Reliability Assessment Model of Network Operating System Based on Convolution Neural Network

Lect. Dr. Jamal N. Hasoon[1], Researcher Sarah S. Qasim[2], Lect. Dr. Mohammed A. Alshomali[3], Assist. Prof. Dr. Jane J. Stephan[4]

1,3 Department of Computer Science, College of Science, Mustansiriyah University, Baghdad, Iraq

4,2 Department of information technology, College of Science, Al- Esraa university, Baghdad, Iraq

1 jamal.hasoon@uomustansiriyah.edu.iq 2 sarah@esraa.edu.iq

3 aziznotaziz@uomustansiriyah.edu.iq 4 jane@esraa.edu.iq

نموذج تقييم موثوقية نظام تشغيل الشبكة على أساس الشبكة العصبية التلافيفية (CNN)

مدرس دكتورجمال ناصر حسون[1]، الباحثة سارة سلمان قاسم[2]، مدرس دكتورمحمد الشوملي[3]، استاذ مساعد دكتورجين جليل استطيفان[4]

1،3 قسم علوم الحاسوب، كلية العلوم، الجامعة المستنصرية، بغداد، العراق

4، 2 قسم تكنولوجيا المعلومات، كلية العلوم، جامعة الإسراء، بغداد، العراق

Abstract

The paper's subject is one of the points of fundamental significance in improving the development of complex computer systems is the effect of errors since their probability of occurrence in a widely distributed system is exceptionally high. Thus, the computer system must be dependable and provide comprehensive support for fault tolerance, fault detection, signal recovery mechanisms, failure prediction, etc. The goal is to develop a way to evaluate the reliability of the Red Hat Linux operating system utilizing data sets of system failures obtained from system injection with error reports using the Fault Injection Benchmarking and Reliability) FIBR((for a data set of system failures). It allows users to inject various faults into a system to simulate failures and analyse their impact on system reliability. First, identify the type and frequency of failures observed in the data set. This can be done by analysing the error reports generated by the FIBR tool. The categorization of the failures is based on their severity and impact on system performance by assigning a score or rank to each failure based on its severity and impact on system performance. This can be done using statistical methods like failure distribution analysis or reliability block diagrams. Finally, use the reliability model to calculate key reliability metrics, such as mean time between failures (MTBF) and failure rate. These metrics can be used to evaluate the overall reliability of the Red Hat Linux operating system under different scenarios and conditions. Moreover, compare the results obtained from the reliability analysis with industry standards and benchmarks to determine if the Red Hat Linux operating system's reliability meets the required performance level for the intended use case. The method is a technique founded on two reliability evaluation models: defect count and failure prediction. An Operating

reliability evaluation technique based on two models: defect count and failure prediction. In the defect count model, operating system defects or errors are counted and used to estimate the operating system's reliability. This is done by analysing the number of defects found during the functional testing. This information is used to predict the number of defects that may occur in the future. The defect count model assumes that the number of defects in the operating system is proportional to its size and complexity and that reducing defects leads to increased reliability. The defect count model provides a quantitative estimate of the number of defects in the operating system. Results: During several attempts to measure and assess the reliability (for total run time: 900 sec., 240 sec., 120 sec., 60 sec., and 30 sec.), it was discovered that the greater the number of bugs injected in a single period, the greater the likelihood of failure situation, and increasing total run time with the probability increased failure states as well.

Keywords: Network Operating System (NOS); UNIX, LINUX; Convolution Neural Network (CNN), Fault Count Models.

االمستخلص

موضوع البحث هو أحد النقاط ذات الأهمية الأساسية في تحسين تطوير أنظمة الكمبيوتر المعقدة وهو تأثير الأخطاء حيث أن احتمال حدوثها في نظام موزع على نطاق واسع مرتفع بشكل استثنائي. وبالتالي، يجب أن يكون نظام الكمبيوتر موثوقًا به ويوفر دعمًا شاملاً لتحمل الأخطاء واكتشاف الأخطاء وآليات إعادة تغطية الإشارة والتنبؤ بالفشل وما إلى ذلك. والهدف هو تطوير طريقة لتقييم موثوقية نظام التشغيل Red Hat Linux باستخدام البيانات مجموعات من حالات فشل النظام التي تم الحصول عليها من حقن النظام مع تقارير الأخطاء باستخدام قياس الأداء والموثوقية لحقن الأخطاء (FIBR) (لمجموعة بيانات من حالات فشل النظام). وهو يسمح للمستخدمين بإدخال أخطاء

مختلفة في النظام لمحاكاة حالات الفشل وتحليل تأثيرها على موثوقية النظام. أولاً، تحديد نوع وتكرار حالات الفشل التي تمت ملاحظتها في مجموعة البيانات. ويمكن القيام بذلك عن طريق تحليل تقارير الأخطاء التي تم إنشاؤها بواسطة أداة FIBR. يعتمد تصنيف حالات الفشل على مدى خطورتها وتأثيرها على أداء النظام من خلال تحديد درجة أو تصنيف لكل فشل على أساس خطورته وتأثيره على أداء النظام، ويمكن القيام بذلك باستخدام الأساليب الإحصائية مثل تحليل توزيع الفشل أو الرسوم البيانية الموثوقة. وأخيرا، استخدم نموذج الموثوقية لحساب مقاييس الموثوقية الرئيسية، مثل متوسط الوقت بين حالات الفشل (MTBF) ومعدل الفشل. يمكن استخدام هذه المقاييس لتقييم الموثوقية الإجمالية لنظام التشغيل Red Hat Linux في ظل سيناريوهات وظروف مختلفة. علاوة على ذلك، قم بمقارنة النتائج التي تم الحصول عليها من تحليل الموثوقية مع معايير الصناعة ومعايير الأداء لتحديد ما إذا كانت موثوقية نظام التشغيل Red Hat Linux تلبي مستوى الأداء المطلوب لحالة الاستخدام المقصودة. الطريقة عبارة عن تقنية مبنية على نموذجين لتقييم الموثوقية: عدد العيوب والتنبؤ بالفشل. تقنية تقييم موثوقية التشغيل تعتمد على نموذجين: عدد العيوب والتنبؤ بالفشل. في نموذج حساب العيوب، يتم حساب العيوب أو الأخطاء في نظام التشغيل واستخدامها لتقدير موثوقية نظام التشغيل. ويتم ذلك عن طريق تحليل عدد العيوب التي تم العثور عليها أثناء الاختبار الوظيفي. يتم استخدام هذه المعلومات للتنبؤ بعدد العيوب التي قد تحدث في المستقبل. يفترض نموذج عدد العيوب أن عدد العيوب في نظام التشغيل يتناسب مع حجمه وتعقيده وأن تقليل العيوب يؤدي إلى زيادة الموثوقية. يوفر نموذج عدد العيوب تقديرًا كميًا لعدد العيوب في نظام التشغيل. النتائج: خلال عدة محاولات لقياس وتقييم الموثوقية (لإجمالي وقت التشغيل: 900 ثانية، 240 ثانية، 120 ثانية، 60 ثانية، و30 ثانية)، تم اكتشاف أنه كلما زاد عدد الأخطاء التي تم إدخالها في فترة واحدة، كلما زاد احتمال حدوث حالة الفشل، وزيادة إجمالي وقت التشغيل مع احتمال زيادة حالات الفشل أيضًا.

1 - Introduction

With the rise in the usage of computing software, performance evaluation has become very stringent for network operating systems. Each network has a type of program dedicated to managing its resources. This software works on special high-capacity computers called a network operating system (NOS) (Qian, *et al.*,2018). The network operating system is one of the essential components of networks (Narayan, *et al.*,2008). One of the most popular operating systems for networks is UNIX and Linux. UNIX was designed with the feature of network support included with all Unix-like operating systems, including Linux and Mac OSX. Linux kernel (Red hat OS), Red Hat Enterprise Linux is a Linux Advanced Server (Linux operating system distribution). The RHEL supports diverse workloads in real hardware, virtualized, and cloud environments (Berde, *et al*,2014)(Hu, *et al.*,2017). Linux kernel (Red hat OS) (Springer, *et al.*,2022): is an open-source operating system. Red Hat Enterprise Linux is a Linux Advanced Server (Linux operating system distribution) (Felter, *et al.*,2015). The RHEL supports diverse workloads in real hardware, virtualized, and cloud environments (Baston, *et al.*,2019) (Png, *et al*.2020).

Motivation reliability is a worthy cause of critical application dependability based on a service composition in web services that may contain errors and faults. A fault system causes inaccurate changes that the system cannot execute (Yu., *et al.*,2019). Uncorrected services may accept diverse structures called "failure modes" and are requested by failure intensity. For system reliability evaluation, fault, error, and failure are used. Failure classification and prediction can fast affect the reliability of those systems, improve performance, and reduce product cost (Nguyen., *et al.*,2019).

Reliability is the closest term to dependability (Andradite, *et al*., 2020); reliability measures rely on data collected through several systems, such as data generation and collection from failure forecasts, fault and defect classification, monitoring systems, and bug tracking systems (Raykov, *et al*.,2020).

2 - Related Work

The researchers have been working to find reliability rates using different metrics, where (Zhu,*et al*.,2018) proposed the NHPP model to measure reliability through two phases. The first is to detect faults or errors by debugging operations and then try to remove them in the second phase of the model. Reliability measures depend on the expense of the detection rate and the rate of not removing the detected errors. The Datasets used as a case study were obtained from three systems, a Real-time control system, a wireless network switching system, and a Tracking Bug system.

Dataset: Most data used in the previous studies have been obtained from legal and public resources such as the Musa dataset (Ullah,*et al*.,2012), (Bhuyan,*et al*.,2016) (presented some metrics of software reliability measures (TBF and Fault number) using some machine learning algorithms such as RNN, NB, DT, and SVM classifiers. On the other hand, (Gervasi, *et al*,.2018) used the (Iyer,*et al*,2022).dataset and presented a Feed-Forward Back-Propagation Network model to predict software reliability; they measured Failure Numbers and TBF.

In addition, (Barraza,*et al*.,2019) uses K - Mean Clustering to isolate and detect the number of errors per day and then find the failure rate within 120 days. The case study is the dataset for monitoring the real-time and client-server systems.

(Wang *et al*.2018) proposed deep learning algorithms CNN and LSTM to predict fault cases to assess a web service system; the data set of SOA services oriented SoS as web service was used to evaluate and measure reliability time series depending on prediction periods. Tamura and Yamad measure MTBF and CTBF, applying the DNN to train the fault data on bug tracking systems of OSS for the Apache HTTP server.

(Felix,*et al*.,2020) implemented some machine learning classifiers, NN, RF, KNN, and SVM, for defect detection and classification. As a case study, the Dataset of ELFF open-source software is used to find the Average of defect, acceleration, and defect density.

Based on the (Bugzilla website) Dataset of Bug tracking system for OSS, (Behera, *et al*.,2019) use some classifiers such as NB, DT, RF, SVM, PNN (Shah,*et al*.,2022), and ANN to the identification of a critical fault and measure reliability based on failure rate (Behera, *et al*.,2019).

This paper introduces an approach to evaluate the reliability of the Red-hat Linux operating system through data sets of system failures collected because of system injection with error reports using the FIBR tool (Alazawi, *et al*.,2020) for each operating system component represented by ext3 ipv4, network driver, and sci. This error data type is selected to reduce execution time and keep the system from experiencing renal failure. Furthermore, this helps measure and evaluate the reliability of the operating system.

3 - Reliability Assessment

When assessing the Dependability of computer systems, researchers use faults data and error information that occurs through the running system time to determine the changes in the system and address these faults that affect the system's behavior (Li.,*et al*.,2012)(Perepelisyn, *et al*.,2022).

The following main features characterize computers and communication systems: functionality, performance, reliability, security, and cost; availability and reliability are expansive definitions of Dependability using immediate measurements; for example, safety cannot immediately be measured, it can be assessed by estimated information to gain system confidence, but reliability can be measured through time failures. When a fault is activated, it can lead to an error (Avizienis, *et al.*,2004). Furthermore, an incorrect state generated by an error may cause another error or failure, indicating a specified system boundary behaviour (Irrera, *et al.*,2016) (Crouzet, *et al.*,2012).

4 - Failure Rate or Fault Count Models

Failure rate (λ), an expected failure value per time unity. The failure rate data is obtainable for separated components, not the entire system; different professional organizations collect and publish failure rate estimates for the most-used components (Goel ,*et al.*,1998). Failure intensity considers the number of failures per unit of time and measures the reliability of a software system operating in each environment. Initially, the failure rate decreases the cause of recurrent failures in components weakness with manufacturing errors during software testing, and the failure rate can be measured (Stringfellow, *et al*,2002).

Through the system's profitable life phase, the failure rate is supposed to equal fixed value λ. In this context, the system reliability varies exponentially to exponential failure law (Trivedi, *et al.*,2017).

5 - Meantime to Failure

The expected time of the first system failure is called mean time to failure (MTTF), defined as when system reliability is. As regards exponential failure law (Andersson, *et al*.,2020).

6 - Software Reliability Assessment Models

Several steps are implemented for monitoring the previously classified system status based on error reports (bug reports). First, classifying system states is a monitoring step, as data sets for each error, including the system status due to that error, were collected. The system's reliability is evaluated at different periods to study the effect of the number of errors with the amount of time on the system's state concerning the type of error. Figure 1 shows the proposed system's general block diagram for the Reliability assessment model.

The monitored data are related to the failures state observed in the system; the analysis studies and monitors system behavior during the workload and creates a new data set that includes the system state:

failure matches the complex fault that makes the system fail, critical match to the less complex fault makes a system in a critical state that must be doing some solution,

every day is a match to a simple fault that not effected on system execution, unknown state when an undefined error is a fund in the system.

Monitoring the client depends on the CNN (Alzubaidi, *et al.*,2021) classifier built and trained in advance (Yaloveha,*et al.*,2022). It is mainly responsible for reading, collecting, and analysing the injected bug data and then forecasting the system's state. The details of the reliability model are

shown in the failure count model algorithm, where the structure of models depends on the fault rate of a run period. The failure rate can either increase or decrease based on the total time, period time, and several bugs at each Period. Reliability is based on the overall failure rate, and Period is based on the exponential failure law, as shown in Figure 2. The procedure of Reliability estimation based on the Failure count model, depending on the time intervals (Pt) for injecting bugs for the entire system (general bug dataset) by computing: the failure rate for each Period, total failures rate, based on total runtime, and the number of the run period. This is schematically shown in Figure 2.

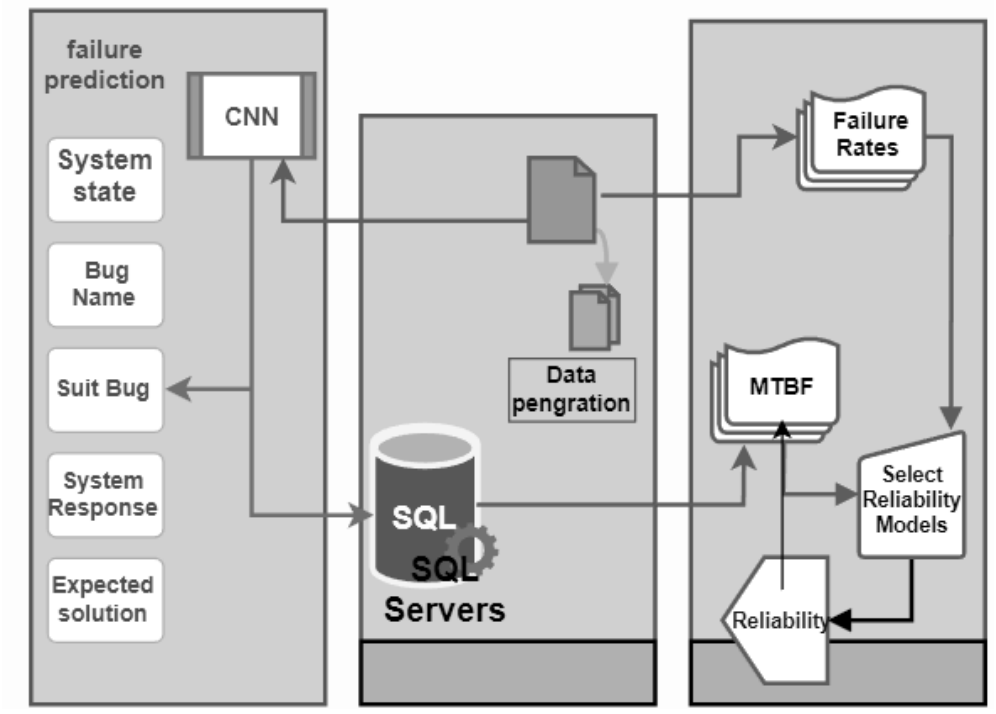

Fig. 1. Structure of proposed model of Reliability assessment

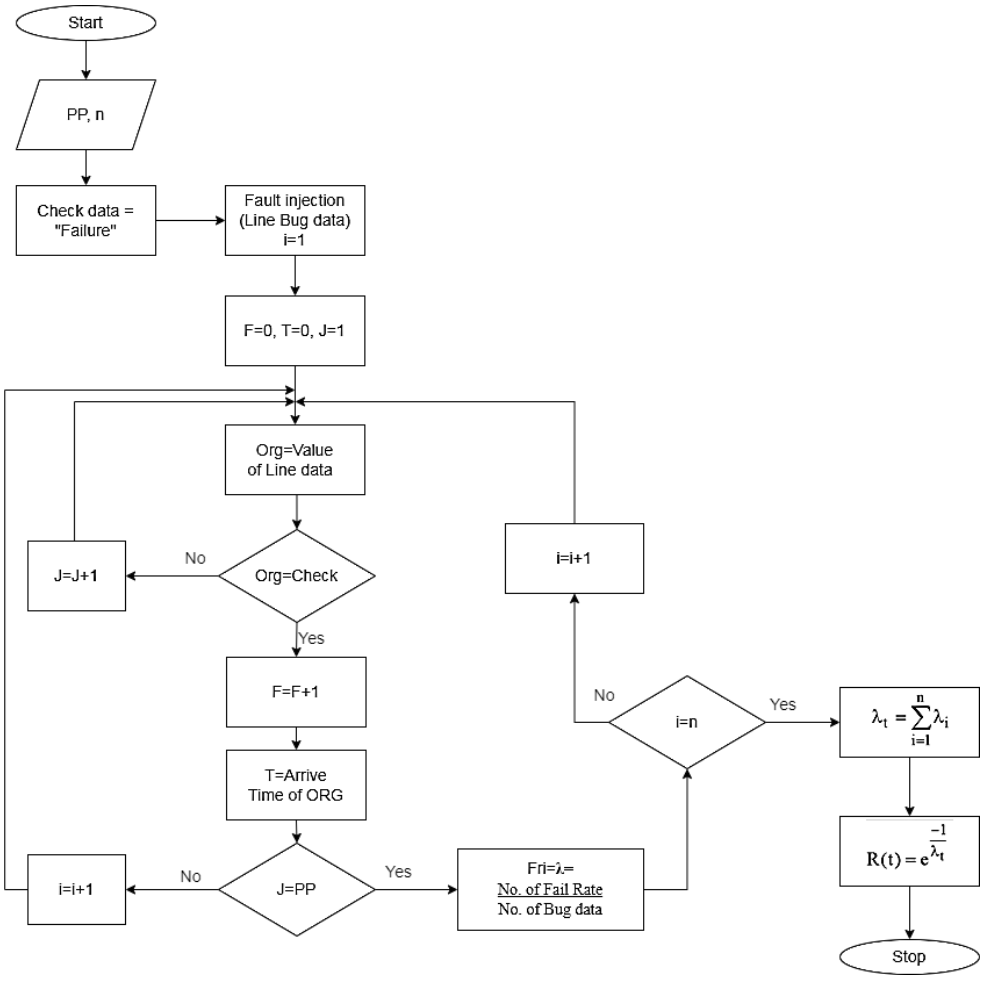

Fig. 2. Failure count model for Reliability estimation

7 - Experimental Results

Here, the dataset parts result of each of the four components in Linux mentioned in this work will be presented. In addition, the failure data collected from the error reports for each of the four software will be displayed with the system states accompanying each tyerror.

Table 1 contains all the system states for the Linux kernel, where Linux_ ext3: Total of ID Bugs = 30, BOH=10, NAM=12, ARB=5, Unk=3. Linux_ipv4: Total of ID Bugs = 38, BOH = 16, NAM=15, ARB=2, Unk=5. Linux_network driver: Total of ID Bugs=167, BOH=72, NAM=75, ARB=10, Unk=11. Linux_sci: Total of ID Bugs = 50, BOH = 25, NAM=17, ARB=5, Unk=3.

Table 1 System state Predication for Linux Kernel

Bug Type	Linux kernel System State			
	Normal	Critical	Failure	Unknown
No. of BOH Bug	123	0	0	0
No. of NAM Bug	34	55	15	0
No. of ARB Bug	0	16	1	0
No. of Unk Bug	0	0	0	22

The new dataset for Linux_Ext3-fs, Linux_ Ext3-fs has a tiny "unknown" state, while the "critical" state of the system is more if compared to cases of an "unknown" state, where the "critical" state is caused by NAM and ARB bugs with the subclass of TIM, MEM, and STO.

The data collected during the generation of the new dataset for Linux_Ipv4, Linux_ ipv4 has a tiny "Failure" state if the Bug-type is NAM with subclass is LAG, while NAM causes the "critical" state of the system too with subclass TIM. Linux_ network driver has a tiny "unknown" state, while the "critical" state of the system is more if compared to the "unknown" state, where NAM bugs cause the "critical" state with the subclass of TIM and LAG.

Linux_ SCSI drivers have a minor failure state if the Bug-type is NAM with subclass SEQ, while the ARB bug causes the "critical" state with subclass MEM and by NAM bug with subclass NAU.

To measure and evaluate the reliability of the open-source software proposed in this thesis, the system computes several necessary measures through run periods, such as the failure rate for each Period Pt and the total failure rate after the end of the total running time TT. The indicated failure rates were used to measure and evaluate reliability in two ways:

First: Based on the failure rate, counting the number of failures for all the systems and each specific Period Pt.

Second: Dependence on calculating the failure rate for each system component (4 components) for each specific Period, then calculating MTBF.

To demonstrate the reliable relationship between the number of bugs injected and the running time, the values of both bugs and time have been changed more than once. If the failure states occurred, including the data for the whole four open-source software (i.e., General Dataset), the following cases were used:

– **TT = 240 second, PP = 20 Bugs chosen randomly Pt = 10 second**

In each attempt, when the bugs are injected, the number of failures for each run period Pt is computed, and then computing the average failure FR for the total runtime; the results for the three attempts are shown in Table 2.

Table 2:Reliability for Runtime = 240

Attempt	No. of Failure	Failure Rate	MTTF	Reliability
First	11	0.045	0.0014	0.632
Second	4	0.016	0.002	0.7165
Third	10	0.041	0.0019	0.6592

– **TT = 120 second, PP = 10 Bugs chosen randomly Pt = 5 second**

In each attempt, when the bugs are injected, the number of failures for each run period Pt is computed, and then computing the average failure FR for the total runtime; the results for the three attempts are shown in Table 3.

Table 3: Reliability for Runtime = 120

Attempt	No. of Failure	Failure Rate	MTTF	Reliability
First	3	0.025	0.0042	0.882
Second	10	0.083	0.0059	0.659
Third	5	0.0416	0.0049	0.8119

– TT = 900 second, PP = 20 Bugs chosen randomly Pt = 10 second

The number of failures for each run period Pt is computed when the bugs are injected every three attempts. Then computing the average failure FR for the total runtime, the results for the three attempts are shown in Table 4.

Table 4 Reliability for Runtime = 900

Attempt	No. of Failure	Failure Rate	MTTF	Reliability
First	43	0.477	0.00239	0.620
Second	45	0.50	0.0022	0.606
Third	56	0.622	0.0022	0.536

They are studying the approach's effectiveness by implementing the fault injector at various times and durations, operating two running times, the first for all the previously general data and the second for each part of the system injecting its data independently. Reliability, as mentioned in the work, is the main attribute of dependability assessment. Therefore, some requirements closely related to reliability were measured, such as Failure rate in a specific period, MTTF for fault counting in the first reliability model, and MTBF for failure prediction in the second reliability model. Assessment

and measurement of reliability are inversely related to the number of detected failures; during several attempts to measure and assess reliability based on the data collected because of faults /errors and failure prediction, it was observed that the greater the number of bugs in a single run period, the greater the likelihood of failure situation, in addition to that by increasing the run time the probability increased failure states too.

8 - Conclusions

Through experimental results for reliability assessment, the work concluded that the CNN model works to detect the failure state of the system and compute the failure rate, MTBF, and MTTF simultaneously with the running of the target system with defects and for periods previously determined in the user interface. A confusion matrix was relied on to accurately predict the proposed CNN algorithm to give better results in terms of accuracy. Through the implementation of several trial periods to the assessment of the reliability of both the proposed models and repeated bug reports at different times to ensure the accuracy of the results, it was observed that the greater the number of bugs introduced in the Period with the increase in the total runtime, the probability of failure states is greater, and thus the software reliability value decreases. At the same time, this cannot be certain because the mechanism for selecting bugs is random. The possibility of selecting the bugs that cause the failure state may be small or not despite the presence of such bugs in the dataset; this has already been seen in some instances where the bugs made do not necessarily lead to a failure state.

Future research should be devoted to collecting the LOG files of several operating systems to build a multi objectives detection model for increasing system reliability.

9 - References

- Qian, J., Wu, H., Chen, H., Li, C. And Li, W., (2018), Fault Injection For Performance Testing Of Composite Web Services. International Journal Of Performability Engineering, Vol. 14, Iss. 6, Article No. 1314. DOI: 10.23940/Ijpe.18.06.P23.13141323.
- Narayan, S., Kolahi, S., Waiariki, R. And Reid, M., (2008), Performance Analysis Of Network Operating Systems In Local Area Networks. Proceedings Of The 2nd WSEAS International Conference On Computer Engineering And Applications, Pp. 186-188.
- Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W. And Parulkar, G. August. ONOS, (2014),: Towards An Open, Distributed SDN OS. Proceedings Of The Third Workshop On Hot Topics In Software-Defined Networking Pp. 1-6. DOI: 10.1145/2620728.2620744.
- Hu, H., Wang, Z., Cheng, G. And Wu, J. MNOS, (2017): A Mimic Network Operating System For Software-Defined Networks. IET Information Security, Vol. 11, Iss. 6, Pp. 345-355. DOI: 10.1049/Iet-Ifs.2017.0085.
- Springer, T., Linstead, E., Zhao, P. And Parlett-Pelleriti, C., (2022), Towards Qos-Based Embedded Machine Learning. Electronics, , Vol. 11, Iss. 19, Article No. 3204. DOI: 10.3390/ Electronics11193204.
- Felter, W., Ferreira, A., Rajamony, R. And Rubio, J.(2015), An Updated Performance Comparison Of Virtual Machines And Linux Containers. IEEE International Symposium On Performance Analysis Of Systems And Software (ISPASS), Philadelphia, PA, USA, , Pp. 171-172. DOI: 10.1109/ISPASS.2015.7095802.
- Baston, D. Comparison Of Real-Time Network Performance Of Redhawk™ Linux® 7.5. 2 And Red Hat® Operating Systems, (2019) Concurrent Real-Time , Pp. 1-7.
- Png, A. And Demanche, L. Oracle Autonomous Linux, (2020),. Getting Started With Oracle Cloud Free Tier. Apress, Berkeley, CA,Pp. 101-106. DOI: 10.1007/978-1-4842-6011-1_6.
- Yu, Q., Ren, J., Fu, Y., Li, Y. And Zhang, W. Cybertwin,(2019),: An Origin Of Next-Generation Network Architecture, 2019. IEEE Wireless Communications, Vol. 26, Iss. 6, Pp. 111-117. DOI: 10.1109/MWC.001.1900184.
- Nguyen, T. A., Min, D., Choi, E. And Tran, T. D.,(2019), Reliability And Availability Evaluation For Cloud Data Center Networks Using Hierarchical Models, 2019. IEEE Access, Vol. 7, Pp. 9273-9313, DOI: 10.1109/ACCESS.2019.2891282.
- Andrade, E. And Nogueira, B.,(2020), Dependability Evaluation Of A Disaster Recovery Solution For Iot Infrastructures. The Journal Of Supercomputing, 2020, Vol. 76, Iss. 3, Pp. 1828-1849. DOI: 10.1007/S11227-018-2290-0.

- Raykov, T., Marcoulides, G. A., Harrison, M. And Zhang, M.,(2020), On The Dependability Of A Popular Procedure For Studying Measurement Invariance: A Cause For Concern? Structural Equation Modeling: A Multidisciplinary Journal, Vol. 27, Iss. 4, Pp. 649-656. DOI: 10.1080/10705511.2019.1610409.
- Zhu, M. And Pham, H.,(2018), A Two-Phase Software Reliability Modelling Involves Software Fault Dependency And Imperfect Fault Removal. Computer Languages, Systems & Structures, Vol. 53, Pp. 27-42. DOI: 10.1016/J.Cl.2017.12.002.
- Ullah, N., Morisio, M. And Vetro, A., October.(2012), A Comparative Analysis Of Software Reliability Growth Models Using Defects Data Of Closed And Open Source Software. 35th Annual IEEE Software Engineering Workshop, Pp. 187-192. DOI: 10.1109/SEW.2012.26.
- Bhuyan, M. K., Mohapatra, D. P. And Sethi, S.,(2016), Software Reliability Assessment Using Neural Networks Of Computational Intelligence Based On Software Failure Data. Baltic Journal Of Modern Computing, Vol. 4, Iss. 4, Article No. 1016. DOI: 10.22364/Bjmc.2016.4.4.26.
- Gervasi, O., Murgante, B., Misra, S., Stankova, E., Torre, C. M., Rocha, A. M. A., Taniar, D., Apduhan, B. O., Tarantino, E. And Ryu, Y. Eds.,(2018), Computational Science And Its Applications–ICCSA 2018: 18th International Conference, Melbourne, VIC, Australia, July 2-5, Proceedings, Part I, Vol. 10960. Springer. DOI: 10.1007/978-3-319-95162-1.
- Iyer, R. K. And Lee, I.,(1996), Chapter 8. Measurement-Based Analysis Of Software Reliability. Handbook Of Software Reliability Engineering, Pp. 303-358. Available At: Https://Www.Cse.Cuhk.Edu.Hk/~Lyu/Book/Reliability/Pdf/Chap_8.Pdf. (Accessed 12 December 2022).
- Barraza, N. R. (2019), Software Reliability Analysis Of Multistage Projects. Amity International Conference On Artificial Intelligence (AICAI), Dubai, United Arab Emirates, Pp. 67-73. DOI: 10.1109/AICAI.2019.8701285.
- Wang, H., Yang, Z., Yu, Q., Hong, T. And Lin, X., 2018, Online Reliability Time Series Prediction Via Convolutional Neural Network And Long Short Term Memory For Service-Oriented Systems. Knowledge-Based Systems, Vol. 159, Pp. 132-147. DOI: 10.1016/J.Knosys.2018.07.006.
- Tamura, Y. And Yamada, S. September.(2017), Reliability And Maintainability Analysis And Its Tool Based On Deep Learning For Fault Big Data. 6th International Conference On Reliability, Infocom Technologies And Optimization (Trends And Future Directions) (ICRITO), 2017, Pp. 106-111. IEEE. DOI: 10.1109/ICRITO..8342407.
- Felix, E. A. And Lee, S. P.,(2020), Predicting The Number Of Defects In A New Software Version. Plos One, Vol. 15, Iss. 3, Pp. 1-30. DOI: 10.1371/Journal.Pone.0229131.

- Behera, R. K., Rath, S. K., Misra, S., Leon, M. And Adewumi,(2019), A. Machine Learning Approach For Reliability Assessment Of Open Source Software. Computational Science And Its Applications–ICCSA ICCSA 2019. Lecture Notes In Computer Science, Springer International Publishing, 2019, Vol. 11622, Pp. 472-482. DOI: 10.1007/978-3-030-24305-0_35.
- Shah Weli, Z. N. Covid-19 Prediction Model Using Data Mining Algorithms, (2022), Al-Mustansiriyah Journal Of Science, Vol. 33, Iss. 1, Pp. 45–50. Available At: Https://Mjs.Uomustansiriyah.Edu.Iq/Index.Php/MJS/Article/View/1076 (Accessed: 2 March 2023).
- Alazawi, S. A. And Al-Salam, M. N., (2020), FIBR-OSS: Fault Injection Model For Bug Reports In Open-Source Software. Indonesian Journal Of Electrical Engineering And Computer Science, Vol. 20, Iss. 1, Pp. 465-474. DOI: 10.11591/Ijeecs.V20.I1.Pp465-474.
- Li, Q. And Zhou, M., (2012), Research On Dependable Distributed Systems For Smart Grid. J. Softw., Vol. 7, Iss. 6, Pp. 1250-1257. DOI: 10.4304/Jsw.7.6.1250-1257.
- Perepelitsyn, A., Kulanov, V. And Zarizenko, I,(2022). Method Of Qos Evaluation Of FPGA As A Service. Radioelectronic And Computer Systems, No. 4, Pp. 153-160. DOI: 10.32620/Reks.2022.4.12.
- Avizienis, A., Laprie, J.C., Randell, B. And Landwehr, C., (2004),Basic Concepts And Taxonomy Of Dependable And Secure Computing. IEEE Transactions On Dependable And Secure Computing, Vol. 1, Iss. 1, Pp. 11-33. DOI: 10.1109/TDSC.2004.2.
- Irrera, I., , (2016), Fault Injection For Online Failure Prediction Assessment And Improvement (Doctoral Dissertation, Universidade De Coimbra (Portugal)). 212 P. Available At: Https://Estudogeral.Sib.Uc.Pt/Bitstream/10316/29182/3/Fault Injection For Online Failure Prediction.Pdf. (Accessed 12 December 2022).
- Crouzet, Y. And Kanoun, K. , (2012),Chaper 3 - System Dependability: Characterization And Benchmarking. Advances In Computers, Vol. 84, Pp. 93-139. DOI: 10.1016/B978-0-12-396525-7.00004-6.
- Goel, A. L., (1985),Software Reliability Models: Assumptions, Limitations, And Applicability. IEEE Transactions On Software Engineering, , Vol. SE-11, Iss. 12, Pp. 1411-1423. DOI: 10.1109/TSE.1985.232177.
- Stringfellow, C. And Andrews, A. A,(2002), An Empirical Method For Selecting Software Reliability Growth Models. Empirical Software Engineering, Vol. 7, Pp. 319-343. DOI: 10.1023/A:1020515105175.
- Trivedi, K. S. And Bobbio,(2017), A. Reliability And Availability Engineering: Modelling, Analysis, And Applications. Cambridge University Press. 726 P.

- Pradhan, S. K., Kumar, A. And Kumar, V., (2023), An Effort Allocation Model For A Three-Stage Software Reliability Growth Model. Predictive Analytics In System Reliability, Springer Series In Reliability Engineering. Springer, Cham, Pp. 263-282. DOI: 10.1007/978-3-031-05347-4_17.
- Andersson, R. And Pardillo-Baez, Y. , (2020), The Six Sigma Framework Improves The Awareness And Management Of Supply-Chain Risk. The TQM Journal, Vol. 32, Iss. 5, Pp.1021-1037. DOI: 10.1108/TQM-04-2019-0120.
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A. , (2021),. Review Of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions. Journal Of Big Data, Vol. 8, Article No. 53, Pp. 1-74. DOI: 10.1186/S40537-021-00444-8.
- Yaloveha, V., Podorozhniak, A. And Kuchuk, H.,(2022), Convolutional Neural Network Hyperparameter Optimization Applied To Land Cover Classification. Radioelectronic And Computer Systems , No. 1, Pp. 115-128. DOI: 10.32620/Reks.2022.1.09.