CHETANA
International Journal of Education (CIJE)

Peer Reviewed/Refereed Journal
(ISSN: 2455-8729 (E) / 2231-3613 (P))

Impact Factor
SJIF 2023 - 7.286

Prof. A.P. Sharma
Founder Editor, CIJE
(25.12.1932 - 09.01.2019)

Experimental Research on Software Quality Assurance using Artificial Intelligence

Dr. Ajay Krishna Tiwari
Academician & Economist and Ph.D. Guide
Dr Pura Ram Meghwal, HOD, Research Director
IASE Deemed to be University, Sardarshahar, huru
Email-drajayhod@gmail.om, Mob.-9251616036

First draft received: 12.07.2023, Reviewed: 18.07.2023, Accepted: 26.07.2023, Final proof received: 30.07.2023

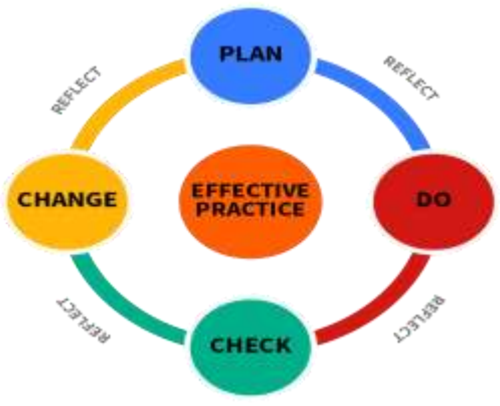
Abstract

Software systems are getting bigger and bigger as the foundation of society. and complexity and connectivity, and in times of extreme uncertainty Innovative development, continuous change, expansion and environmental adaptation Under such circumstances, it is essential to build quality, verify and Evaluation/correction/management activities are exploratory and adaptive in nature. where machine learning with generalization ability Data-driven optimization for new objects and situations genetic algorithm genetic Algorithm; From request to maintenance. The use of AI for quality activities is being researched and practiced in this research paper types of AI mainly machine learning and As an AI activity related to software quality in general overview of requirements followed by design and implementation of specific requirements, Testing debugging Other quality control and maintenance procedures Separately, we'll explain AI use efforts related to quality in a broader way.

Keywords: Artificial Intelligence, Software Quality Assurance, Innovative development, Data-driven optimization, enhancing qualities etc.

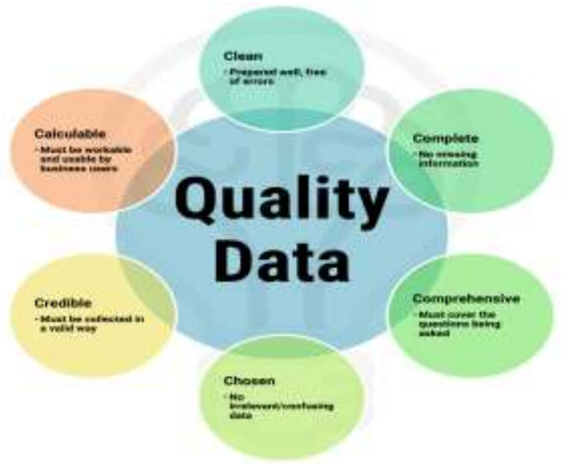
Introduction

Holistic Quality Activities using AI - from a process and objective perspective Main use of AI for pipeline quality related activities Table 1 summarizes the results of AI use. On the basis of classification. properties of the source process product or data) Attribute prediction and inference Attribute identification and extraction Transformation and generation of targets As a process that is directly related to product quality assurance, This includes testing, debugging, quality control, etc. use of AI through systematic testing for verification and validation by product quality improvement, product defect prediction, and They have a quality assessment and defect report management initiative simultaneously In the process of requirements, design, implementation and maintenance Efficient production and quality improvement through the use of AI is a group, Later including author's results It is necessary to explain the various AI usage initiatives.



Use of AI for Quality in Requirements

Complex requirements documents continue to achieve and change a lot Especially in difficult situations through machine learning-based natural language processing Efficient and adaptive processing is useful. Efforts to use AI in education holistically to achieve need analysis of requirements Organized to prioritize requests for particular requests in analysis, classification of non-functional requirements, validation summary, etc. Enhancing qualities such as consistency and clarity, Afterwards, requirements can be incorporated and verified correctly and efficiently. and contributes to the improvement of product quality. explain the quality assessment and classification efforts of.


Quality Data
Clean
• Prepared well, free of errors
Complete
• No missing information
Comprehensive
• Must cover the questions being asked
Chosen
• No irrelevant/confusing data
Credible
• Must be collected in a valid way
Calculable
• Must be workable and usable by business users

Requirements Quality Evaluation

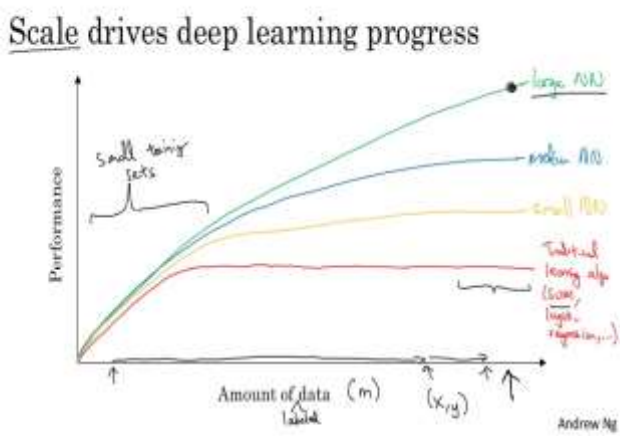
As a representative quality of the required specifications IEEE/ISO/IEC 29148 2018 lists the following quality characteristics: Individual Requirements Implementation Freedom Ambiguity Consistency Completeness Uniqueness Feasibility Traceability Verification and completeness consistency satisfaction of set of requirements Clarity, especially unambiguity, of the scope of the definition evaluable by natural language processing For example in 4, long-term short-term memory; Transfer of deep learning model based on LSTM by learning and fixing in the target domain It automatically judges grammar and ambiguity in the problem domain. focused on machine learning Software quality using AI Guarantee

Here, since the scale of the data is important in deep learning, mechanical

Back-translation (e.g. English → Spanish → English)

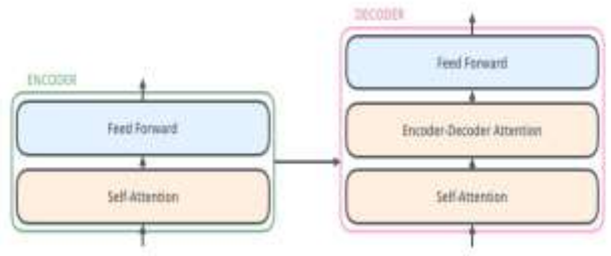
expanding training data

classification of needs by capturing the features of the requirement document using machine learning, non-machine security requirements and utility requirements automatically classified into individual quality requirements such as Can support efficient and error free processing of large volume request documents for example in [5], support is provided using lexical and syntactic features. Support Vector Machine (SVM) Automatic classification is done by. [6] in basic A comparative evaluation of various machine learning algorithms is done in brief. re BERT Bidirectional Encoder in Natural Language Processing Since Representation from Transformers 7 Based on large number of generic documents using transformer model Initial learning of linguistic expressions and fine-tuning in the target domain Analytics is active and relevant analytics are expected. Tracing that is being implemented on essential documents as waiting is possible.



Encoder/Decoder and Self-Attenuation

It is a neural machine translation model that combines B uses the transformer model labeled ERT mask some unmarked text and Part guessing task and input sentence continuity recognition task Understanding context and sentence relationships is possible through training For example, in 8, B ERT is used Classification of functional requirements documents Four Using AI for Quality in Design and Implementation complex quality requirements massive or variable state Then, among a myriad of possible design and implementation options, It is difficult to keep selecting things manually and have AI support.



Self-adaptive re-autonomy

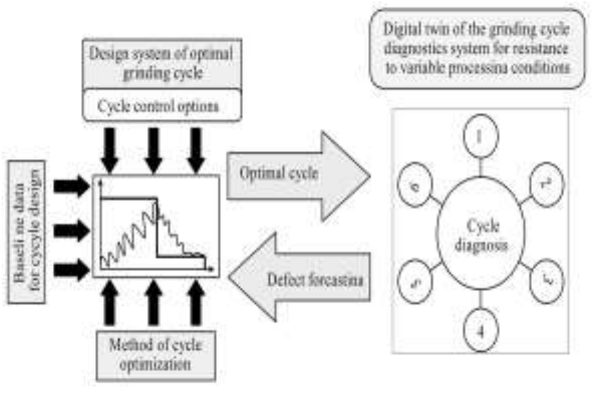
Explain quality assurance and improvement through dynamic programming.

1 Self-optimization of the architecture built-in or Fixed in IoT software system the design and implementation of such systems often need to respond to changes in the environment and conditions. Has properties like performance efficiency and energy efficiency So, keep on securing the desired quality with AI There are attempts to self-optimize the architecture so that for example 9 especially energy efficiency and Can you predict when you'll exceed your allowance for data traffic? Reinforcement learning reveals optimal architectural adaptive patterns selection and adaptation turn these self-adaptive practices Pairing can also be considered part of adaptive maintenance.

2 components reuse Proven data in software design and implementation design pattern and component library frame work's reuse only makes the development process more efficient Contributes to quality improvement including reliability Here, the application of machine learning is specifically Searching for a set of components matching a requirement or context, this is useful for evaluating reusability etc.

Automatic Programming (Generative and Synthesized)

There are also reuse and sequence generation Efficient and high-quality design and automated programming through generation and synthesis useful for implementation Exploratory generation by GA and machine translation generation through translation, etc. Sing individual solution candidates expressed as genes, Selected evolution-inducing genes based on fitness functions then repeat operation like crossover with other genes and mutations A genetic program that iteratively searches for an approximate solution Genetic programming (GP) extends GA Treat the representation of the tree structure of the program as a gene. It is a method of sequentially generating a program that turns out to be a solution. For example [again about GA target programming Induction of language-specific gene creation and deletion mechanisms Furthermore, it restricts the commands available on the programming language.


Design system of optimal grinding cycle
Cycle control options
Baseli ne data for cycyle design
Optimal cycle
Defect forcastina
Method of cycle optimization
Digital twin of the grinding cycle diagnostics system for resistance to variable processina conditions
1
2
3
4
5
6
Cycle diagnosis

To reduce the search space and meet simple requirements. GA based efforts that make it possible to generate code Incorporate features of programming languages accurately Expensive and of limited size and material by machine translation use of effort transformer Code table from a large collection of program source code Pre-learn and fine-tune the current language model A program through code generation and completion from requirements documents Translating code from one riming language to another and even program improvement conciseness/comprehension support identification of similar code fragments code BERT 1 2] and Cu BERT [13] known as Code BERT is a natural language description that describes the source code and its contents Bimodal learning by finding code based on demands and code generation, and vice versa from code to documentation trying to generate For example, the code T5 [14, as shown in Figure 1 In pre-training, we learned the correspondence between the natural language description and the code.

Masking and predicting identifiers like pre-training Through practice, highly accurate code translation is done considering the identifiers. Benchmark data for comparison and evaluation of methods Tastes have also been received 15 these attempts General-purpose programming language without manual feature design A large and high-quality code corpus that can be handled is the key for example in 16 there will be 54 million repositories on GitHub Acquisition of a language model from a total of 1 59 GB of code obtained from and use it to automatically extract the contents of the function from the function name and comments. GitHub Copilot, a service that is complementary.

Use of AI in testing and input with basic description and code

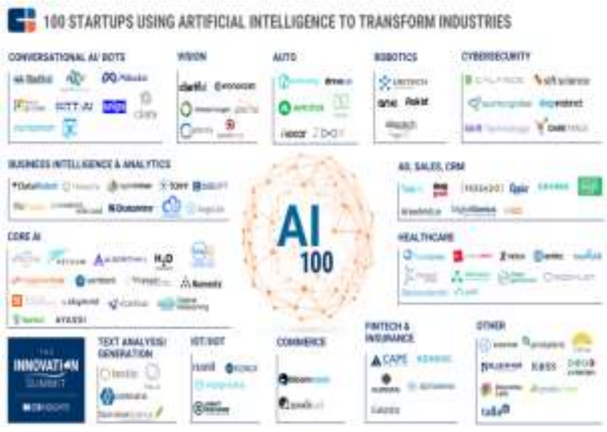
In dynamic verification of programs using test cases, all the following activities will be optimized and made effective using AI. Working on efficiency. one of the security tests AI use is also active to detect vulnerabilities in the form of Ring.

● Test Planning: Highly accurate prediction of test man-hours and defects

● Test Design and Execution: Test case generation and exploratory discovery-based testing included

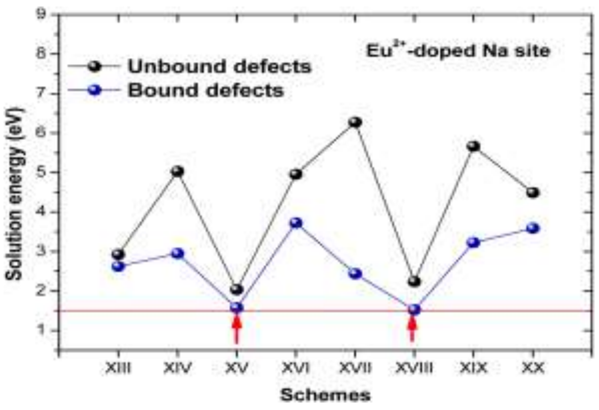
● Test iteration: for test case priority Efficient regression testing with test man-hour prediction the effort and effort put into testing affects the quality of the product. There is a report that this is the most deciding factor [17].

There In planning the test, type of project, requirements, Estimating the appropriate number of man-hours to scale is an important research topic. In general, situations where there is a certain amount of historical project data, Machine learning-based prediction should be machine learning-based. Predictors are known to outperform predictions (such as simple regression analysis). This type of test is particularly useful for man-hour forecasting projects. While planning the cost and schedule of the entire project, It is useful for building a team. For example, in [18] a test engineer for a public dataset Case-based as a machine learning algorithm for number prediction Case-Based Reasoning (CBR), Multi-layer Perceptron (Multilayer Perceptron; MLP), support vector Regression (Support Vector Regression; SVR), GP, Decision Tree (Decision Tree; DT), especially SVR, GP and DT Said to be excellent.



Defect Prediction (Bug Prediction)

Properly align the effort and activities required during the testing process Timely and highly accurate latent defect (bug) prediction for the target product is useful for management. by prophecy This will allow for a more thorough examination of the plan of the places, and review, design and re-examine the entire test plan, It is also possible to reduce capabilities. fault prediction model Dell believes it is a defect density model. software reliability development model; SRGM) [19]. Defect density model is mainly based on number of lines of code and input/output. and use product attributes as a machine learning algorithm latent defects for each module or part of the product due to Predict the presence and number of damages. Module prone to failure It is also called fault-prone module prediction. For example, [19] considers the utility cost of dealing with imbalanced data. Based on Cost-Sensitive Learning (CSL) Extended by large margin dispensing machine Distribution Machine (LDM) is used Imbalanced data set with 7% to 20% of the total We have been able to make high-precision predictions on the other hand, the reliability enhancement model is based on time-series fault detection data.



Using data to focus on future specifics Predict the cumulative number of defects discovered by time. typical A non-homogeneous Poisson overpass has been proposed as a predictive model in the fault detection process. Modeled by Non-homogeneous Poisson Process (NHPP) There is ornamentation and its elaboration. In addition to the number of defects detected, predictive models that take as input the characteristics of a project or process or person There is also a model that predicts by module [20]. more time series an approach to identifying more complex and unique patterns in columnar data. The combination is called data-driven SRGM, and is used in SVM, GA, neural Network (Artificial Neural Network; ANN) AI use is active [21]. For example, the author as shown in figure 2, the past Using time-series fault detection data from similar projects LSTM model by training new running projects [22]. 5.3 Test Case Generation Test schedules useful for defect detection can be generated from a near-infinite execution range. in choosing a suitable finite set of tests, symbolic execution, search-based test testing; set), combination

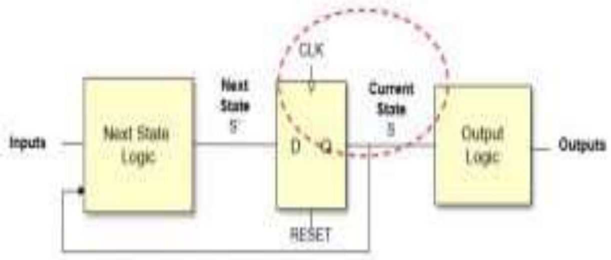
There are input value output methods like test, CT.

Especially with respect to SBT, AI is used to evaluate and discover useful input values. Applications are active. SBT is a measure of the degree of achievement of the requirements to be achieved. It can be evaluated objectively based on the designed evaluation function, I want to achieve using a predictive search algorithm It is a way of generating a set of test cases that satisfy the requirements. (see Figure 3). Mainly follow steps A to D below [23].

a) Evaluation function that quantitatively evaluates the degree of achievement of requirements (e.g. program path coverage, execution time)

b) Test suite in the form of a set of ready-made test cases Input and execute the test and get the value of the evaluation function

c) Based on test suite with excellent evaluation function values, the heuristic search algorithm finds to generate a new test suite (algorithm (e.g. GA/GP, particle swarm optimization, machine learning, etc.)



d) Repeat b and c until the search termination condition is met. Run (e.g. coverage ≥ 90%) SBT is actively involved in testing continuous control systems. It is implemented. For example, in [24], as shown in Figure 4, Genes can be defined as signal amplitudes, time widths, changes (e.g., step functions). number, sine function, etc.), and adding them up, A signal train is generated. Then, by applying gap, gene by rearranging, different signal sequences are generated automatically. Linking with model-based development tools such as Simulink They are active too. In [25], several studies of Simulink model testing are presented. Automatic generation of test input values by SBT considering the diversity It is showing. In particular, the structural network of the Simulink model However, the output signal is not sufficient for fault detection. The output varies depending on the characteristics and distance between the signals. originates in Improving search efficiency is an issue in the actual operation of SBT. Various attempts have been made. For example, in [26] Unrealistic input signal, taking into account the SBT of the control system To stop tests based on combinations (scenarios), Based on the actual driving position distribution obtained from the actual driving data, Search efficiency is improved by evaluating the keywords and prioritizing the search location. we are trying to improve Additionally, the values assigned to several parameters are a combination of to test the CT, the combinations are complete and the number is pressed. Attempts have been made to apply AI to derive a set of test cases, for example, [27] uses GA, and [28] uses other meta- It uses both the Lis tic algorithm and Reinforcement Learning.

Test Priority

Regression testing can be done if program changes are applied to existing machines. It should be verified that the changes were successful before committing to ensure that functionality is not adversely affected. This is to be confirmed by executing the existing test case group. Massive development and continuous building in agile development Continuous Integration (Continuous integration; CI), once build and Limited man-hours and time for regression testing is increasing. In such situation give priority to test case group Attempt to execute priority test from highest ranking It is becoming active. Finding the optimal solution is often NP-hard. Prioritization and selection by machine learning within the problem Advocacy efforts are active [29]. 5.4 Vulnerability Detection Large-scale analysis of known security vulnerabilities through machine learning Efforts are underway to automatically detect this on simulated code.

For example, in [30], as shown in Figure 5, a convolutional neural network Convolutional Neural Network (CNN) and Recurrent Neural Networks We use a neural network) to extract features on a distributed representation of the code. Identify and list common vulnerability types (common weakness Calculation; CWE) vulnerabilities are labeled and randomized Detected by Forest. here in machine learning Since [30] the quality and quantity of training data is an issue in rule the warnings and issues found by base static analysis tools are It is associated with each CWE, and based on the results of applying the tool, we've got a huge amount of correct data for vulnerable spots.

Use of AI in debugging Creating defects in code

mainly taking advantage of test results Defect localization that identifies locations and automated processing that implements the results Applications of AI for program modification are active.

Defect Localization (Bug Localization)

Fault localization (FL) includes dynamic testing. For test execution results, static code analysis results, and defect reports. statements and actions the root cause of the defect is determined in units of each program element in the code, such as Count the suspect as a causal factor. For example, Failed test cases, based on test run results the more elements that are executed by the static analysis results in the same Based on this, the more complex the element, the more suspicious it is. doubt it Based on ranking by manual or automatic the objective is to promote efficient program modification. However, there is no single FL method available that is suitable for all situations. No. Therefore, different conjectures are proposed based on different feature quantities. we add the values of Rank Learning (learning-to-rank) FL efforts are being studied. For example, extended in [31] Using Extended MLP, test execution results, complexity and parallelism High-precision FL is realized by various feature quantities.

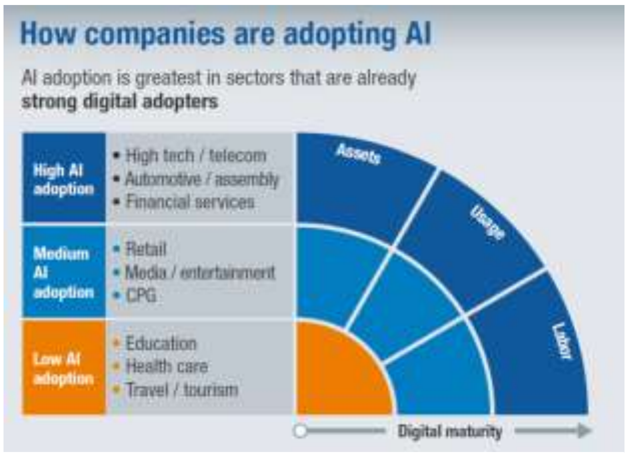
Automatic Program Repair (APR)

A technique for automatically correcting defects found in tests etc. Happen. For code, typically as shown in Figure 6, After the defect condition has been identified by FL etc., the correction patch candidate generate complements, rank them by machine learning etc., and test After checking the correctness by, we finally receive the revised patch. The generation of fix patch candidates is based on past fix trends. manually created conversion patterns/rules and templates, Automatically learned by applying machine learning to past improvement results Transformation, exploratory transformation by GA/GP and their combination This is realized by, for example, in [32] the variable type extension changes, modifications to conditional statements, changes/insertions in method execution, and A patch candidate group is generated using various conversion rules Then, a pre-trained logistic regression model Ranked by [33] in Encoder/Decoder Program before and after correction at the time of previous defect correction, by combining the data Learning the Correspondence of Village Abstract Syntax Tree with Surrounding Context and apply the patch to the new code. Candidates are generated and ranked by CNN. Many of the current APR methods are simple It only fixes minor defects, but sufficient quantity and quality of test cases By preparing the AI, we entrusted the correction of simple defects to the AI and developed it. The originator can expect a form that tackles complex problems. In [34] is the use of AI in the development of open source software. Submit an automatic correction proposal that is It has been reported that there are cases in which patients are accepted without child This can be said to be a form of value co-creation by AI and humans.

AI utilization in other quality control

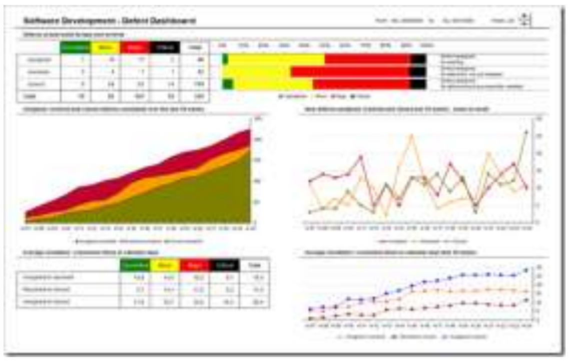
in quality control, other than testing and debugging, quality characteristics There are efforts to utilize AI for quality evaluation and defect report management.

Quality assessment and prediction in addition to functional suitability and reliability-based defect handling, Evaluation of various product quality characteristics by machine learning and future Prediction has been researched and practiced [36]. Machine learning can be used to evaluate quality. It is also useful for adjustment and adaptation of price standards. The authors for the evaluation, DT based on the review result by the developer Supervised learning by using We have implemented a method that incorporates existing features into criteria.



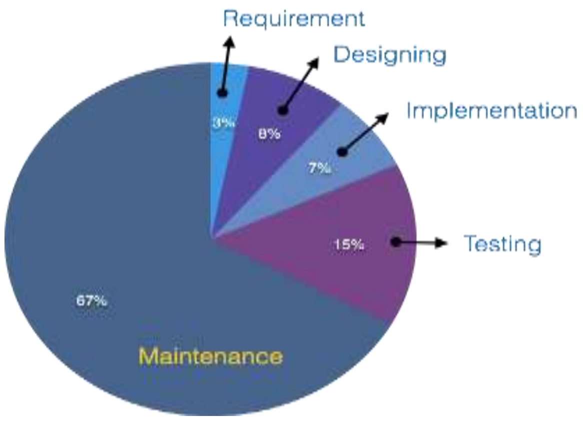
Defect Report Management

A large number of defect reports (bug reports and defect slips) are obtained. In a continuing situation, mechanics, as well as the treatment of requirements documents, Processing through learning-based natural language processing is useful. example for example, the authors applied BERT to eliminate duplicate defect reports. We have implemented a method to identify [38]. In particular, in the use of deep learning, explain ability is important for decision making. holds the key to As shown in Fig. 7, the authors used CNN to The defect report group is fixed in a short time and it takes a long time. When classifying into objects, we apply a visualization method to classify them. Based on the concreteness of the representation of defect reports in the model and being specific will save time (e.g. "using 19990914 build"), identifying a long tendency to be abstract.



Using AI for Quality in Maintenance & software maintenance-

Based on ISO/IEC 14764 2006 Corrective Maintenance Preventive Maintenance Adaptive Maintenance Emergency Maintenance the APR described above is also a helpful method for corrective maintenance. In addition, in preventive maintenance, design and implementation detects favorable and unfavorable sizes of It is important to continue using AI. .1 Finding Design Patterns etc. A specific preferred design that solves a common design problem. The shape of an aggregate is called a design pattern, and the result of its application is called a design pattern. By detecting this during loading, we can understand the intent and quality of the design, It is possible to move forward effectively with the maintenance of descendants. However, for specific applications Form topics vary according to constraints and requirements, and often Rule-based detection is difficult. Features such as the complexity and scale of known design patterns from various application results through learning by ANN of the measured values of A highly accurate automatic detection method has been implemented Similar detection efforts include: Complexity that makes maintenance a problem and Provisional Adverse Design, etc. (Design Proposal) (also called code smell, technical debt etc.) auto detection attempt 2 Improve and track design and implementation to eliminate smear and technical debt, external behavior Refactor code to improve internals while preserving is useful, and AI is used 4 3]3]. To improve It is often necessary to track and handle various artifacts, not just code. Traceability among deliverables. AI use is also active for identification.



Conclusion

In this paper Overview of AI Uses Related to Quality Assurance In particular, he explained the quality assurance activities that have made significant progress in the application. To promote inductive technology activities based on data, infrastructure to ensure the quality and quantity of data trained by its model quality (specifically predictive performance, robustness and interpretability) and the key to success is the correspondence of the achieved results to the goals Therefore, improvements in robustness and interpretability, such as machine learning and AI, Joint use of quality assurance technology is also essential. Synergistic effects should be demonstrated with AI quality assurance as two wheels of a vehicle. For example, the author Systematization of Machine Learning Design Patterns [4 5 We are working on multi-digit requirements analysis, and Waiting to join AI application efforts.

References

[1] D. Zhang, et al. al., (2005) “Machine Learning Applications in Software Engineering,” Series on Software Engineering and Knowledge Engineering- PP 16-19

[2] S. Shafiq, et al. al., (2007) A Literature Review of Using Machine Learning in Software Development Life Cycle Stages,” IEEE Access, PP- 11-23,

[3] K. Kaur, et al. al., (2021). “A Review of Artificial Intelligence Techniques for Requirement Engineering,” Computational Methods and Data Engineering, PP-56-58

[4] I. M. Subedit, et al. al. (2021), “Application of Back translation A Transfer Learning Approach to Identify Ambiguous Software Requirement nets,” ACMSE PP-23-34

[5] Z. Kurchatovia, et al. al. (2017), “Automatically Classifying Functional and Nonfunctional Requirements Using Supervised Machine Learning,” PP-78-79

[6] P. Teele, et al. al., (2021) “Classification and Prioritization of Software Requirements using Machine Learning g A Systematic Review,” Confluence.PP-67-69

[7] J. Devlin, et al. al., (2018) “BERT: Pre-training of deep bidirectional transformers for language understanding,” arXiv:1810.04805, PP- 78-82

[8] T. Hey, et al. al., (2020) “Norbert: Transfer Learning for Requirements Classification,” PP56-58

[9] H. Muccino, et al. al., (2020) “Leveraging Machine Learning Techniques for Architecting Self Adaptive IoT Systems,” SMARTCOMP PP-25-32

[10] D. P. Wando, (2018) “Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design,” ICCCA. PP-76-78

[11] K. Be clear, et al. al. (2021), “AI Programmer: Autonomously Creating Software Programs Using Genetic Algorithms,” GECCO.PP-45-48