iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** VIII **Month of publication:** August 2017

DOI: http://doi.org/10.22214/ijraset.2017.8050

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VIII, August 2017- Available at www.ijraset.com

Architecture and Design for Regression Test Framework to Fuse Big Data Stack

Shashank Sexena[1,] Pawan Gandhi[2]
[1,2] *Lead QA Engineer, Make My Trip India Pvt. Ltd, Gurgaon*

Abstract: Regression is a statistical measure that attempts to determine the strength of the relationship between one dependent variable and a series of other changing variables (known as independent variables). Big Data bring to real world with paradigm to automatically parallelized tasks and executed them on a large cluster of commodity machines. Regression is part of integrity to bring test automation faster and become need to achieve target of reinvesting resources at lower cost for huge clusters. Our Architecture and design of regression framework to support big data stack on one location. Provide flexibility to compile big data ecosystem components together to allow processing proof and data accuracy. Programmers will find it smooth to implement and ease to integration with big data stack.

Keywords: Big Data Quality, Quality Assurance, Regression Test, Test prioritization, Data Validation, Regression Test Framework

I. INTRODUCTION

With the increase of big data applications in diverse domain, Big Data computing and service is becoming a very hot research and application subject in academic research, industry community, and government services. According to IDC forecasting [2] that the Big Data technology market will "grow at a 27% compound annual growth rate (CAGR) to $32.4 billion through 2017. Today, with the fast advance of big data science, analytics and technology, more and more researchers, businesses, and application professionals are able to use diverse data mining and machine learning algorithms, open-source platforms & tools, as well as cloud database technologies. This suggests that big data computing and application services bring new business opportunities and demands in people's daily life. Quality big data and service applications can help enterprises in problem identification, process improvement, productivity increase, efficient customization support, intelligent decisions, and optimized solutions. However, due to the huge volume of generated data, the fast velocity of arriving data, and the large variety of heterogeneous data, the quality of data is far from perfect[1]. Since data is an increasingly important part of big data based application and service systems, data quality becomes a major concern.

Although there has been a numerous of published papers [3][4][5][6][7] [8] addressing data quality and data quality assurance in the past, most of them focus on conventional data quality and quality assurance for traditional data warehouses. There is an emergent need in research work to quality study issues and quality assurance solutions for big data and related application services. This paper is written to determine one of the major goal of validating the data quality. There are number of major challenges and needs in big data quality validation and assurance [9]. This paper is written to provide our perspective approach to big data validation. It represent our informative insights with major components that comprise a big data stack to validate. Rest of the paper will talk about quality measures for big data validation. Architecture and design for validation components fused to big data stack to cook the desired results for functional validations.

II. BIG DATA QUALITY

A. *Data Quality Parameters*

According to ISO 9000:2015, data quality can be defined as the degree to which a set of characteristics of data fulfills requirements. According to [10], *data quality assurance* is the process of profiling the data to discover inconsistencies, inaccuracy, incompleteness and other anomalies in the data, as well as performing data cleansing activities, data aggregation, data transfer et al.to improve big data quality.. In general, *big data quality assurance* refers to the study and application of various assurance processes, methods, standards, criteria, and systems to ensure the quality of big data in terms of a set of quality parameters. There are various data quality parameters need to consider.

1) *Data accuracy* - which refers to the closeness of results of observations to the true values or values accepted as being true. This quality indicator usually is used to measure collected sensor data by comparing multiple sources. \

2) *Data currency and timelines* - David Loshin in [11] explains the two concepts. According to him, data currency - refers to the degree to which data is current with the world that it models. Currency can measure how up-to-date data is, and whether it is correct despite the possibility of modifications or changes that impact time and date values. Currency rules may be defined to assert limits to the lifetime of a data value, indicating that it needs to be checked and possibly refreshed. Loshin also noted that timeliness refers to the time expectation for the accessibility of data. Timeliness can be measured as the time between when data is expected and when it is readily available for use.
3) *Data correctness* - This data quality factor is useful to evaluate the correctness of collected (or acquired) big data sets in term of data types, formats, profiles, and so on
4) *Data consistency* - which is useful to evaluate the consistency of given data sets in different perspectives. For example, in a repository, data consistency refers to data type and format. In a data collection system, data consistency refers to data collection approach, schedule, and location.
5) *Data usability* - This quality factor indicates the degree about how well the given big data sets have been used in big data analytics for application services. This could be an important quality factor for data users.
6) *Data security & privacy* - This quality parameter could be used to evaluate the security and privacy of the given big data sets in different perspectives. This quality factor is very important for big data analytics services, but also big data posting and sharing.
7) *Data completeness* - which is a useful indicator to measure the completeness of big data sets within a selected domain. As indicated in [12], data completeness is a quantitative measure that is used to evaluate how many valid analytical data were obtained in comparison to the amount that was planned. Data completeness is usually expressed as a percentage of usable analytical data.
8) *Data accessibility* - This indicator is useful to evaluate a given big data set about how easy to be accessed by enterprises and public users.
9) *Data accountability* - This factor could be a critical successful indicator for any big data providers. Big data accountability could be defined and evaluated in different ways. For example, it could be evaluated in: a) data collection and pre-processing; b) data management and processing [15]; or c) analytics application services.
10) *Data scalability* - This is a distinct quality factor for big data service, and it is useful to evaluate how well big data are structured, designed, collected, generated, stored, and managed to support large-scale services in data achieving, access, transport, migration, and analytics.

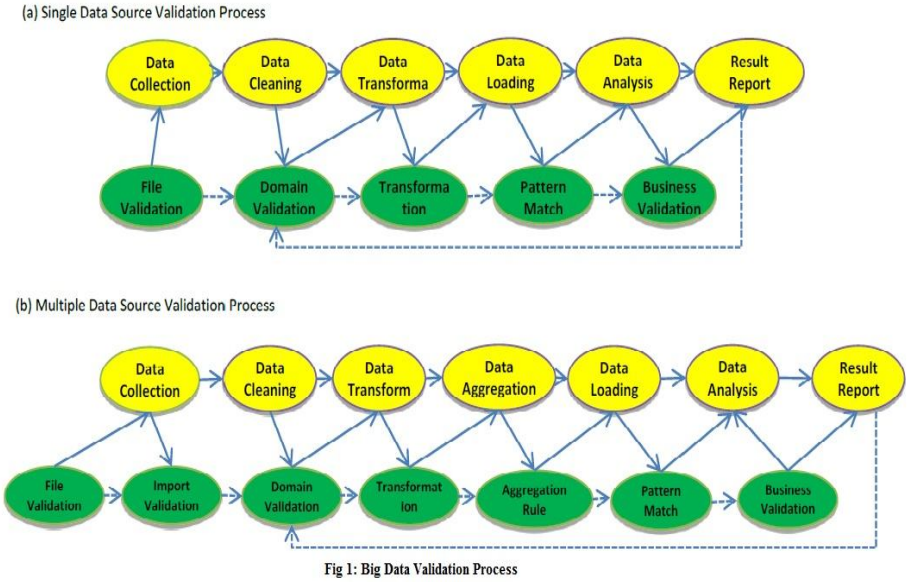


Fig 1: Big Data Validation Process

B. Big Data Quality Validation Processes

This section discusses the validation process for big data services [9]. As shown in Fig 1, there are five types of big data services: a) data collection, b) data cleaning, c) data transformation, d) data loading, and e) data analysis. For multisource, data need aggregation before loading. The detailed illustration as follows.

1) *Data collection* is the process of gathering and measuring information on targeted variables in an established environment which allow business to provide effective outcomes
2) *Data cleaning* is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database. The major purpose is to detect and identify incomplete, incorrect, inaccurate, irrelevant, data parts in data sets and they can be replaced, modified, or deleted [14].
3) *Data transformation* converts a set of data values from the data format of a source data system into the data format of a destination data system [15]
4) *Data Loading* refers to data loading activities in which data are loaded into a big data repository, for example, a Hadoop environment or a NoSQL Big database. Depending on the requirements of the organization, this process varies widely. Some data warehouses may overwrite existing information with cumulative information; updating extracted data is frequently done on a daily, weekly, or monthly basis. Other data warehouses (or even other parts of the same data warehouse) may add new data in a historical form at regular intervals—for example, hourly, daily or weekly
5) *Data Analysis* - a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains [15]
6) *Data Aggregation* refers to the compiling of information from databases with intent to prepare combined datasets for data processing [16].bThe typical validation process includes file validation, domain validation, transformation validation, pattern match, and business validation.

III. REGRESSION TEST TECHNIQUES

The regression term coined for investigation of relationships between variables. Usually, quality analysis seeks to ascertain the causal effect of one variable upon another. Regression test verifies that system in previous release or build accomplish exactly same results compared to previous developed releases.
These areas may include functional and non-functional areas of the system or software.

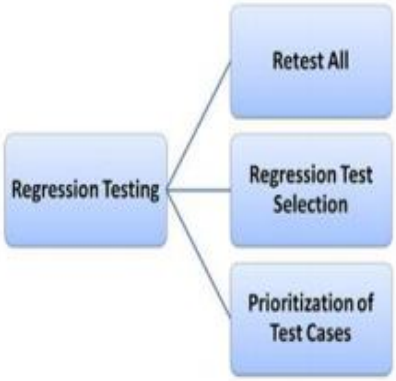


Fig 2: Regression Testing Techniques [18]

Regression Testing can be carried out using following techniques [18] as illustrated in Fig 2.
Retest All - This is one of the methods for regression testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources due to execution of unnecessary tests. When the change to a system is minor, this strategy would be rather wasteful.
Regression Test Selection - Instead of re-executing the entire test suite, it is better to select a part of test suite to be run. Test cases selected can be categorized as:
Reusable Test Cases - Re-usable Test cases can be used in succeeding regression cycles. b) Obsolete Test Cases - Obsolete Test Cases cannot be used in succeeding cycles.
Prioritization of Test Cases - Prioritize the test cases depending on business impact, critical & frequently used functionalities. Selection of test cases based on priority will greatly reduce the regression test suite. In big data scenarios, prioritization give advantage in time saving.

IV. TEST CASE PRIORITIZATION

Talking about big data quality involves the test cases that build the data quality. Therefore, this is important to look at prospect of test case priorities to ensure better quality. Typically, Test case is the triplet [I,S,O], where I is the data input to the system, S is the state of the system at which the data is input, and O is the expected output of the system. Test suite is the set of all test cases with which a given software product is to be tested [17]. Test Cases represent a set of conditions or variables in which a tester will determine whether a system under test satisfies requirements or works correctly.

Test case prioritization techniques provide a way to schedule and run test cases which have the highest priority in order to provide earlier detection of faults. Each test case is assigned a priority. Priority is set according to some criterion and test cases with highest priority are scheduled first. There are various prioritization techniques. Some of them discusses here:

A. Fault Severity

Test case prioritization depends upon the severity of faults. The order of execution of test case depends on the size of the test suite and how long each test case takes time to run [19]. Thus, through the use of an effective prioritization technique, testers can re-order the test cases to obtain an increased rate of fault detection.

B. Code Coverage Technique

Test coverage analysis is a measure used in software testing known as code coverage analysis for practitioners. It describes the quantity of source code of a program that has been exercised during testing [20]. The following lists a process of coverage-based techniques: a)

Finding areas of a program not exercised by a set of test cases

1) Creating additional test cases to increase coverage, c) : Determining a quantitative measure of code coverage, which is an indirect measure of quality, d) Identifying redundant test cases that do not increase coverage.

C. Mutation Faults

Test cases are prioritized by FEP (Fault-Exposing-Potential) Technique. This technique is achieved by the ability to expose faults and mutation analysis is used to determine this value. Each application of a mutation operator will create a mutant of the source code, which makes a single valid syntactic change. The mutation score represents the ratio of mutants that a test-suite can distinguish from the source code. The mutation score can be calculated for each test case separately [21].

FEP is calculated as:

$$FEP(t_i, s_j) = \frac{|\text{mutants } (s_j) \text{ killed by } t_i|}{|\text{mutants } (s_j)|}$$

$$FEP(t_i) = \sum_j FEP(t_i, s_j)$$

Given program P and test suite T, for each test case $t \epsilon T$, for each statement s in P, determine the mutation score of t on s i.e. the ratio of mutants of s exposed by t to total mutants of s. A mutant is killed by a test case if the original and the mutated program give different outputs when the test case is executed.

D. Customer Requirement-Based Prioritization

Customer requirement-based techniques are the methods to prioritize test cases based on the requirement documents. Many weight factors have been used in these techniques, including custom-priority, requirement complexity and requirement volatility [22]. Prioritization techniques use several factors to weight (or rank) the test cases. Those factors may be customer-assigned priority (CP), requirements complexity (RC) and requirements volatility (RV). The scaling factor can be assigned value (1 to 10) for each factor for the measurement. Highest factor values indicate a need for prioritization of test case related to that requirement. If requirements have high complexity then it leads to maximum number of faults.

E. History based Technique

History based techniques are the methods to prioritize test cases based on test execution history. It can enhance the effectiveness of regression testing. For a test case, using historical information of each prioritization, increase or decrease its likelihood in the current test session. A probability value is given to the history and the value of the current session is calculated based on the number of the fault detection (or coverage).

Finally, the calculated value of a test case is equal to the sum of the current number multiplied by a probability and the historical value multiplied by another probability [23].

F. Cost Effective-based Technique

Cost effective-based techniques are methods to prioritize test cases based on costs, such as cost of analysis and cost of prioritization [24]. The cost of a test case is related to the resources required to execute and validate it. Cost-cognizant prioritization requires an estimate of the severity of each fault that can be revealed by a test case. To minimize the costs associated with testing and with software failures, a goal of testing must be to uncover as many defects as possible with as little testing as possible i.e. write test cases that have a high likelihood of uncovering the faults that are the most likely to be observed as a failure in normal use. The cost–benefits should be considered first before they are applied to large programs.

Regression test cycles for big data use cases run through framework based on these test case prioritization are effective and design take care for use case based on these techniques. Flows carry out in different format like Csv, Json or Avro belong to respective and effective test case technique. For example, use cases based on trained data implies with machine learning algorithm suitable for history based technique.

V. REGRESSION TEST FRAMEWORK: ARCHITECTURE AND DESIGN

Design and architecture presenting here support big data stack with ease to integrate real world use cases for release quality. Be it related to field of industry, community, and government services, framework can handle the data sources and process them to provide the fruitful results in timely fashion. Speed up the functional and regression cycle in the world of automation focused testing goals. Here, big data technologies fused to common Continuous Integration (CI) and testing tools to provided end to end execution picture to establish the desire results for test regression cycles.

The Architecture defined the interaction of tool with external world. External world belongs to Big Data eco-system composed of various technologies involved to build the tool. Design for Regression Framework which fuses big data stack for functional and data flow analysis illustrated below in the Fig3.

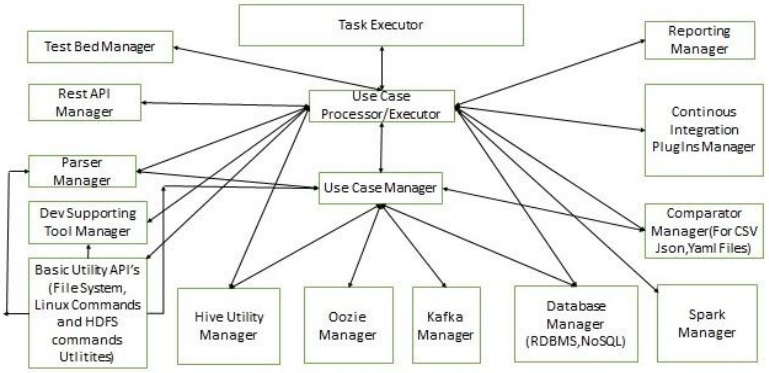


Fig3: Design and Architecture- Big Data Regression Test Framework

Regression Framework has been created to complete functionality tests and data validations for flows across to achieve data quality on various different Big data stack components i.e. HDFS, Hive DB, NoSQL DBs, parquet/orc/Avro/csv/json format files, Json, Yaml or Xml files. The design and architecture presented here is composed of various functional managers to fuse all the required components and arrows defined the interactions among the component managers defined below.

A. Test Bed Manager

This component keep in design to make responsible for selection of Test Bed. Test bed refers as an executing environment configuration for a testing environment. It may consist of specific software or hardware topology, and other specific configuration that required for real time use cases. Manager can keep various provided test bed information for regression tests. A scalable component to iterative and interactive approach to fuse use cases with provision of environmental details.

Test Case Manager This component in design provide flexibility of selection of test cases on the basis of test case prioritization. Prioritization varies on the basis of batch processing, real time or trained data machine leaning processing. So, each test case is associated with test management tool where repository of every test case resides. Responsibility of this manager lies in the window of selecting suitable test cases for big data quality use cases.

Use Case Processor/Executor: This component play critical role in the design where trigger mechanism build for running and processing of test cases This provide interaction of all components including big data utilities, Operating system utilities or test utilities. Every use case touch the physical and logical processing with respective processor and provide the result set for big data quality, iteratively.

B. Task Executor

This component provided in design for scheduling and performing tasks. A critical component in the sense of behaving user interface and abstract all the complexity of execution from users. Programmer create one executor per environment. All the fused use cases processed just behind this layer from user endpoint. Use case processor took the trigger action from task executor. There are various open source schedulers like cron, Oozie or Azkaban can be integrated to achieve the task executor.

C. Parser Manager

This component in design provide handling of Csv files, Yaml, Xml or Json file parsing, Parent data cleaning action to achieve data quality serve from this manager action. Common parse library from programmer contribute to parsing logic which get integrate on use case processor.

Comparator Manager

This component provide comparison of result set like test validators with Development API. Comparator manager is necessity to provide report for any type of test case. Test case execution status achieved with this module. This manager act as library for comparison for different file or data types. Ingested data or processed data screened from comparator to achieve the data quality goals.

D. Rest API Manager

This module place in design with keeping in mind that responsible for Rest API support. This provided in design because in today's world, every other interface rely on REST. Most efficient and convenient way to integrate the applications. Therefore, all the required methods in library can be integrated from this manager.

Basic Utility Manage

Utilities for filesystem, Linux OS and HDFS are essentials in deal with big data functionality. This manager keep the common library segregated for data placing. In big data, when efficient space or proper processing capabilities require then this utility in design help to transfer data from one source to other for multi-tenant facility. Churning and processing on local file system or on distributed file system clubbed for utility manager.

E. Hive Utility Manage

Intent to keep hive utility separate because HiveQL or any other fast processing components like Presto DB, Cloudera Impala, sparkSQL can utilize or interact with hive meta-store. All the plugins require for access big data on query module can be achieved from here. Regression test cases for use cases of machine leaning or batch processing utilize this module efficiently for larger set of data.

F. Oozie Manager

This manager taken from open source popular big data tasks scheduler name Oozie. Therefore, keeping it differently from task executor to provide the distinct numerous features of batch processing that provide big data. Scheduled tasks from Oozie or any other scheduler like Azkaban provide data for use case processor in big chunk. Scheduler support to achieve data granularity according to limitation of environment varied for use cases. Test cases prioritization arranged with scheduled tasks.

Spark Manager: To run Real-time tasks, spark is most common framework today. Design exhibit a single manager for real time data quality support spark streaming output. These output data required less space but high memory requirement to validate through use case processor

Persistent connections with socket to listen the data ingestion or processing achieve from such manager to ease real time monitoring.

Kafka Manager: To interact with Kafka messaging queues, this manages connections for producer, consumer and Clients to interact the data consumption. In real time data processing, Kafka play critical role in most of the data analysis pipelines. Data ingested in Kafka brokers can be capture from this manager and enhances the data quality as probe on Kafka flows.

Database Manager

This manager is essential to connect with RDMS like MySQL or PostgresDB In most of the approaches, RDBMS provide Meta information in big data. Hence, connectors can be treat as separate entity for data reading and writing for RDMS. Similarly, massive data output to fetch for data analysis available to noSQL databases like Hbase, Cassandra. So, connectors for NoSQL designed and provide library through this manager. Test cases interacted with NoSQL or Meta information achieve isolation from other big data technology if this manager available in design as independent.

Reporting Manager

User interface for result set, a must require module for any testing framework. Various open source test management tools provide essential reporting mechanism. So, to leverage such tools, integration layer to test management tools achieve with reporting manager. Reporting manager can be treated as endpoint for the regression test framework where every test case publish the result set. Reporting module track the type of result set and publish it to specific medium. For instance, table result set can be directed to email or document like applications. However, assertions output could be design to integrate with test management tool. Reporting manager as part of design provide regression results on every iteration of task executor to configured endpoint.

Continuous Integration Plugins Manager: In today's era where fast development cycles are require to achieve desired results, continuous integration play an important role. Every build need to gone through testing phase in continuous integration so that early bug detection can be achieve. However, if include big data use cases, to ensure data quality for every use cases become tedious task for every build. Therefore, provision to integrate continuous integration tools like Jenkins with big data environment achieve in regression test architecture. Tools like Jenkins provide flexibility to interact with build processes and test processes. Big data use cases coupled in test processes leverage build features from Jenkins through this manager.

Dev Supporting Tool Manager: Development supported common repository require to provide use case specific requirements like security, encryptions, data headers, in built development library or patent based application usage. In any organization, big data use case can be integrate with these tools, separately and do not disturb the execution of other use cases which not dependent on these supporting. This manger available in design to fuse such critical tools with test cases and prioritize these functionality among test case manager. Keeping such library in separate manager help organization to achieve security isolation.

VI. DATA FLOW AND INTEGRATION ANALYSIS

Ingested data from different sources stored at Hadoop distributed file system (HDFS) for batch analysis. Kafka Streams are available on real time or near real time data. Thus, an application server reside on remote node in big data ecosystem which can have direct access to Hadoop, Hive, Spark, Kafka, NoSQL, relational DBs and other integral part of data platform eco-system. The application node from execution manager launches the test suites contained the executable test cases. Each test cases gel up with utilities required for the use case and process together within use case processor. Utilities manager loosely coupled and well bind with test case to provide the desired results. Let's illustrate one use case where a chuck of million records are processed in MapReduce paradigm and sources consist of data reside on HDFS and annotation require from static files placed on local filesystem or schema based in RDMS. So, Test case launched from execution manager connect with all the required location remotely and placed the actual processed logic on HDFS. According to expected baseline, methods associated with comparator manager call in the use case processor and passed the desire result to reporting manager. Reporting manager tends to publish the results on email or connect to test management tool and updated the test runs with actual results within quick pace of time. In iterative manner, all the desired test cases will be executed and provided the complete report for validation results.

Continuous Integration apply to use case which included test cases regression cycle on every build achieve with plugin manager. Each big data development build cycle triggers the regression test framework executor and associated test cases run in prioritized iteration and publish result on reporting tools.

VII. ADVANTAGES

A. Open Source

Being in the world of open source, technologies available to support all the functionality illustrated in architecture. These days for virtually every paid for proprietary software system you will find an open source version. Therefore, better to choose open source technologies to build the test framework for cost effectiveness.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VIII, August 2017- Available at www.ijraset.com

B. Ease of use case integration

Use cases to integrate for Regression tests can be implanted with ease. Integration of use cases attain in test framework with test case manager and use case processor.

C. Scale-ability

Something that has been used and tested by a bunch of other programmers, far more likely to scale better than something you come up with alone.

D. Execution Velocity

Speed up the development cycles for big data use cases essential to achieve goals in timely fashion. Execution of each cycle become faster with usage of regression framework build for big data engineers.

E. Setup Time saving

Big data require component setup time which may rise exponentially with increase of components. When fusion of big data components to test framework achieve, it reduce ample of setup time. Programmer increase the component coverage with each business use case integration.

VIII. RESULTS

In pre automation era, lot of manual efforts required to test big data use cases like data preparation and generation, schedule the jobs, and execute the jobs. Functional test and regression test cycles took ample of time in yield the desired results. Preserve the processed results and compare the results for each data model then publish the final report. So, for say, consider 7 functionalities, Total efforts can be seen with graphical representation. Various functionalities took handsome 7 days of efforts in aggregated to 33 days for functional test and 19 days of regression test efforts.

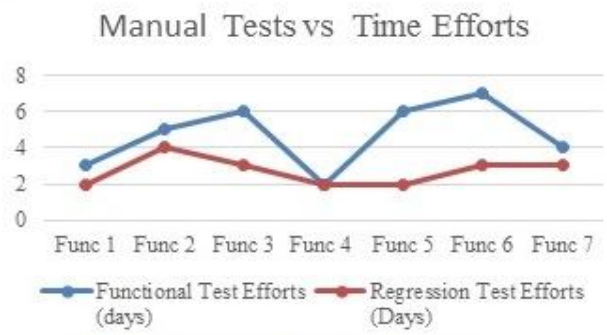


Fig 4: Manual Efforts Estimation

To overcome such time efforts, designed the automation in optimized manner to support integration of real world use cases with ease of use. With real world use cases of e-commerce stream, we have achieved the results for functional and regression efforts within aggregated 30 minutes. So, overall time gain equivalent to 50 days of efforts. Graph representation below shows that time cycle overlapped for functional tests and regression tests execution.

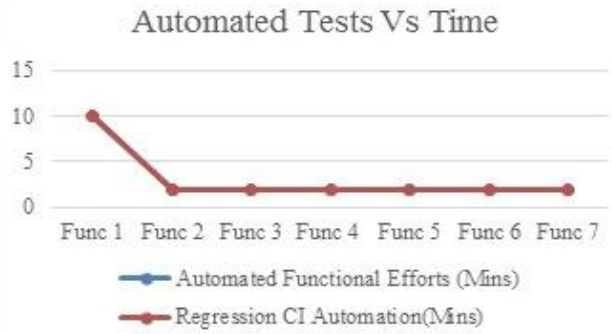


Fig 5: Automated test estimations

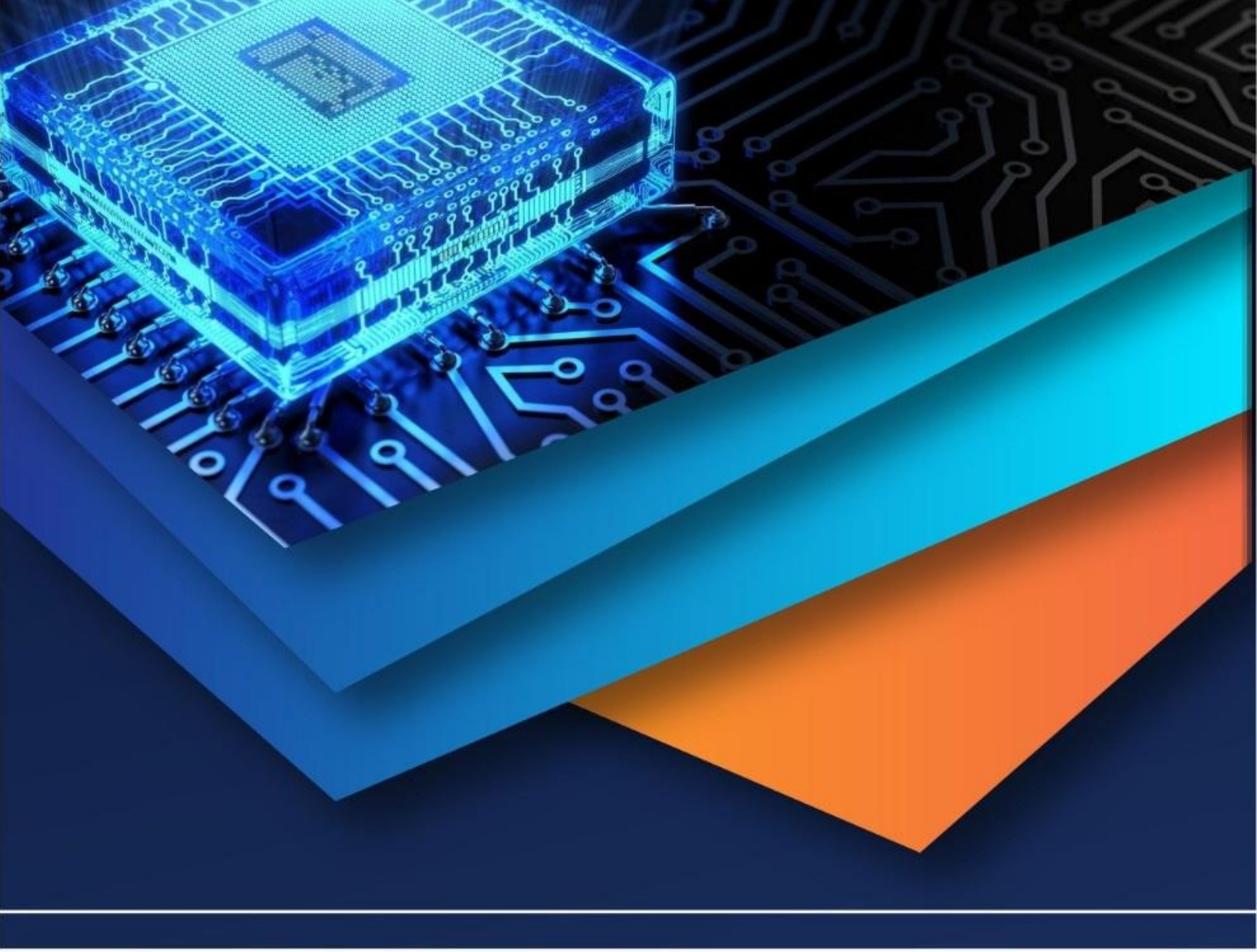
Result set for executed cycles are published through reporting manager. This stage manages test tools and communication tools to publish test results. Use case processor passed the actual result to manager. Now, resulted data processed for publishing on test tools. For instance, consider test rail where REST APIs are available to integrate. All the test cases configured to run have specific IDs in test management tool. So, test case IDs treated as primary key to publish the results. Test management tool provided the test runs functionality where test run IDs are tightly coupled with test cases ID. So, each execution cycle associated with test run mapped the results and published the results on test management tool directly. So, reporting feature on test management tool supported in design and ease the integration of use cases with test management tool reporting. Similarly, if results need to publish on email, then SMTP protocol supported in the reporting manager. Each execution cycle capable to publish results on configured emails. Treat the email as alerts for each execution cycle.

IX. CONCLUSION

In this paper, architecture and design to fuse regression testing for big data stack presented. There are various factors on the basis of which test case prioritize can be decided for variety of big data use cases which can strengthen regression testing for finding more severe fault in earlier stages. With earlier catch of bugs, defects or errors in big data development cycles, we can reduce cost, time and effort and hence, achieve maximize product quality. A system can be developed using open source technologies available as discussed in paper to analyze the test case prioritization after performing regression testing on the developed software. If we have a large test suite then we can implement the clustering to categorize the data faults and report them early with implementation of regression test framework discussed.

REFERENCES

[1] Steve Sarsfield, “How a Small Data Error Becomes a Big Data Quality Problem”, September 3, 2012. Retrievable at
[2] https://www.linkedin.com/grp/post/103526-162333712.
[3] Editor of Business Wire, December 18, 2013, [Online]-http://www.businesswire.com/news/home/20131218005150/en/IDC-Worldwide-Big-Data-Technology-Services-Forecast
[4] M. R. Wigan and R. Clake, “Big Data’s Big Unintended Consequences”, IEEE Computer, Vol. 46, Issue 6, Pages:46-53, 2013
[5] D. M. Strong, Y. W. Lee, R. Y. Wang, “Data quality in context”. Communications of the ACM, Vol.40, Issue 5, Pages: 103-110,1997.
[6] L. L. Pipino, Y. W. Lee, R. Y. Wang, “Data quality assessment”. Communications of the ACM, Vol.45, Issue 4, Pages: 211-218, 2002.
[7] A. O. Mohammed, S. A. Talab, “Enhanced Extraction Clinical Data Technique to Improve Data Quality in Clinical Data Warehouse”. International Journal of Database Theory and Application, Vol.8, Issue 3, Pages: 333-342, 2015
[8] M. I. Svanks, “Integrity analysis: Methods for automating data quality assurance”. Information and Software Technology, Vol.30, Issue 10, Pages: 595-605, 1988.J. Alferes, P.Poirier, C. Lamaire-Chad, et al. “Data quality assurance in monitoring of wastewater quality: Univariate on-line and off-line methods”. Proceedings of the11th IWA conference on instrumentation control and automation, 18-20, September, 2013
[9] A.L.Ali and F.Schmid, “Data quality assurance for volunteered geographic information”. Geographic Information Science. Springer International Publishing, Pages126-141, 2014
[10] Jerry Gao and Chuanqi Tao “Big Data Validation and Quality Assurance - Issues, Challenges, and Needs”. 10th IEEE International Symposium on Service-Oriented System Engineering, March 201
[11] J. J. Gassman, et al. “Data quality assurance, monitoring, and reporting” Controlled clinical trials, Vol.16, Issue 2, Pages: 104-136, 1995.
[12] A. Immonen, P. Paakkonen, and E. Ovaska, “Evaluating the Quality of Social Media Data in Big Data Architecture”. IEEE Access, Volume 3, Pages: 2028 - 2043 October 16, 201
[13] New Jersey Department of Environmental Protection, Data Quality Assessment and Data Usability Evaluation Technical Guidance, April 2014
[14] Katarina Grolinger et al, “Challenges for MapReduce in Big Data”, IEEE World Congress on Services (SERVICES), Pages: 182-189, Anchorage, AK, June, 2014.
[15] W. W. Eckerson, “Data quality and the bottom line: Achieving business success through a commitment to high quality data”. Data, Warehousing Institute, 2002
[16] J. W. Osbourne, -Notes on the Use of Data Transformation‖, Practical Assessment, Research & Evaluation, 2002, 8(6): n
[17] I. Taleb, R. Dssouli R, and M. A. Serhani, “Big Data Pre-processing: A Quality Framework”, IEEE International Congress on Big Data, Pages: 191
[18] Rajib Mall, “Fundamentals of Software Engineering”, Third edition, prentice hall of India, August 2011
[19] Guru[99], What is Regression Testing? TestCases, Tools & Examples [Online]. Available:
[20] Jaspreet Singh Rajal, Shivani Sharma “A Review on Various Techniques for Regression Testing and Test Case Prioritization”, International Journal of Computer Applications (0975 – 8887), Volume 116 – No. 16, April 2015
[21] Hyunsook Do and Gregg Rothermel, “A Controlled Experiment Assessing Test Case Prioritization Techniques via Mutation Faults”, Proceedings of the IEEE International Conference on Software Maintenance, pages 411-420, 2005
[22] Elbaum, S., Malishevsky, A. G., & Rothermel, G.,"Prioritizing test cases for regression testing", ACM,Vol. 25, No. 5, pp. 102-112,2000.
[23] G. Rothermel, R. Untch, C. Chu, and M. Harrold, “Test Case Prioritization”, IEEE Transactions on Software Engineering, vol. 27, pp. 929-948, 200
[24] Jung-Min Kim and A. Porter.,"A history-based test prioritization technique for regression testing in resource constrained environments", ICSE ’02: Proceedings of the 24th International Conference on Software Engineering, pages 119–129, New York, NY, USA, 2002. ACM Press.
[25] Alexey G. Malishevsky, Gregg Rothermel and Sebastian Elbaum, “Modeling the Cost-Benefits Tradeoffs for Regression Testing Techniques", Proceedings of the International Conference on Software Maintenance (ICSM’02), 2002.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY