

tag	Type	Multiplicity
REconnfailprob	Real(0.1)	[0..*]
REnummg	Integer	[0..*]

Stereotype << REservice >> shows the basic class and the labels of this stereotype. The REprob label indicates the possible need for a specific service. For each REservice, the REprob tag is obtained from the total of multiplying REaccessprobs by Reserviceprobs.

Table 4. Stereotype << REservice >> and its tags

Stereotype	Base Class	Tags
<<REservice>>	Classifier	REprob

Tag	Type	Multiplicity
REprob	Real (0.1)	[0..1]

Stereotype << REhost >>: Table 5 shows the basic classes and labels of this stereotype. The REindexhost tag displays the names of hosts that are physically interconnected. In this tag, the REhosttype type represents the host name that is of string type.

Table 5. Stereotype << REhost >> and its tags

Stereotype	Base Class	Tags
<<REhost>>	Node Classifier Classifierrole	REindexHst

tag	type	muliplicity
REindexHost	REhosttype	[0..*]

In the following, the sequence diagram along with the labeled stereotypes is shown to illustrate the architecture of the proposed method (Fig. 5).

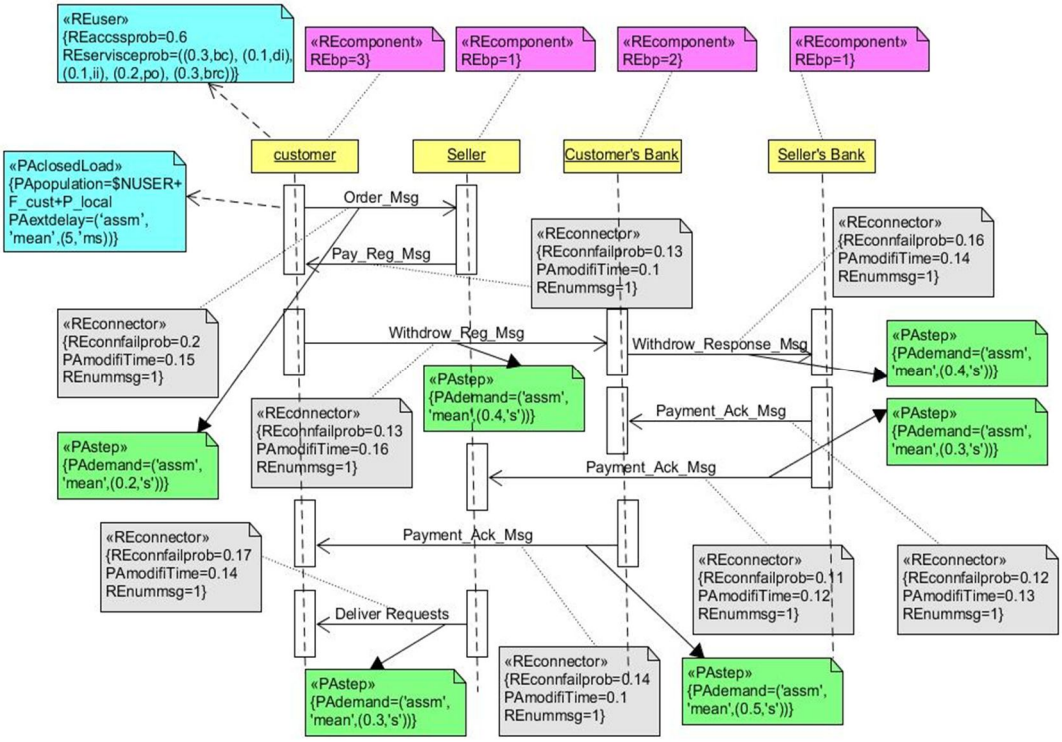


Figure 5. Sequence diagram resulted from the proposed architecture along with stereotyping

At this stage, we could add the required labels to evaluate the studied metrics to the desired diagram. Then, we had to run the Petri net conversion algorithm.

6.2. Algorithm for converting UML sequence diagrams into the colored Petri network

The goal of this step is to convert the message sender and receiver components to the Petri network. The sequence diagram in assessing the qualitative characteristics indicates that how a customer of a particular type moves through the service centers. For each set of scenarios attributed to a customer, one can choose a color for the bead in the Petri net. In the sequel, each of the message sender and receiver components and the messages between them are converted to the Petri net.

6.3. Converting asynchronous and instantaneous messages into the colored Petri network

In this mode, each of the recipient and sender elements of the message are converted to the location-transition-location. The connection between these two components takes place through a shared location. This location has the role of the input location for the sender component of the message. This common place can be considered as a waiting queue as well (Figure 6).

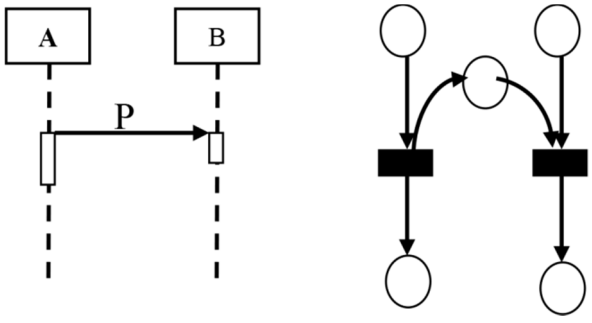


Figure 6. Asymmetric and instantaneous message and the equivalent Petri network

6.4. Converting the synchronous and instant messages to Petri network

In this case, both the sending and receiving components of the message are converted to location-transition-location. The middle position in the message recipient component is the waiting area and in the recipient component is the actx action provider (Figure 7).

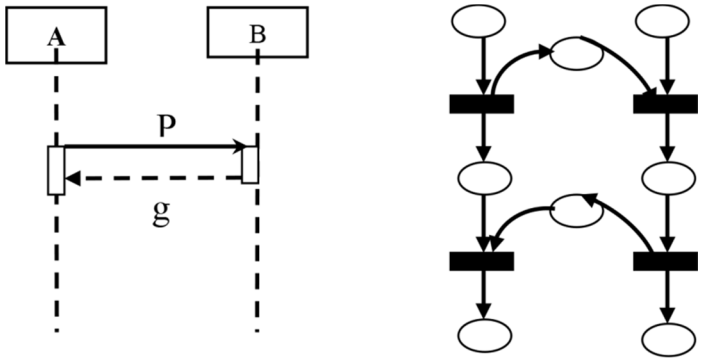


Figure 7. Synchronous and instant message and its equivalent Petri net

6.5. Selection Structure

This structure is expressed in the sequence diagram with two "alt" and "opt" operators. The "alt" operator shows the selection of an operand from a set of operands. The operation must also have an implicit or explicit protective statement to be evaluated at this point of interaction to choose the correct value (Figure 7).

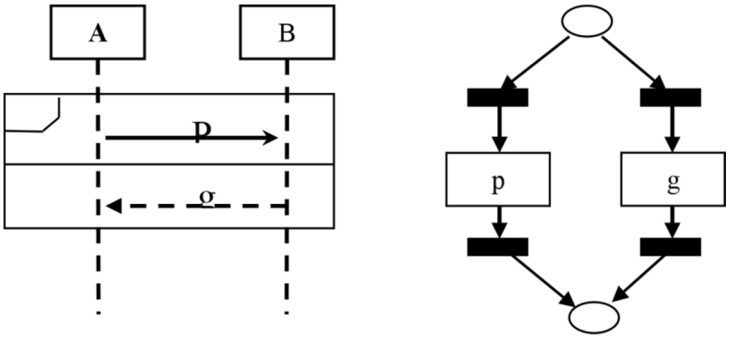


Figure 7. Selection structure and the equivalent colored Petri network

6.6. Repeat structure

This repeat structure shows one or more messages and operations in the sequence diagram. The repetition frequency or the condition for repeating the loop are determined by the operator's protectors and can include the low and high levels of the frequency of repetitions (Figure 8).

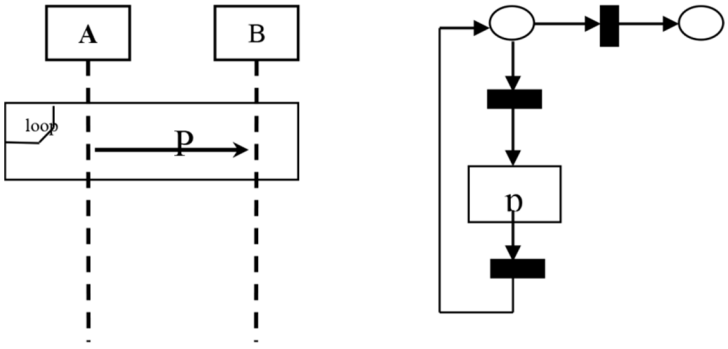


Figure 8. Repetition structure and the equivalent colored Petri network

Step 3: Petri network equivalent to the sequence diagram resulted from Figure 4 in accordance with the above algorithm (Fig. 9)

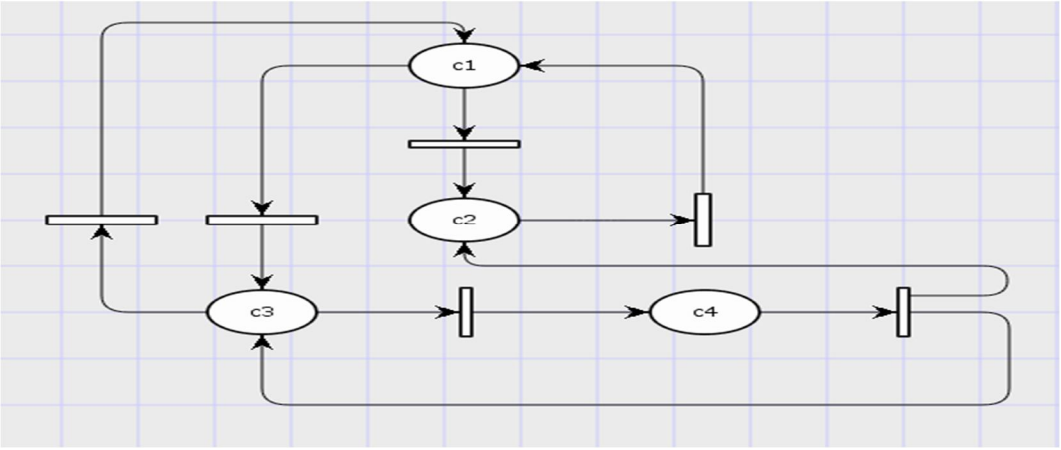


Figure 9. Petri net resulting from the sequence diagram (Figure 2)

Step 4: Calculating the Availability

Availability is calculated in a part of the system that is running and shows a proportion of the time in which the system is running. The availability is a fraction of the time that the system is alive and run. That is, how much the system is able to fix its problems and return to its normal mode? The probability level of correct operation of the system at time "t", when "t" tends to infinity determines the size of this attribute; but this attribute in a constant state of the system is a portion of the time that the system is running correctly. The availability rate can be evaluated through metrics such as MTTF[1], MTBF[2] and MTTR[3]. The availability of a system can be calculated using the MTBF and MTTR criteria, which can be calculated using equation (2).

$$\text{Availability} = \text{MTBF} / (\text{MTBF} + \text{MTTR}) \quad (2)$$

In the formula above, MTBF refers to the mean time between two successive failures and the MTTR also refers to the time required to replace or repair the failure component of the system.

Step 5: **Modeling with the help of cpntools**

At this stage, each token consists of four parts. It should be noted that the number of users can be defined dynamically.

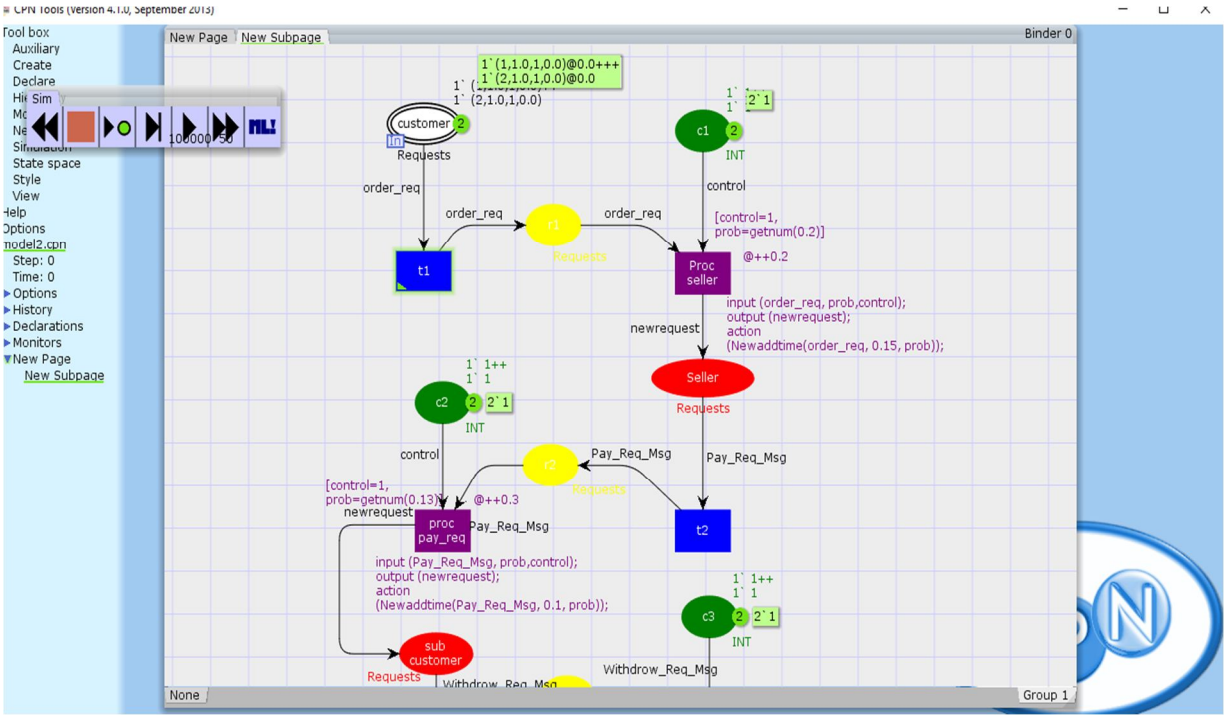
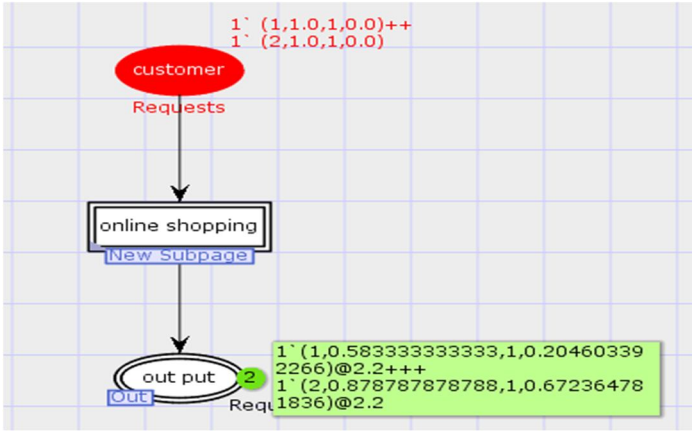


Figure 10. The colored Petri net resulting from the proposed architecture with the help of cpntools

The first parameter specifies the user's number, while the second one identifies the system's availability and the third is the default probability, which is always 1 (this parameter is used for the programming of functions). The fourth specifies the response time. Therefore, using the resulting Petri Net and the

[1] . time to failure
[2]. time between failures
[3] . time to repair

applicable model in CPN TOOLS software, we could calculate the availability and the response time. Considering the simulation of the model in CPN Tools software, the assessment of system requirements and features would be possible.



The data from the proposed architecture implementation for 5 users using the cpntools software is presented in Table 6.

Table 6. Calculating the availability rate and response time for 5 users

Running results for two users		
User Number	Availability	Response Time
1	0.583333333	0.204603392
2	0.878787879	0.672364782
Running results for five users		
1	0.986111111	0.214344239
2	0.789916363	0.461041197
3	0.838940256	0.507236835
4	1	0.493825345
5	1	0.218189585

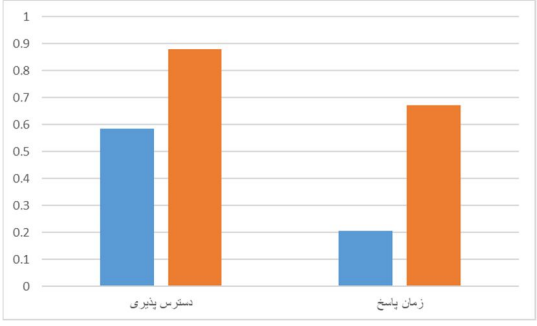


Figure 11. Results from Table 6 for 2 users

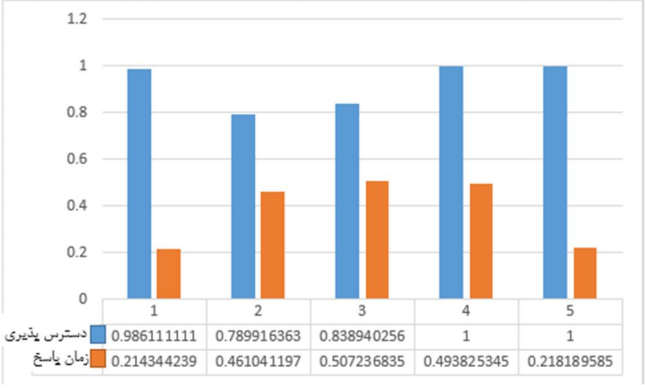


Figure 12. Results from Table 6 for 5 users

Table 7. The total mean of the implementation of the proposed method

Total mean		
Number of users	Availability	Response time
5	0.922993546	0.37893

Due to simulation of the model in the CPN Tools software, the system requirements and features can be evaluated. Based on the results of the evaluation (Table 7), the studies system has an availability of 0.922993546, which means the system cannot be used at the time of need by the probability of 0.077 (i.e., it cannot operate). As a result, the studied architecture has a high availability.

7. Conclusion

The aim of this paper was to assess the availability at the of level software architecture regardless of the features of the hardware context. Then, it was used on an applicable model based on colored Petri networks. These networks have a strong mathematical backing. In this paper, the sequence diagram was used to describe the architecture and the studied metric parameters were annotated. Finally, using the proposed method, an executable model was obtained using the colored petri networks. In fact, what achieved in this article is how to create an executable model that can examine the availability at the software architecture level. Using the results of the simulation of this model and its analysis, we succeeded to firstly identify problems in the planning phase and improve our products to avoid high economic and time costs during the implementation. Secondly, the results from the availability assessment well illustrate that how useful the system is at the time of need and how much it cannot provide service. The above system has a high availability.

REFERENCES

Allen, R., Douence, R., & Garlan, D. (1997). Specifying dynamism in software architectures. Journal of Systems Engineering, 6(4), 52-94.

Avgeriou, P., Guelfi, N., & Medvidovic, N. (2004, October). Software architecture description and UML. In *International Conference on the Unified Modeling Language* (pp. 23-32). Springer, Berlin, Heidelberg.

Avizienis, A., Laprie, J. C., & Randell, B. (2001). *Fundamental concepts of dependability* (pp. 7-12). University of Newcastle upon Tyne, Computing Science.

Bass, L., Clements, P., & Kazman, R. (2010). Software Architecture in Practice: Addison- Wesely Professional. 2nd edition, 52, 291-432.

Bobbio, A., Gribaudo, M., & Horvath, A. (2006, September). Modelling a car safety controller in road tunnels using hybrid Petri nets. In *2006 IEEE Intelligent Transportation Systems Conference* (pp. 1436-1441). IEEE.

BS EN 60300-2 (2004). Dependability management-Part 2: Guidelines for dependability management. British standard

BS EN 60300-4 (2008). Dependability management-part 3-: Application guide-Guide to the specification of dependability requirements. British standard.

Clements, P., kazman, R., & Klein, K. (2002). Evaluating Software Architecture Methods and Case Studies, AddisonWesely.

Dantas, J., Matos, R., Araujo, J., & Maciel, P. (2015). Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. *Computing*, *97*(11), 1121-1140.

David, R., & Alla, H. (2010). *Discrete, continuous, and hybrid Petri nets* (Vol. 1). Berlin: Springer.

Dehkordi, Z., Harounabadi, A., & Parsa, S. (2013). Evaluation of software architecture using fuzzy color Petri net. *Management Science Letters*, *3*(2), 555-562.

Dittrich, A., & Rezende, R. (2013, May). Model-Driven Evaluation of User-Perceived Service Availability. In *European Workshop on Dependable Computing* (pp. 39-53). Springer, Berlin, Heidelberg.

Elena Dubrova, "Fault Tolerant Design: An Introduction", Royal Institute of Technology, 2005.

Franke, U., Johnson, P., & König, J. (2014). An architecture framework for enterprise IT service availability analysis. *Software & Systems Modeling*, *13*(4), 1417-1445.

Fukuzawa, K., & Saeki, M. (2002, July). Evaluating software architectures by coloured petri nets. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering* (pp. 263-270).

Gonzalez, A., Gronsund, P., Mahmood, K., Helvik, B., Heegaard, P., & Nencioni, G. (2015, December). Service availability in the NFV virtualized evolved packet core. In *2015 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE.

Gorton, I. (2006). *Essential software architecture*. Springer Science & Business Media.

Hoque, K. A., Mohamed, O. A., & Savaria, Y. (2015, March). Towards an accurate reliability, availability and maintainability analysis approach for satellite systems based on probabilistic model checking. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1635-1640). IEEE.

http://www.omg.org/members/spec/UML/2.5/

Johanna, V., Liquan, C., Yao, Z., & Yuewei, M. (2011). Challenges in Qualitative Accelerated Testing of WSN Hardware. *Engineering*, *2011*.

Macedo, D., Guedes, L. A., & Silva, I. (2014, April). A dependability evaluation for Internet of Things incorporating redundancy aspects. In *Proceedings of the 11th IEEE international conference on networking, sensing and control* (pp. 417-422). IEEE.

Macedo, D., Silva, I., Guedes, L. A., Portugal, P., & Vasques, F. (2013, September). A framework for dependability evaluation of industrial processes. In *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)* (pp. 1-4). IEEE.

Martinello, M. (2005). *Availability Modeling and Evaluation of Web-based Services-A pragmatic approach* (Doctoral dissertation). Institut National Polytechnique de Toulouse.

Menon, S., Georgea, E., Osterman, M., & Pecht, M. (2014). Physics of failure based reliability assessment of electronic hardware. In *Engineering Asset Management 2011* (pp. 251-258). Springer, London.

Milanovic, N., & Milic, B. (2011). Automatic generation of service availability models. *IEEE Transactions on Services Computing*, *4*(1), 56-69.

Pourali, A. (2014, April). The presentation of an ideal safe SMS based model in mobile electronic commerce using encryption hybrid algorithms AES and ECC. In *8th International conference on e-commerce in developing countries: with focus on e-trust* (pp. 1-10). IEEE.

Pourali, A. (2017). Availability evaluation of Software architecture of object oriented Style using coloured Petri nets. Journal of advances in computer engineering and technology, 3(1), 1-10.

Pourali, A., Malakooti, M. V., & Yektaie, M. H. (2014). Reliability evaluation of a payment model in mobile e-commerce using colored Petri net. *Journal of Advanced Computer Science & Technology*, *3*(2), 221.

Ramos, G., Sanchez, J. L., Torres, A., & Rios, M. A. (2009). Power systems security evaluation using petri nets. *IEEE Transactions on Power Delivery*, *25*(1), 316-322.

Silva, I., Leandro, R., Macedo, D., & Guedes, L. A. (2013). A dependability evaluation tool for the Internet of Things. *Computers & Electrical Engineering*, *39*(7), 2005-2018.

Tanenbaum, A. S., & Van Steen, M. (2007). *Distributed systems: principles and paradigms*. Prentice-Hall.

Tian, F., & Yan, J. (2013). Study of Mobile E-Commerce Safety System Model Based on Petri Net. *Journal of Convergence Information Technology*, *8*(10), 33.

Valavi, M. R., BARARI, M., & Mohtashami, G. (2017). A Method for Multilayer Computing of IT Services Availability. Journal Of Electronical & Cyber Defence, 5(3).

Zhu, H. (2012, January). Reliability and availability analysis for large networking system. In *2012 Proceedings Annual Reliability and Maintainability Symposium* (pp. 1-6). IEEE.