International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Software Quality Assurance Models and Application to Defect Prediction Techniques

Jammel Mona[1], Radhwan Hussein Abdulzhraa Al-Sagheer[2] ,Salah.M.M. Alghazali[3]

Submitted: 27/10/2022 **Revised:** 29/11/2022 **Accepted:** 26/12/2022

Abstract*:* The needs for hardware and software applications have emerged because of recent technological breakthroughs. Along with this advancement in technology, the need for software across a wide range of applications has dramatically increased. Any software sector, creating high-quality software and preserving its renown for users the most crucial undertaking for the expansion of the software industry. This must be accomplished, for software industries, software engineering is crucial. However, applying such standards to instill trust in the minds of consumers is not always straightforward. In general, the software development team perceives quality assurance in software development as an additional lengthy and extremely documentation-intensive operation that is of little value to the client. Consequently, this paper will demonstrate that the quality of the notion can be addressed from various perspectives depending on the individual's take and interest might be challenging to determine. In addition, we discussed some standards, models and applications of quality and assurance in software engineering by utilizing soft computing-based machine learning approaches that help to forecast, optimize, and efficiently learn the features, we intend to develop an effective method for predicting software defects.

Keywords: *Defect prediction techniques, Quality Assurance, Quality Models, Quality Standards, Quality Applications*

1. Introduction

The variety and complexity of software are increasing with each passing day; therefore, software quality assurance must be employed to balance quality and productivity. A unique set of software quality challenges arises in each application or business sector, and software quality must be defined following these differences. During the planning phase of a software development project, it is critical to determine the unique definition of software quality for the project at hand. Applying software parameters to a software process and a software product is a combination task that necessitates study and discipline while also providing knowledge of the status of the process and/or development of software about the goals that the software process and product must achieve. Software quality assurance (SQA) is a method of monitoring the software engineering tools and procedures used to ensure that the software is of high quality. Many other approaches can be used to do this, including establishing adherence to one or more standards, such as ISO 9000, or a model, such as the (CMMI). When developing software, software metrics should be used to measure and anticipate the quality of software products across the whole software development cycle. This will assure high-quality software while also achieving cost-effective software development and maintenance. Software is becoming increasingly crucial in today's culture. Fifty years ago, software was only utilized in specialized calculation machines; today, the software is found in virtually all consumer devices and safety-critical equipment. As a result of this change, the emphasis has shifted to achieving and maintaining software quality. Software processes play a significant role in attaining and evaluating software quality [2]. Software differs from other sorts of products in several ways. In his well-known piece "No Silver Bullet," Brooks discusses four different factors.

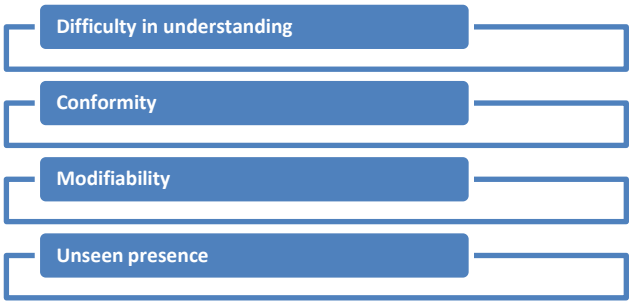

Fig. 1. Common challenges with software quality assurance

Software products may be more complex than human creations. While the computers that operate software are involved, the software itself has orders of magnitude more states than the computers. It is impossible to confirm

[1] *Faculty Medicine, University of Kufa, Najaf, Iraq*
ORCID ID : https://orcid.org/0000-0001-8189-0583
[2] *Faculty of Education for Women, University of Kufa, Najaf, Iraq*
ORCID ID : https://orcid.org/0000-0003-2999-8743
[3] *Faculty of Education for Women, University of Kufa, Najaf, Iraq*
ORCID ID :https://orcid.org/0000-0002-6205-734X
** Corresponding Author Email: radhwan.hu@uokufa.edu.iq*

software to "physical laws" since programming has no such laws. Instead, the program must unify a variety of interfaces and structures. Even while it is simple to make changes to software, the ramifications of those changes are sometimes disregarded. Geometric abstractions are a great way to think about things, but they can't be used because software is not seen. As an alternative, the software is made up of many different abstractions, each with its own set of dimensions, like control flow, data flow, and dependencies. Even though the software is distinct from other disciplines, many generic strategies can be used in software development. Quality assurance is one of these processes. As defined by "A planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements."

Since the 1980s, software processes have been emphasized to produce high-quality software development results. According to Brooks and others, the methods are required to overcome the issues associated with the unique nature of software, which they have identified. The distributional characteristics of datasets influence the selection of training samples. Few experimental studies evaluated the practical benefits of cross-project defect predictors with various programming metrics, such as process measurements, static code metrics, system metrics, etc., and how to use such metrics in a complimentary manner [18]. [19] Even though there have been multiple attempts to implement CPDP, it is still not sufficiently developed and performs poorly in actual use [20]. Furthermore, since there isn't any accurate historical data on the project, it's impossible to say with certainty how the defect predictors among WPDP and CPDP are rationally chosen. A variety of software metrics, such as the history of code change, static code metrics, network metrics, process metrics, etc., are used to create defect predictors for different forms of fault detection [21].

2. Related Works

Defect prediction models in software programming have developed into one of the important study fields since 1990, as stated by Catal and Diri (2009). Just two decades later, there were more research articles in this field overall. both ways a variety of techniques and processes were used, for example, in defect prediction models. tree decisions (Selby and Porter, 1988) brain network Naive Bayes system (Hu et al., 2007) (Menzies et al., 2004), fuzzy reasoning (Khoshgoftaar et al., 1997), case-based reasoning (Yadav and Yadav, 2015) and synthetic immunity Catal and Diri recognition framework technique (2007). Menzies et al. (2004) conducted a study drawn from using a small number of open-source NASA datasets using data mining methods. The outcomes were afterwards assessed. Using the balancing parameter, the likelihood of a false positive alarm and likelihood of being discovered. Before the During the algorithm's implementation, the authors have made The benefit of processing logs with Info-Gain filters. They continued. ensured that J48 algorithm was outperformed by Nave Bayes in terms of defect anticipated outcomes. Since the authors went on to claim that some models with low accuracy functioned brilliantly, using such models as a trustworthy There was no recommendation for a performance evaluation parameter. Yldz and Okutan (2014) used Bayesian networks to assess the defect risk and calculated the probabilistic strong linkages between software metrics. One example is the metrics used in the Promise data repository. In this study endeavor, additional metrics were defined, including LOCQ for source code quality and NOD for developer density. These measures can be collected by looking at the source code of the targeted website, Archive data with promise There was a slight possibility that the system may malfunction once the model was complete. a collection of pertinent measurements, together with a correlation between knowledge of the metrics and faults. There were additional learning algorithms based on dictionaries. more typical in the field of anticipating software flaws. Models were utilized by Jing et al. (2014) to predict software problems. focuses on the foundations of machine learning techniques. With the help of a few more components, the Comparability between several software modules can be utilized to represent a small portion of a small number of modules. Additionally, the pre-defined dictionary's coefficients incorporate the pathetically insufficient historical software data. The researchers find many dictionaries utilizing open-source programming components, including but not limited to the full dictionary, defective modules, defective modules-free modules, additional dictionaries, etc. with the help of the features of the metrics produced by the researchers. The expense of misclassification and the fact that it frequently carries greater risk than other ones that are defect-free have also been taken into account by the researchers. We offer a cost-sensitive, discriminative lexicon in this way. method for software defect learning (CDDL) classification and prediction.

3. Proposed Methodology

The training model phase and the prediction phase make up the two primary sections of the overall architecture of the model presented in this study. In the training phase, the feature data is first normalized and compressed to a particular interval, as illustrated in Fig. 2.

By enlarging the data and sampling it using the SOMTE sampling method, a new training data set is produced. Subsequently, by proportionally changing the index,

labelled and unlabeled training sets are randomly selected. The labelled and unlabeled training sets are then supplied to the Tri-training algorithm for learning. During the prediction step, the trained classifier is given the test module as input to see whether it has any problems.

3.1. Data Pre-processing

The data preprocessing it is from the most important steps in SQA and it comes as follows:

1- Profiling of data is one. Data profiling is the process of looking at, assessing, and examining data to gather statistics on its quality. An examination of the characteristics of the existing data comes first. Data scientists identify the relevant data sets for the problem at hand, make a list of their main features, and then guess which of those qualities would be useful for the suggested analytics or machine learning activity. They also consider the potential preprocessing libraries to use and link the data sources to the relevant business principles.

2- Data cleansing. Finding the most straightforward solution to quality problems, such as erasing erroneous data, filling in data gaps, or generally ensuring that the raw data is suitable for feature engineering, is the objective here.

3- Data compression. In Raw data sets frequently include duplicate information that comes from classifying events in many ways, as well as information unrelated to a particular ML, AI, or analytics application. The raw data is streamlined using principal component analysis and other data reduction techniques to make it more suitable for particular use cases.

4- Data modification. In this situation, data scientists think about how different aspects of the data should be structured to make the objective as distinct as feasible. This may require structuring unstructured data, incorporating pertinent factors when it makes sense, or deciding which critical ranges to focus on.

5- Data enhancement. In this step, data scientists use the various feature engineering libraries to apply the desired changes to the data. The outcome should be a data collection that is designed to optimize the trade-off between computation and training time for new models.

6- Verifying the information Two sets of the data have now been created. Using the first set, a deep learning or machine learning model is trained. The precision and robustness of the final model are assessed using the testing data in the second set. This second step assists in identifying any problems with the hypothesis that was used during the feature engineering and data cleaning stages. If the data scientists are satisfied with the results, the preprocessing assignment can be given to a data engineer who will figure out how to scale it for production. If not, data scientists can go back and change how the feature engineering and data purification processes were carried out.

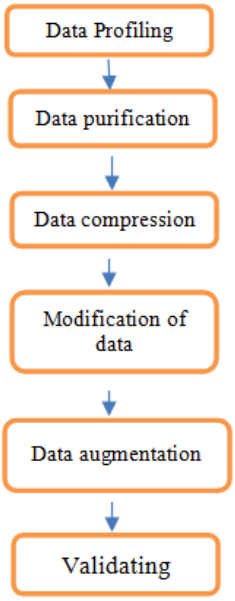

Fig. 2. Steps of preprocessing the data

Fig.2 shows the research methodology which includes two specific phases. The first phase is collecting data and the second phase is the sentiment analysis based on data mining for covid-19 health records. The research proposed methodology is shown in Fig. 3 which presents the flow chart of data collection, classifications, and applying unsupervised learning algorithm on health record text. It starts with data collection of covid-19 health records. Following the utilization of covid-19 health records in text and views. This stage is important for analyzing the results effectively. Then, the data mining algorithm is applied in the classification level with neutral entity denotes as (0), positive entity denotes as (1), and negative entity denotes as (-1). Finally, the unsupervised learning algorithm based on Word2Vec is applied for data analysis and modeling in such way it performed as opinion data mining.

3.2. Verification and validation are two critical aspects of any project.

verification and validation (V&V) are a way to make sure that software products are safe and work well at every stage of their development [6]. It is essential to make sure that the program is good at what it does, and that the product meets the needs of the people who use it (IEEE1059-93).

It is imperative to V&V that the quality of software products be looked at directly, so they use testing methods

that can find and fix problems as soon as they happen. However, it also looks at intermediate products, and in this way, it looks at the intermediate steps of the software development life cycle.

V&V is a process that checks to see if the results of a specific development or maintenance activity meet the needs of the people who use the software. It also contains to see if the final software product serves its intended purpose and meets the needs of its users. This means that a verification attempt is trying to ensure that a product is built correctly. For example, it is trying to make sure that the output products of an activity meet the specifications set by previous actions. When a product is made, validation is a way to ensure that the product is built correctly, which means that the product does what it was meant to do. Both verification and validation processes start early in making or maintaining something. They work together. It is possible to use them to get a better sense of what is important about a product with its immediate predecessor and the criteria it must meet.

Planning V&V is meant to ensure that each resource, job, and duty is clearly defined and given to the right person. The V&V plan documents that come out of this process list all of the resources that will be used and the methods and technologies that will be used. In addition, the project talks about the management, communication, policies, and procedures of the V&V activities and how they work together, as well as the rules for reporting problems and documenting the work. Understanding the different goals of each V&V movement will help plan the best ways and resources to reach them. It's important to follow standards (IEEE1012-98:s7 and IEEE1059-93) when making a V&V plan.

4. Model / Standard

4.1. McCall's Quality Model (1977)

The quality model that Jim McCall and his colleagues came up with is one of the most well-known predecessors of the quality models that we use today (also known as the General Electric's Model of 1977). In the same way as other modern models, this model came from the United States military. It was made for the US Air Force and then spread through DoD. It is mainly for systems developers and others who work on systems. McCall wants to close the gap between users and developers in his quality model by focusing on essential factors both to users and developers [7]. Product revision, product transition, and product operations are the three main ways to think about and measure the quality of a software product in the McCall quality model, shown in Figure 1. In other words, how it works. Product revision includes features like maintainability (how long it takes to find and fix a problem with the program in its operating environment), flexibility (how easy it is to make changes to the working environment), and testability (the ability to run the program in a way that can be used to test it) (the ease of testing the program, to ensure that it is error-free and meets its specification). Consider portability (the amount of work it takes to move software from one place to another), reusability (the ease with which software can be used again), and interoperability when moving products (the effort required to couple the system to another system). It is crucial to think about how a program meets its specifications, how reliable it is, how efficient it is, how safe it is, and how easy it is to use (the ease of the software).

4.2. International Organization for Standardization (ISO 9000)

ISO is short for the International Organization for Standardization4. This is a well-known group. Many standards are made by the International Organization for Standardization (ISO). The ISO 9000 series is one of the most prominent, widely used, and well-known. ISO 9001 is an international standard for quality management systems. It can be used by businesses of all sizes and types to improve their quality management systems. ISO 9001 is concerned with the procedures and methods.

The International Organization for Standardization (ISO) 9001 is a process-based way to manage quality. In contrast, ISO 9001 is concerned with the management (monitoring, ensuring, etc.) of the quality of the products and services that an organization sells or gives away [9]. There should be a lot of planning and documentation done for each of the processes below. They should also be executed, supported, monitored, and improved to some degree or another.

4.3. CMMI (Capability Maturity Model Integration)

Software Engineering Institute (SEI) at Carnegie Mellon University in Pittsburgh, Pennsylvania, came up with the Capability Maturity Model Integration (CMMI) Framework for process improvement in the United States, which the SEI made. This is an abbreviation for Capability Maturity Model Integration, and it stands for Capability Maturity Model Integration. It is a collection of best practices for improving how things work [12]. It is very organized and systematic. If you want to use the CMMI process model framework, there are three different ways to use it: CMMI for Development, CMMI for Services, and CMMi for Acquisition.Software engineering, manufacturing, financial services, aerospace, computer hardware, defense, and telecommunications are just a few of the businesses covered by these three groups of companies. People who work for businesses have found using the CMMI Model Framework implementation. This includes but isn't limited to:

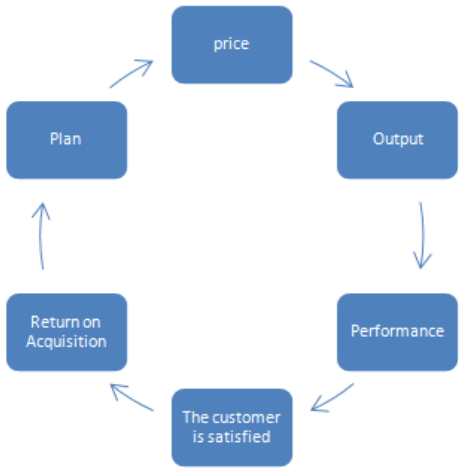

Fig. 3. CMMI Model Framework implementation

5. Application

There is no room for mistakes when it comes to today's fast-paced business world. People and businesses want things to be perfect. On the other hand, corporations must take risks with production quality because of short periods, limited budgets, and the need to adopt new technologies quickly. Businesses don't have to put themselves in harm's way to do well to make money now. It's easier and faster to get your product to market faster with Hexaware's quick and flexible Quality Assurance and Testing Services (QATS). This means you can get your product to market quicker and meet or even exceed your customers' expectations.

Quality Assurance and Testing Services helps businesses all over the world achieve business transformation by providing the following services:

• Excellence in software quality

• Test Consulting and Strategic Planned

• Engineering for Performance and Automation

• Testing Centers of Excellence (TCoEs) that have proven test processes, techniques, and governancemodels that work well for them.

5.1 AQtime:

A profiler automatically keeps track of how well it runs and how much memory it uses during the program's execution. It is an all-in-one profiling tool with a set of profilers that collect actual performance and memory allocation information at runtime and show it to the user in both summary and detailed formats. The results are displayed, most likely in a comparable way, and can be examined in more detail. To show how much time each function has spent, it might record when each function call starts and ends, then show that percentage as a percentage of the time each part has spent. It also provides customized filters for the show results, graphical views of function call hierarchies, source code views, disassemble views, etcetera., which makes it much easier to analyze the profiled application [14][27].

5.2 Code Coverage

The internet applications growth is increasing and boosting, the software testing is challenged for this application. Many code-based techniques introduced for testing web applications such as java-based testing techniques; source code based automatic test generation. In these approaches, you learn how to test transactions over a network, how to look for mistakes in complex systems, and how to make sure your job is efficient. Some people use automatic test generation from source code to ensure that the tests cover every statement in a program. PathGen and TestGen are two parts of the eXVantage method for code coverage. PathGen and TestGen are two parts of the eXVantage implementation of test generation. PathGen is a program that looks at the source code of a program and comes up with a list of the most critical paths. TestGen reads in the ways and creates test data for each of them based on its information about each of them. TestGen establishes a set of test cases that can be run. They are written in the same way as the source code of the original program [15][28].

5.3 Test

In this section we are going to show some results related to the affect of testing a proposed system in government or private sector that the organization used inside of Iraq, the survey was indicated to check the validity of the software and the ease of use of it the survey questions we made some section to make sure of the outcome from this paper

As follows first how we can make of the quality assurance of an app, and we had some questions related to that

Table 1: Sample of Questions used in different sections**.**

1	Ease of installation	Ease of use	Hardware compatiblity	Operationg system compatibility
2	Security	Ability to integrate with otther apps	Look and feel	Application software compatibility
3	Documentation	Collaborate with team	Clarity of documentation	Accessibility of product support

The scale of the answers tothese questions were defined as 5 as strongly agree then 1 as strongly disagree.

In the following figure showing the results of mean analysis for the power analysis and the value was shown as 0.5 and the data we used for our analysis shows significant impact.

Table 2: Mean Analysis

Power Analysis Table

	N	Actual Power[b]	Test Assumptions Power	Std. Dev.	Effect Size	Sig.
Test for Mean[a]	18	.516	.5	1	.500	.05

a. Two-sided test.

b. Based on noncentral t-distribution.

In the linear regression table as a result the data were showing the effect as 0.05 percent in the data processing for the correlation value of the survey as shown in the following table.

Table 3: Data Correlation

Power Analysis Table

	N	Actual Power[b]	Predictors Total	Predictors Test	Test Assumptions Power	Test Assumptions Partial[c]	Sig.
Type III F-test[a]	153	.502	100	100	.5	.5	.05

a. Intercept term is included.

b. Predictors are assumed to be random.

c. Multiple partial correlation coefficient.

And the following figure will show main relationship between the age group and the ease of using systems as will be shown in the following figure

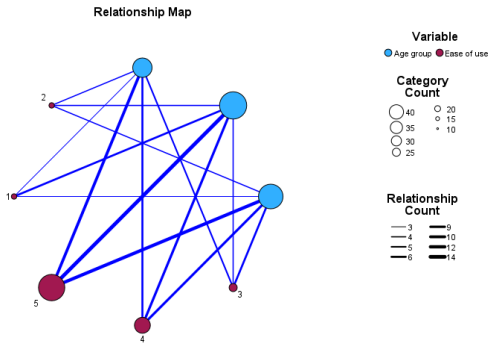

Fig. 4: Relationship between Age and Case of use.

Table 6: Variable Processing Summary

In the above figure we shown the relationship between the age group and the ease of use in each application that conducted in their firm.

Table 4: Dependent Variable

Model Description

Model Name		MOD_1
Dependent Variable	1	Ease of installation
	2	Operating system compatibility
	3	Security
	4	Hardware compatibility
	5	Clarity of documentation
Equation	1	Linear
Independent Variable		Application software compatibility
Constant		Included
Variable Whose Values Label Observations in Plots		Accessibility of product support

In the above table we show all the factors that could affect the use of applications and check the quality assurance of it so in the first mode we have five dependent variables and one independent variable and that would be compatibility of the application with different operating systems and the use of them for each single or multitasks

Variable Processing Summary

		Variables: Dependent: Ease of installation	Dependent: Operating system compatibility	Dependent: Security	Dependent: Hardware compatibility	Dependent: Clarity of documentation	Independent: Application software compatibility
Number of Positive Values		99	99	99	99	99	99
Number of Zeros		0	0	0	0	0	0
Number of Negative Values		0	0	0	0	0	0
Number of Missing Values	User-Missing	0	0	0	0	0	0
	System-Missing	0	0	0	0	0	0

per time.

Table 5: Case Processing Summary.

	N
Total Cases	99
Excluded Cases	0
Forecasted Cases	0
Newly Created Cases	0

In the above tables the test shows 99 valid data were collected and included in the analysis stage and all of them were complete and positive values that could be used and get high accuracy of the results as we will show in the following table.

Table 7: ANOVA Analysis

ANOVA					
	Sum of Squares	df	Mean Square	F	Sig.
Regression	191.636	1	191.636	.	.
Residual	.000	97	.000		
Total	191.636	98			
The independent variable is Application software compatibility.					

The Anova analysis results show 191.646 mean square were that show the affect of the experience of users and the knowledge they have will measure the quality assurance and which technique could be used to check the system outcome, since all the data we show linear increment in the user decency.

Testing software that automatically checks the classes made in Microsoft's. NET Framework without having to write a single test case or stub. Parasoft® dotTEST™ is an integrated Development Testing tool that automates a wide range of best practices that have been shown to improve the efficiency of software development teams while also improving the quality of their work. This tool is called dotTEST. Make it possible to:

• Static analysis, which includes static code analysis, data flow analysis, and measurements analysis

• Unit testing is when you write, run, optimize, and manage unit tests.

• Plugin testing: It sets up the environment for the plugins to run in and run them.

Team members will benefit from this because it gives them a realistic way to find, expose, and fix problems in their.NET code (such as C#, VB.NET, ASP.NET, and Managed C++) to work as it should. The severity of each issue can be set, and each one is automatically assigned to the developer who wrote the code that linked to it [26]. It also gives the developer a direct link to the incorrect code and explains how to fix it to their IDE.

5.4 GS Data Generators

Automated testing and data creation software can be used to make test data for software quality assurance (QA), performance and usability tests, and database load tests. The GS Data Generator tool can be used to make test data for QA, usability, performance, and database load tests. If you want to run random tests, meaningful tests, and business-intelligent tests for system integration testing, system development, and marketing, you can use this computerized testing application. It's also suitable for testing ERP, CRM, and data warehouse systems.

5.5 jenny

Jenny is a tool for logically making regression tests. Consider using jenny if exhaustive testing seems like a lot of work because of the significant number of features and interactions that need to be tested in each situation. It will cover most of the interactions with a lot fewer tests. If you use this method, you'll be able to test all the features that work together and avoid combinations of features that don't work well together.

5.6 WebART

The Online Computer Library Center (OCLC) created and uses WebART, a test automation tool, to test intranet, internet, and World Wide Web applications and content. It makes it possible for you to quickly create, run, and analyze automated tests that validate a website's or an application's functioning.

5.7 Web Performance

Web-based application quality assurance testing is a planned and systematic sequence of procedures for

internet client and server products that guarantees they conform to a predefined set of metrics. The tools' effectiveness was measured by their response times and latency, their structural quality, content, accuracy and consistency, and overall performance [16][25]. Web-based testing will become a critical component of continuous quality assurance, with web-testing tools ensuring that the testing process is repeatable and uniform.

6. Prediction Techniques

All modern defect prediction models are founded on a sophisticated combination of programming metrics, which allows a defect predictor to normally attain a high level of precision. One of the few feature selection algorithms, principal component analysis (PCA), has the potential to considerably reduce the number of data dimensions [11].

The number of defects that would be found in a project or product module can be predicted using programming defect prediction (PDP) methods, which are also used to classify which modules are most likely to include errors. Many novel methods have been developed to characterize and forecast disappearances; they can be broadly divided into methods to determine whether a particular programming antique rarity is likely to have a deformity (Classification) and methods to determine how many flaws would be present in each programming antique (Prediction).

Expert viewpoints were employed in a study by Staron and Meding [10], and their performance was contrasted with that of other information-based models. The author's earlier studies illustrate the long-term analytical capabilities of SRGMs (Software Reliability Growth Models), demonstrating its value in determining or foreseeing failure and consistency in the context of the automotive sector. To categorize the software modules that are most likely to be defective or to assess the compactness of software defect, several software modules connected to code attributes like complexity, size, etc., have been successfully employed.

Additionally, Iker Gondra [12] and Xie et al. have examined techniques that leverage code and altered measurements as information sources and make use of machine learning strategies for categorization and forecasting (2011).

5.1 Evaluation Measures for Prediction of Software Bugs

several methods for predicting software defects, such as The terms true positive (TP), true negative (TN), false positive (FP), and false negative will be covered in this section (FN). TN stands for the number of instances of clean software that was improperly identified as problematic, while TP stands for the number of clean software that was incorrectly classed as clean. The numbers FP and FN represent, respectively, the number of clean software instances that are mislabeled as faulty and the number of clean software instances that are mislabeled as defective.

Classification accuracy, commonly referred to as the right classification rate, is one of the most important and straightforward criteria to evaluate the effectiveness of predictive models. It is employed to determine the contribution of each correctly classified case to the total number of occurrences. A alternative statistic called as accuracy is produced by dividing the total number of occurrences identified as faulty (TP + FP) by the total number of instances accurately diagnosed as faulty (TP) [16]. Recall [22] additionally computes the ratio of instances that were correctly identified as defective (TP) to all instances that were defective (TP + FN). Because they represent a harmonic mean of precision and recall, F-score measures have been employed in numerous investigations [24] [23]. By balancing trade-offs between TPR and FPR, ROC-AUC determines the area under the receiver operating characteristic (ROC) curve.

The G-measure is an additional metric for forecasting software faults. It is defined as a harmonic mean of recall and specificity. The likelihood of false alarm is defined as the ratio of clean instances that were wrongly classified as defective (FP) to all clean instances (FP + TN) (PF).

using several datasets and a variety of machine learning methods The dataset was chosen for further comparisons because the majority of pertinent papers used it to evaluate the efficacy of its SDP techniques. Table 1 includes the statistics and a list of the datasets used in the investigations. We choose the base classifiers RF, DS, Linear SVC SVM, and LR. Boosting and bagging classifiers were also considered for each base classifier. The investigation was carried out in a Python environment. In this study, the performance of the classifiers was assessed using the classification accuracy, precision, recall, F-score, and ROC-AUC score. It is important to stress that these metrics were calculated using the weighted average approach. The weighted average was chosen with the intention of computing metrics for each class label and correcting for label imbalance.

The prediction of software defects is a critical task in the realm of software engineering. In the chapter before, the methods for defect prediction using machine learning software are explained. These methods also dealt with the problem of software bug mismatch, although classification accuracy and general efficacy are still a problem for researchers. We propose a hybrid feature reduction approach and an artificially dependent neural network approach for the prediction of software issues. The first subparagraph of this article presents the

improved PCA methodology for dimensional reduction and numerical modeling, while the second subparagraph gathers information on the combined usage of the neural network and the traditional PCA method.

7. Discussion

Given the degree of predictability present in the process and its multiple subprocesses, as well as the diversity of developers, users, and uses, it is unlikely that a deterministic control system will help to improve the software development process. Similar to statistical physics, only a method based on statistical modeling, such as statistical control, can work. The panel believes that the current situation and the stage Deming was at when he began to promote statistical process control in the 1950s are not all that dissimilar. The process of software engineering needs to be thoroughly understood by statisticians, and software engineers need to be aware of what statisticians can and cannot perform. If cooperative contacts and the growth of this mutual understanding can be encouraged, a significant influence likely will take place on a par with Deming's introduction of statistical process control techniques in hardware manufacturing.

Of course, this does not imply that all software issues will be resolved using statistical methods, just as not all issues with vehicle manufacture will be resolved through statistical methods. On the other hand, the software business has historically been heavily reliant on technology, and most future increases in productivity will come from innovative new ideas. For instance, most of the productivity increase

1996. National Academies of Sciences, Engineering, and Medicine Engineering Statistical Software.

8. Conclusions and further considerations

The main goal of this research is to forecast software problems using information-mining techniques. Additionally, this area has become an important area of research where a variety of techniques have been investigated to improve the effectiveness of identifying software flaws or foreseeing vulnerabilities. The purpose of this study has been to present a high-level overview of various types of quality structures without diving too deeply into any one model or philosophy. The objective was to paint a nuanced and thorough picture of the terrain of what is occasionally (and largely unthinkingly) referred to as "quality." The study established that quality is a challenging concept to grasp and can be approached from various angles depending on one's perspective and level of interest. In addition, we discussed some standards, models, applications of quality and assurance in software engineering.

9. References

[1]S. S. Yau, et al., "An integrated expert system framework for software quality assurance," in Computer Software and Applications Conference, 1990. COMPSAC 90. Proceedings., Fourteenth Annual International, 1990, pp. 161-166.

[2]P. Runeson and P. Isacsson, "Software quality assurance-concepts and misconceptions," in Euromicro Conference, 1998. Proceedings. 24th, 1998, pp. 853-859 vol.2. DOI: 10.1109/EURMIC.1998.708112

[3]L. Schrettner, et al., "Software Quality Model and Framework with Applications in Industrial Context," in Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on, 2012, pp. 453-456. DOI: 10.1109/CSMR.2012.57

[4] C. G. Manak, "Software Quality Assurance Management," in Military Communications Conference - Communications-Computers: Teamed for the 90's, 1986. MILCOM 1986. IEEE, 1986, pp. 21.2.1-21.2.2.

[5]F. B. Brown, et al., "MCNP version 5," Trans. Am. Nucl. Soc, vol. 87, p. 4, 2002.

[6]O. Balci, "Validation, verification, and testing techniques throughout the life cycle of a simulation study," in Simulation Conference Proceedings, 1994. Winter, 1994, pp. 215-220.

[7]D. Samadhiya, et al., "Quality models: Role and value in software engineering," in Software Technology and Engineering (ICSTE), 2010 2nd International Conference on, 2010, pp. V1-320-V1-324.

[8]R. Saini, et al., "Analytical study of maintainability models for quality evaluation," Indian Journal of Computer Science and Engineering, vol. 2, pp. 449-454, 2011.

[9]L. P. Dreyfus, et al., "The impact of just-in-time implementation and ISO 9000 certification on total quality management," Engineering Management, IEEE Transactions on, vol. 51, pp. 125-141, 2004.

[10]Rajbahadur, G.K., Wang, S., Kamei, Y., Hassan, A.E. 2017, May. The impact of using regression models to build defect classifiers. In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR) 135–145. IEEE..

[11]He, P., Li, B., Liu, X., Chen, J., Ma, Y. 2015. An empirical study on software defect prediction with a simplified metric set. Information and Software Technology, 59, 170–190.

[12]Kim, S., Zhang, H., Wu, R., Gong, L. 2011, May. Dealing with noise in defect prediction. In 2011 33rd International Conference on Software Engineering (ICSE). 481–490. IEEE.

[13]L. Davila-Nicanor and P. Mejia-Alvarez, "Reliability improvement of Web-based software applications," in Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference on, 2004, pp. 180-188.

[14]N. Chen, An analysis of a NIDS for hardware/software implementation: University of New Brunswick (Canada). 2006.

[15]J. J. Li and H. Yee, "Code-coverage guided prioritized test generation," in Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International, 2004, pp. 178-181 vol.2.

[16]W. Emmerich, "Unit Testing Tools."

[17]D. Barbosa, et al., "ToXgene: a template-based data generator for XML," in Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002, pp. 616-616.

[18] Li, Z., Jing, X.Y., Zhu, X., Zhang, H., Xu, B., Ying, S. 2017. On the multiple sources and privacy preservation issues for heterogeneous defect prediction. IEEE Transactions on Software Engineering.

[19] Zimmermann, T., Nagappan, N., Gall, H., Giger, E., Murphy, B. 2009, August. Cross-project defect prediction: a large-scale experiment on data vs. domain vs. process. In Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 91–100. ACM.
[20] Rahman, F., Posnett, D., Devanbu, P. 2012, November. Recalling the imprecision of cross-project defect prediction. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, 61. ACM.
[21] Radjenović, D., Heričko, M., Torkar, R., Živkovič, A. 2013. Software fault prediction metrics: A systematic literature review. Information and software technology, 55, 1397–1418.
[22] Elish, K.O. and Elish, M.O. (2008) Predicting Defect-Prone Software Modules Using Support Vector Machines. Journal of Systems and Software, 81, 649-660.
[23] Kim, S., Zhang, H., Wu, R. and Gong, L. (2011) Dealing with Noise in Defect Prediction. 2011 33rd International Conference on Software Engineering, 21-28 May 2011, Waikiki, 481-490.
[24] Lee, T., Nam, J., Han, D., Kim, S. and In, H. (2011) Micro Interaction Metrics for Defect Prediction. Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, 5-9 September 2011, Szeged, 311-321.
[25]Nahi, A., Flaih, L., & Jasim, K. (2022). . In Communication Engineering and Computer Science. Retrieved from https://conferences.cihanuniversity.edu.iq/index.php/COCOS/22/paper/view/754
[26]Al-Rabeeah, A., & Hashim, M. (2019). Social Network Privacy Models. Cihan University-Erbil Scientific Journal, 3(2), 92-101. https://doi.org/10.24086/cuesj.v3n2y2019.pp92-101
[27]A. A. N. Al-Rabeeah and F. Saeed, "Data privacy model for social media platforms," 2017 6th ICT International Student Project Conference (ICT-ISPC), 2017, pp. 1-5, doi: 10.1109/ICT-ISPC.2017.8075361.
[28]Al-Majdi, K., Salman, A., Abbas, N., Hashim, M., Taha, M., Nahi, A., Saleh, S. (2022). MLCM: An efficient image encryption technique for IoT application based on multi-layer chaotic maps. International Journal of Nonlinear Analysis and Applications, 13(2), 1591-1615. doi: 10.22075/ijnaa.2022.6571