ISSN 2278-3091
Volume 9 No.2, March -April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse123922020.pdf
https://doi.org/10.30534/ijatcse/2020/123922020

Continuous Testing Real-Time Health Analytics Dashboard

Jayasri Angara[1], Srinivas Prasad[2]
[1]KLEF (KL Deemed to be university),AP, India, angara.jayasri@gmail.com
[2]GITAM University, India, srinivas.prasad@hotmail.com

ABSTRACT

The goal of Continuous Testing is to take full advantage of iterative development and attain the time-to-market objective. However, Continuous Testing becomes a bottleneck and reduces the speed of the project. In that context, project monitoring and measurement is a herculean task for the project managers. There is a need for well-designed metrics and standards which should consider change causing factors and project interdependencies. Software project success depends on how well these metrics measured on a real-time basis. The Real-Time Project Metrics Dashboard becomes an important tool to monitor project by all important stakeholders (Customers, Project Managers, Dev-Test-Ops Teams, Management, etc). This paper presents the design and development of various metrics and data points related to continuous testing in the DevOps setting. This paper presents more than 42 key metrics/data points and 150 ancillary metrics/data points. This paper also presents the key algorithms developed for implementing these metrics. These metrics are generated using illustrative project datasets and published using Django-Python web Framework.

Key words : Continuous Testing, Agile Testing, DevOps Metrics, Software Metrics

1.INTRODUCTION

DevOps is an emerging cross-disciplinary philosophy. It enhances communication and collaboration between Business, Development, Testing and Operations teams. Continuous Testing is defined as a software testing process which promotes test early and tests often. The role of continuous testing is to cut down the development cycle, increase the number of releases so that business can reach the market faster. In Continuous Testing, deployment takes place early in the lifecycle, detect defects early and reduces the cost of fixing. Teams are able to release code at any point of time in this model. Continuous testing demands quantitative and qualitative assessment of all the risks and their mitigation plans before the project moves to next sprint [1]. This type of testing makes the developer code faster and write better code [2].

The success of Continuous Testing lies in how well the relevant project information is displayed to all project stakeholders, how well test cases are designed, prioritized and allocated to the teams, how well risk zones are identified and alerted stakeholders. Ultimately, it reduces the feedback loop, improves quality and organization performance.

The objective of this paper is to design critical continuous testing metrics in the DevOps context and present in the form of real-time application health analytics dashboard. This paper is organized as follows. Section II presents related work. Section III proposes a conceptual design of various testing metrics and real-time implementation. Section IV presents the threats to validity and Section V presents the conclusion.

2.RELATED WORK

The primary goal of Continuous Testing (CT) is to assess business risk coverage. CT establishes a safety net to protect the user experience from accelerated development processes. CT has become part of the development process. It evaluates each layer of modern software architecture at the appropriate phase of the software life cycle. It reduces false positives and eliminates redundancy [3]. Business demands uninterrupted service with seamless continuous integration of service upgrades. This model results in shorter, frequent and efficient releases[4]. This type of releases is only possible through continuous testing. Continuous Testing brings three major business benefits - the decision to go or no go in SDLC, new features to market faster, the trade-off between time, quality and functionality [5]. The impact of frequent releases should be well managed. Typically impact could be from technical factors, organizational factors, and interactional factors. If we go little detailed, they are connected to one of this four dimensions-security, velocity, productivity and quality[6]. The negative impact could be contained through proper monitoring of metrics [7]. Continuous testing needs systematic stitching between people, processes, and technology[8]. Continuous Testing is successful when it follows a systematic hierarchical test strategy [9]. Domain understanding and grasp on application behavior are needed for the teams in order to manage software development, testing, and maintenance. It is critical for continuous testing. It ensures high coverage, early detection of defects, better utilization of resources and seamless communication between business users, domain experts, testers and developers [10].

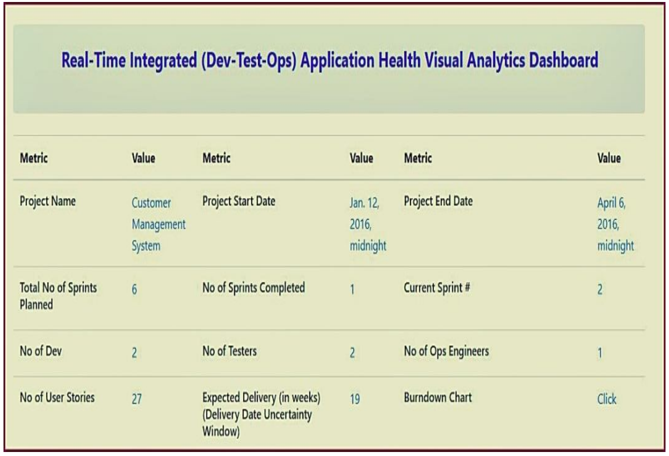
Communication and Collaboration are critical in the continuous testing process. Metrics and Dashboards provide confidence and action among all stakeholders. It should be real-time monitoring and truly depict the health of project[11]. Metrics facilitate better business decisions, provide a challenge to the project teams, increase the satisfaction, etc[12]. Typical metrics should cover product/project attributes (size, quality, requirements, burn down, effort estimation, percentage of test cases automated, availability of tools and infrastructure, user stories traceability, test case prioritization and their allocation etc), process attributes (cycle time, build status, average velocity, release frequency, test efficiency patterns, etc), resource attributes (allocation, task completion status, performance, business value delivered, etc) [12][13][14]. Metrics should also cover non-functional aspects like project management (Sprint duration, estimate confidence, risk management, team, etc)[15]. It is a good practice to define key KPIs like frequency of deployment, speed of deployment, speed and frequency of build verification, deployment success rate, incident/defect volumes, requirements coverage ratio, feature usage, mean time to restore service, security test pass rate, etc along with core metrics [6]. The success of CT lies in how well the Test First process executed [16]. Test Case generation and corresponding test case related metrics using machine learning techniques play a major role in CT success [17-18].

3. CONTINUOUS TESTING METRICS AND IMPLEMENTATION

3.1 Conceptual design of Continuous Testing Metrics (Part 1-Basic Project Details)

Metrics and key performance indicators present meaningful information flow. Information flow takes place between customer desk, development environment, integration environment, pre-production/production environment, defect tracking system, version management system, project management tools and other organization-specific dashboards. DevOps Continuous Testing demands the design of metrics/measures which presents the real-time status of the project. These metrics may not be mere numbers but measure the un-measurable attributes like trust, confidence, culture strength & cohesion within the teams, etc. Few are difficult to measure and present but they are needed for successful completion of the project.

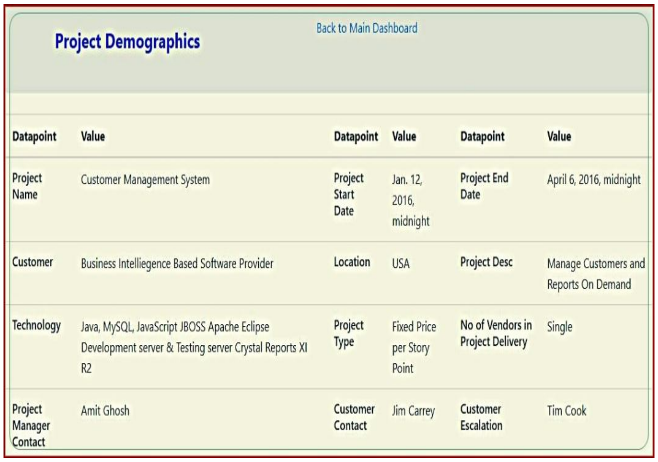
In this section, basic project demographic details are presented. We used Django, a Python-based open-source web framework for implementation of these metrics. Django follows the MVT (model-view-template) architectural pattern. As showed in Figure1, Dashboard-Part 1 presents basic demographic information like Project Name, Project Start Date, Project End Date, Total Number of Sprints Planned, Number of Sprints Completed, Current Sprint Number, No of Developers, No of Testers, No of Operation Team members, No of User Stories, Expected Delivery (Delivery Date Uncertainty Window) and Burn down Chart.


Real-Time Integrated (Dev-Test-Ops) Application Health Visual Analytics Dashboard

Metric	Value	Metric	Value	Metric	Value
Project Name	Customer Management System	Project Start Date	Jan. 12, 2016, midnight	Project End Date	April 6, 2016, midnight
Total No of Sprints Planned	6	No of Sprints Completed	1	Current Sprint #	2
No of Dev	2	No of Testers	2	No of Ops Engineers	1
No of User Stories	27	Expected Delivery (in weeks) (Delivery Date Uncertainty Window)	19	Burndown Chart	Click

Figure 1: Dashboard- Part 1

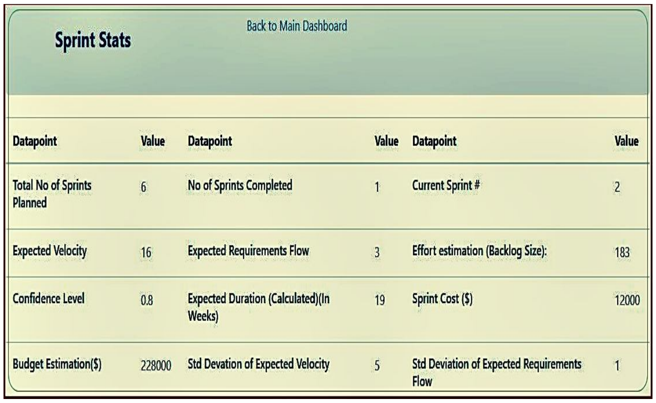
On click of "Project Name's Value" in Figure 1, Project Demographics page is displayed as showed in Figure 2. This page presents details like customer details, technology details, project location details, key project contacts, etc.


Project Demographics

Datapoint	Value	Datapoint	Value	Datapoint	Value
Project Name	Customer Management System	Project Start Date	Jan. 12, 2016, midnight	Project End Date	April 6, 2016, midnight
Customer	Business Intelliegence Based Software Provider	Location	USA	Project Desc	Manage Customers and Reports On Demand
Technology	Java, MySQL, JavaScript JBOSS Apache Eclipse Development server & Testing server Crystal Reports XI R2	Project Type	Fixed Price per Story Point	No of Vendors in Project Delivery	Single
Project Manager Contact	Amit Ghosh	Customer Contact	Jim Carrey	Customer Escalation	Tim Cook

Figure 2: Project Demographics

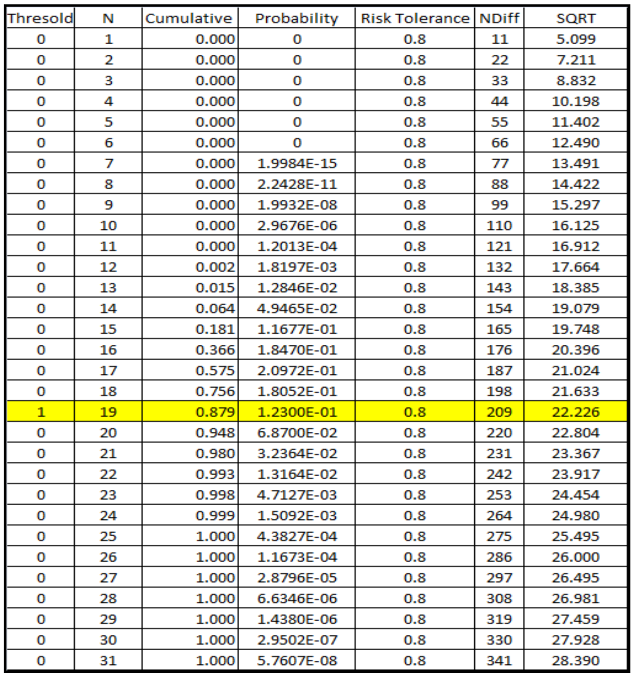
On click of "Total No of Sprints Planned Value" in Figure 1, Sprint Stats page is displayed as showed in Figure 3. This page presents Sprint related details like Total No of Sprints Planned, No of Sprints Completed, Current Sprint number, Expected Velocity, Expected Requirements Flow, Effort estimation (Backlog Size), Confidence Level, Expected Duration (Calculated)(In Weeks), Sprint Cost ($), Budget Estimation($), Std Deviation of Expected Velocity, Std Deviation of Expected Requirements Flow.



Datapoint	Value	Datapoint	Value	Datapoint	Value
Total No of Sprints Planned	6	No of Sprints Completed	1	Current Sprint #	2
Expected Velocity	16	Expected Requirements Flow	3	Effort estimation (Backlog Size):	183
Confidence Level	0.8	Expected Duration (Calculated)(In Weeks)	19	Sprint Cost ($)	12000
Budget Estimation($)	228000	Std Devation of Expected Velocity	5	Std Deviation of Expected Requirements Flow	1

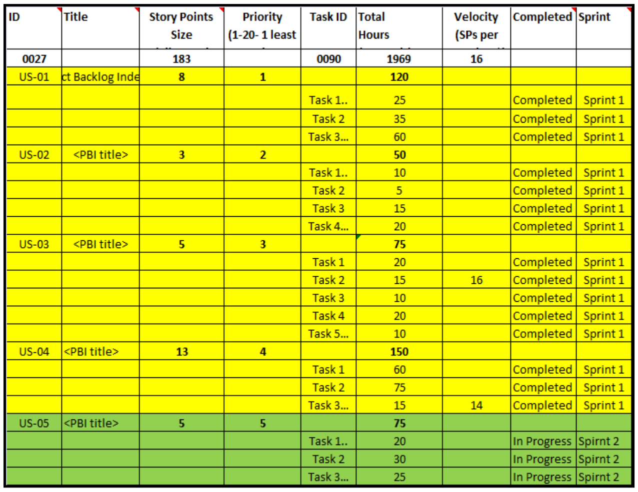
Figure 3: Sprint Stats

Expected Duration is calculated using the normal distribution curve as presented in Figure 4. This algorithm contains the Threshold week, Week Number, Cumulative Confidence number, Probability, Risk Tolerance, etc. In the given illustration, the cumulative confidence level stands at 0.879 during Week 19 which crossed the 0.8 threshold value. This number becomes the expected duration in weeks. Expected Velocity is calculated (Expected Velocity Calculator developed as an illustration) as the average of all completed sprints velocities as showed in Figure 5. Std Deviation of Expected Velocity and Std Deviation of Expected Requirements Flow are determined based on the previous history.



Thresold	N	Cumulative	Probability	Risk Tolerance	NDiff	SQRT
0	1	0.000	0	0.8	11	5.099
0	2	0.000	0	0.8	22	7.211
0	3	0.000	0	0.8	33	8.832
0	4	0.000	0	0.8	44	10.198
0	5	0.000	0	0.8	55	11.402
0	6	0.000	0	0.8	66	12.490
0	7	0.000	1.9984E-15	0.8	77	13.491
0	8	0.000	2.2428E-11	0.8	88	14.422
0	9	0.000	1.9932E-08	0.8	99	15.297
0	10	0.000	2.9676E-06	0.8	110	16.125
0	11	0.000	1.2013E-04	0.8	121	16.912
0	12	0.002	1.8197E-03	0.8	132	17.664
0	13	0.015	1.2846E-02	0.8	143	18.385
0	14	0.064	4.9465E-02	0.8	154	19.079
0	15	0.181	1.1677E-01	0.8	165	19.748
0	16	0.366	1.8470E-01	0.8	176	20.396
0	17	0.575	2.0972E-01	0.8	187	21.024
0	18	0.756	1.8052E-01	0.8	198	21.633
1	19	0.879	1.2300E-01	0.8	209	22.226
0	20	0.948	6.8700E-02	0.8	220	22.804
0	21	0.980	3.2364E-02	0.8	231	23.367
0	22	0.993	1.3164E-02	0.8	242	23.917
0	23	0.998	4.7127E-03	0.8	253	24.454
0	24	0.999	1.5092E-03	0.8	264	24.980
0	25	1.000	4.3827E-04	0.8	275	25.495
0	26	1.000	1.1673E-04	0.8	286	26.000
0	27	1.000	2.8796E-05	0.8	297	26.495
0	28	1.000	6.6346E-06	0.8	308	26.981
0	29	1.000	1.4380E-06	0.8	319	27.459
0	30	1.000	2.9502E-07	0.8	330	27.928
0	31	1.000	5.7607E-08	0.8	341	28.390

Figure 4: Normal Distribution Curve



ID	Title	Story Points Size	Priority (1-20- 1 least	Task ID	Total Hours	Velocity (SPs per	Completed	Sprint
0027		183		0090	1969	16		
US-01	ct Backlog Inde	8	1		120			
				Task 1..	25		Completed	Sprint 1
				Task 2	35		Completed	Sprint 1
				Task 3...	60		Completed	Sprint 1
US-02	<PBI title>	3	2		50			
				Task 1..	10		Completed	Sprint 1
				Task 2	5		Completed	Sprint 1
				Task 3	15		Completed	Sprint 1
				Task 4...	20		Completed	Sprint 1
US-03	<PBI title>	5	3		75			
				Task 1	20		Completed	Sprint 1
				Task 2	15	16	Completed	Sprint 1
				Task 3	10		Completed	Sprint 1
				Task 4	20		Completed	Sprint 1
				Task 5...	10		Completed	Sprint 1
US-04	<PBI title>	13	4		150			
				Task 1	60		Completed	Sprint 1
				Task 2	75		Completed	Sprint 1
				Task 3...	15	14	Completed	Sprint 1
US-05	<PBI title>	5	5		75			
				Task 1..	20		In Progress	Spirnt 2
				Task 2	30		In Progress	Spirnt 2
				Task 3...	25		In Progress	Spirnt 2

Figure 5. Expected Velocity Calculator

The delivery data uncertainty window is presented in Figure 6 which depicts the probability vs. cumulative vs. risk tolerance values.

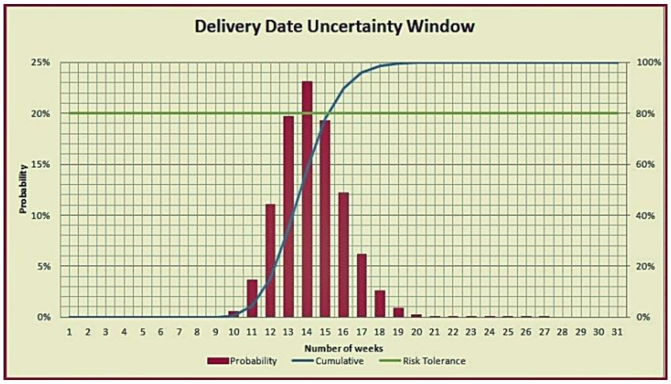


Figure 6: Delivery Date Uncertainty Window

On click of "Burn down Chart Value" in Figure1, Burn Down Chart is displayed as showed in Figure 7.

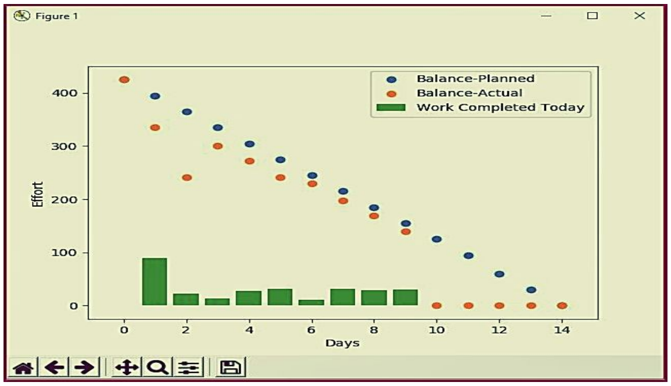
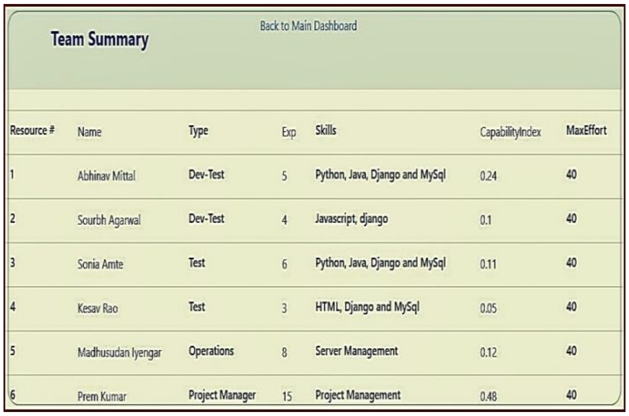


Figure 7: Burn Down Chart

On click of "No of Dev" or "No of Tester" or " No of Ops Engineers" value in Figure 1, Team Summary page is displayed as showed in Figure 8. This page presents Resource ID, Name, Type of Resource, Skills, Capability Index (calculated based on previous performance history in the organization) and Max Effort per Week.

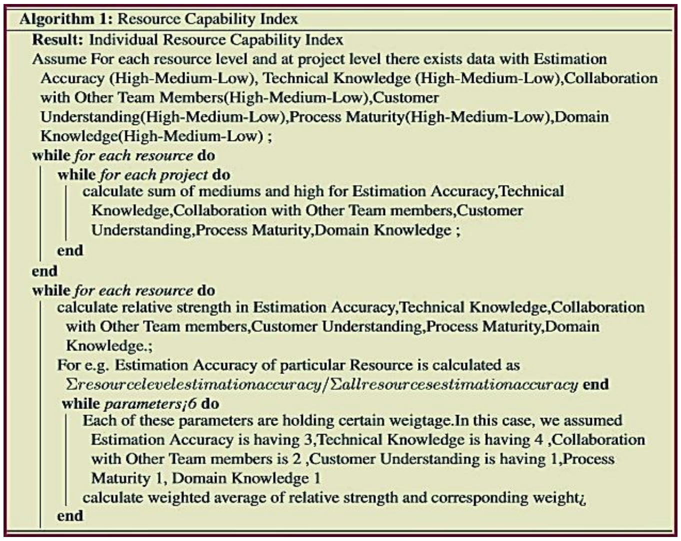


Team Summary

Resource #	Name	Type	Exp	Skills	CapabilityIndex	MaxEffort
1	Abhinav Mittal	Dev-Test	5	Python, Java, Django and MySql	0.24	40
2	Sourbh Agarwal	Dev-Test	4	Javascript, django	0.1	40
3	Sonia Amte	Test	6	Python, Java, Django and MySql	0.11	40
4	Kesav Rao	Test	3	HTML, Django and MySql	0.05	40
5	Madhusudan Iyengar	Operations	8	Server Management	0.12	40
6	Prem Kumar	Project Manager	15	Project Management	0.48	40

Figure 8: Team Summary

Capability Index is calculated using an algorithm which is presented in the Figure 9. The key fields to calculate are - Resource ID, Project ID, Estimation Accuracy (EA), Technical Knowledge (TK), Collaboration within the team(CT), Customer Understanding(CU), Process Maturity(PM), Domain Knowledge(DK). These fields take numerical values (3-High, 2-Medium,1-Low). The following sum values are calculated where field value >= 2 or 3 (Medium or high) - $\sum_{i=1}^{n} \mathbf{EAi}$, **where n = total number of records** $\sum_{i=1}^{n} \mathbf{TKi}$, $\sum_{i=1}^{n} \mathbf{CTi}$, $\sum_{i=1}^{n} \mathbf{CUi}$,$\sum_{i=1}^{n} \mathbf{PMi}$, $\sum_{i=1}^{n} \mathbf{DKi}$. A similar exercise is done at team member level where field value >= 2 or 3. The relative performance values at resource level (j= resource number) is presented as $\sum_{i=1,j=1}^{n} \mathbf{EAij}$ / $\sum_{i=1}^{n} \mathbf{EAi}$. Finally, the weighted average (sum product of effort estimated * weight) /sum of weights) is being calculated.

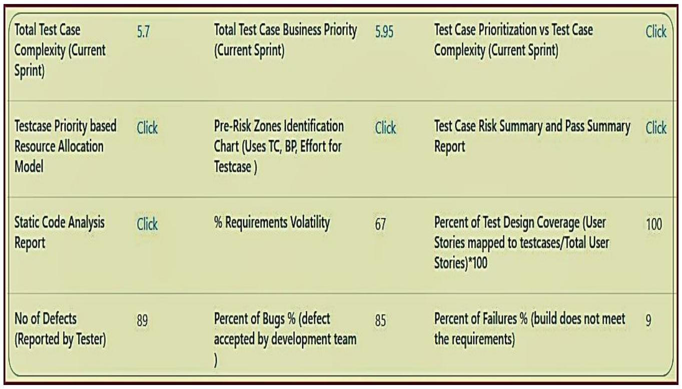


```
Algorithm 1: Resource Capability Index
  Result: Individual Resource Capability Index
  Assume For each resource level and at project level there exists data with Estimation
    Accuracy (High-Medium-Low), Technical Knowledge (High-Medium-Low),Collaboration
    with Other Team Members(High-Medium-Low),Customer
    Understanding(High-Medium-Low),Process Maturity(High-Medium-Low),Domain
    Knowledge(High-Medium-Low) ;
  while for each resource do
      while for each project do
          calculate sum of mediums and high for Estimation Accuracy,Technical
            Knowledge,Collaboration with Other Team members,Customer
            Understanding,Process Maturity,Domain Knowledge ;
      end
  end
  while for each resource do
      calculate relative strength in Estimation Accuracy,Technical Knowledge,Collaboration
        with Other Team members,Customer Understanding,Process Maturity,Domain
        Knowledge.;
      For e.g. Estimation Accuracy of particular Resource is calculated as
        $\Sigma resourcelevelestimationaccuracy/\Sigma allresourcesestimationaccuracy$ end
       while parameters¡6 do
          Each of these parameters are holding certain weigtage.In this case, we assumed
            Estimation Accuracy is having 3,Technical Knowledge is having 4 ,Collaboration
            with Other Team members is 2 ,Customer Understanding is having 1,Process
            Maturity 1, Domain Knowledge 1
          calculate weighted average of relative strength and corresponding weight¿
      end
```

Figure 9: Algorithm for Resource Level Capability Index

3.2 Conceptual design of Continuous Testing Metrics (Part 2-Test Analysis)

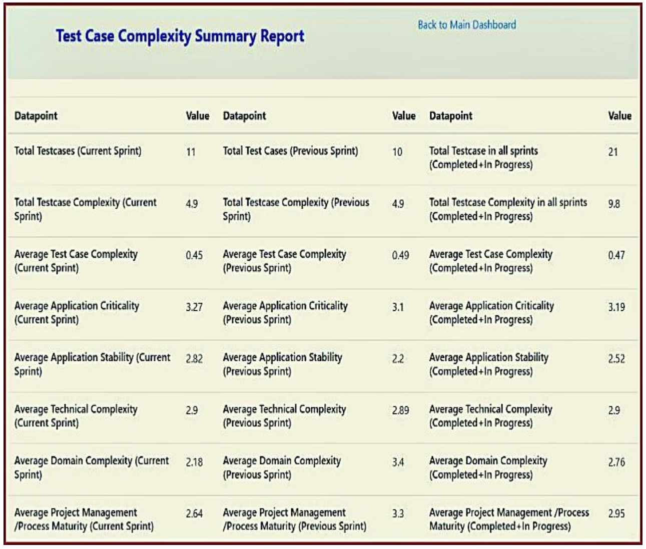
The second part of the metrics is related to Test Analysis. Test Cases play a major role in Test Analysis. They should be analyzed from the Test Case Complexity Perspective, Business Priority Perspective, and Test Case Risk Analysis perspective. Also, there are few important measures to be monitored like Static Code Analysis, % Requirements Volatility, Test Design Coverage, Number of Defects, Percentage of Bugs, Percentage of Failures, etc. These are processed and displayed as showed in Figure 10.



Total Test Case Complexity (Current Sprint)	5.7	Total Test Case Business Priority (Current Sprint)	5.95	Test Case Prioritization vs Test Case Complexity (Current Sprint)	Click
Testcase Priority based Resource Allocation Model	Click	Pre-Risk Zones Identification Chart (Uses TC, BP, Effort for Testcase)	Click	Test Case Risk Summary and Pass Summary Report	Click
Static Code Analysis Report	Click	% Requirements Volatility	67	Percent of Test Design Coverage (User Stories mapped to testcases/Total User Stories)*100	100
No of Defects (Reported by Tester)	89	Percent of Bugs % (defect accepted by development team)	85	Percent of Failures % (build does not meet the requirements)	9

Figure 10: Dashboard- Part 2

On click of Total Test Case Complexity in Figure 10, Test case technical complexity related metrics are displayed. Test Case complexity is analyzed from 4 different aspects- 1. Product / Application Criticality (AC) 2. Product / Application Stability (AS) Product / Application Technical Complexity (TC) 3. Product / Application Domain Complexity (DC) 4. Project Management / Process Maturity (PM) which is presented in Figure 11. These metrics are calculated for the current sprint, the previous sprint and completed and in-progress sprints perspective.

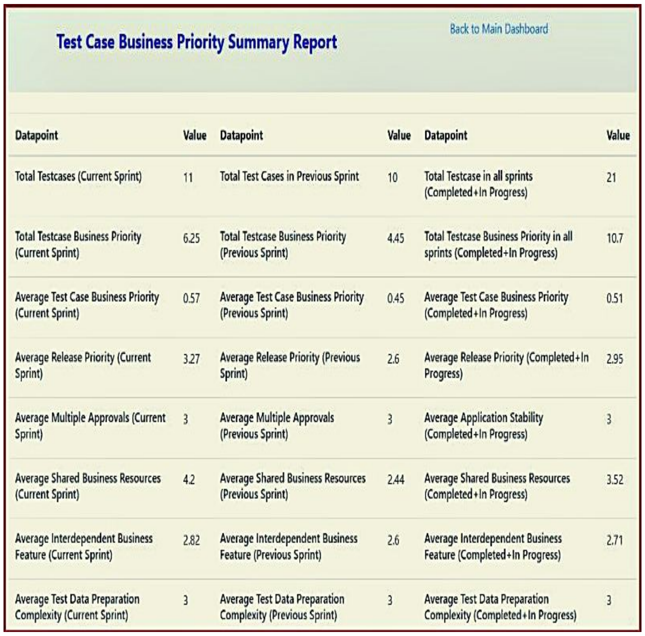


Test Case Complexity Summary Report

Datapoint	Value	Datapoint	Value	Datapoint	Value
Total Testcases (Current Sprint)	11	Total Test Cases (Previous Sprint)	10	Total Testcase in all sprints (Completed+In Progress)	21
Total Testcase Complexity (Current Sprint)	4.9	Total Testcase Complexity (Previous Sprint)	4.9	Total Testcase Complexity in all sprints (Completed+In Progress)	9.8
Average Test Case Complexity (Current Sprint)	0.45	Average Test Case Complexity (Previous Sprint)	0.49	Average Test Case Complexity (Completed+In Progress)	0.47
Average Application Criticality (Current Sprint)	3.27	Average Application Criticality (Previous Sprint)	3.1	Average Application Criticality (Completed+In Progress)	3.19
Average Application Stability (Current Sprint)	2.82	Average Application Stability (Previous Sprint)	2.2	Average Application Stability (Completed+In Progress)	2.52
Average Technical Complexity (Current Sprint)	2.9	Average Technical Complexity (Previous Sprint)	2.89	Average Technical Complexity (Completed+In Progress)	2.9
Average Domain Complexity (Current Sprint)	2.18	Average Domain Complexity (Previous Sprint)	3.4	Average Domain Complexity (Completed+In Progress)	2.76
Average Project Management /Process Maturity (Current Sprint)	2.64	Average Project Management /Process Maturity (Previous Sprint)	3.3	Average Project Management /Process Maturity (Completed+In Progress)	2.95

Figure 11: Test Case Technical Complexity Report

On click of Total Test Case Business Priority in Figure 10, Test case business priority related metrics are displayed. Test case business priority is calculated based on Release Priority, Multiple Approvals Needed, Shared Business Resources (Customer / Partners / Vendors), Interdependent Business, Test Data Preparation Complexity, etc. The metrics are displayed as showed in Figure 12.



Test Case Business Priority Summary Report

Back to Main Dashboard

Datapoint	Value	Datapoint	Value	Datapoint	Value
Total Testcases (Current Sprint)	11	Total Test Cases in Previous Sprint	10	Total Testcase in all sprints (Completed+In Progress)	21
Total Testcase Business Priority (Current Sprint)	6.25	Total Testcase Business Priority (Previous Sprint)	4.45	Total Testcase Business Priority in all sprints (Completed+In Progress)	10.7
Average Test Case Business Priority (Current Sprint)	0.57	Average Test Case Business Priority (Previous Sprint)	0.45	Average Test Case Business Priority (Completed+In Progress)	0.51
Average Release Priority (Current Sprint)	3.27	Average Release Priority (Previous Sprint)	2.6	Average Release Priority (Completed+In Progress)	2.95
Average Multiple Approvals (Current Sprint)	3	Average Multiple Approvals (Previous Sprint)	3	Average Application Stability (Completed+In Progress)	3
Average Shared Business Resources (Current Sprint)	4.2	Average Shared Business Resources (Previous Sprint)	2.44	Average Shared Business Resources (Completed+In Progress)	3.52
Average Interdependent Business Feature (Current Sprint)	2.82	Average Interdependent Business Feature (Previous Sprint)	2.6	Average Interdependent Business Feature (Completed+In Progress)	2.71
Average Test Data Preparation Complexity (Current Sprint)	3	Average Test Data Preparation Complexity (Previous Sprint)	3	Average Test Data Preparation Complexity (Completed+In Progress)	3

Figure 12: Test Case Business Priority Report

On click of Test Case Prioritization vs Test Case Complexity (Current Sprint) in Figure 10, Test Case Prioritization vs Test Case Complexity matrix is presented as showed in Figure 13. This matrix helps in finding complexity-priority zones in managing test cases. This process is helpful in delivery and allocation.

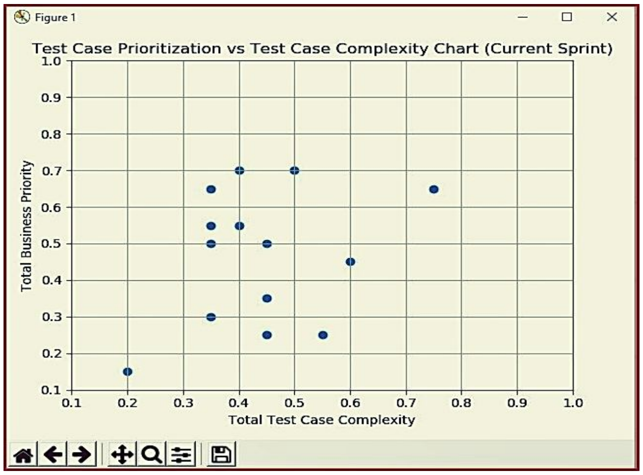
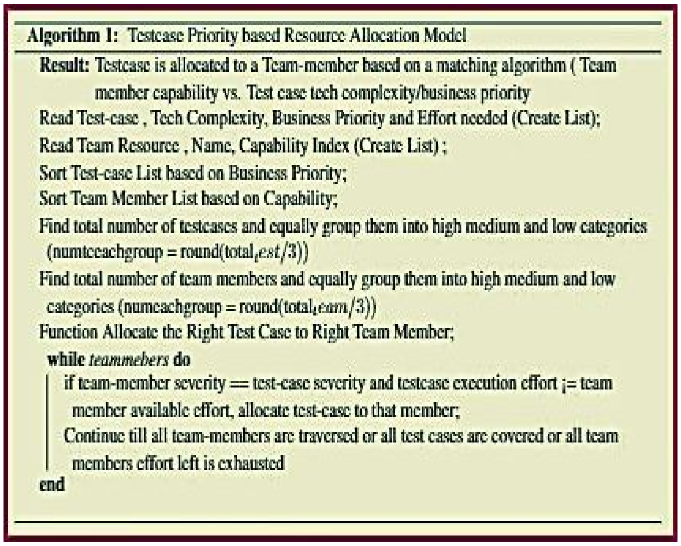


Figure 13: Test Case Prioritization Vs. Test Case Complexity

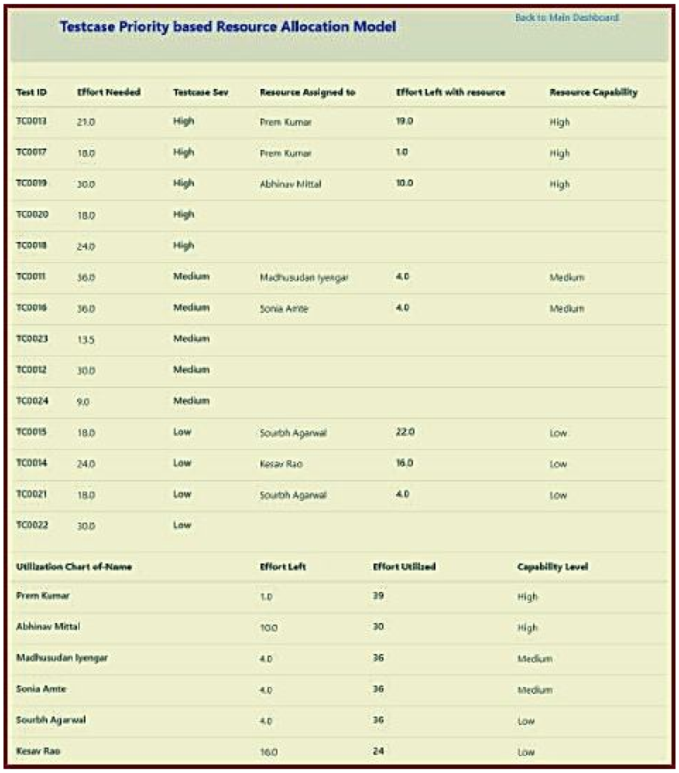
On click of Test case Priority based Resource Allocation Model of Figure 10, Test case Priority based Resource Allocation Model is displayed. This allocation is done through an algorithm which is explained in Figure 14.



```
Algorithm 1: Testcase Priority based Resource Allocation Model
  Result: Testcase is allocated to a Team-member based on a matching algorithm ( Team
          member capability vs. Test case tech complexity/business priority
  Read Test-case , Tech Complexity, Business Priority and Effort needed (Create List);
  Read Team Resource , Name, Capability Index (Create List) ;
  Sort Test-case List based on Business Priority;
  Sort Team Member List based on Capability;
  Find total number of testcases and equally group them into high medium and low categories
    (numteachgroup = round(total_est/3))
  Find total number of team members and equally group them into high medium and low
    categories (numeachgroup = round(total_eam/3))
  Function Allocate the Right Test Case to Right Team Member;
  while teammebers do
      if team-member severity == test-case severity and testcase execution effort ;= team
        member available effort, allocate test-case to that member;
      Continue till all team-members are traversed or all test cases are covered or all team
        members effort left is exhausted
  end
```

Figure 14: Test case Priority based Resource Allocation Model

Post-allocation, the allocated test cases, resource names, and their utilization and leftover effort details are automatically presented by the system which is shown in Figure 15.



Testcase Priority based Resource Allocation Model

Back to Main Dashboard

Test ID	Effort Needed	Testcase Sev	Resource Assigned to	Effort Left with resource	Resource Capability
TC0013	21.0	High	Prem Kumar	19.0	High
TC0017	18.0	High	Prem Kumar	1.0	High
TC0019	30.0	High	Abhinav Mittal	10.0	High
TC0020	18.0	High			
TC0018	24.0	High			
TC0011	36.0	Medium	Madhusudan Iyengar	4.0	Medium
TC0016	36.0	Medium	Sonia Amte	4.0	Medium
TC0023	13.5	Medium			
TC0012	30.0	Medium			
TC0024	9.0	Medium			
TC0015	18.0	Low	Sourbh Agarwal	22.0	Low
TC0014	24.0	Low	Kesav Rao	16.0	Low
TC0021	18.0	Low	Sourbh Agarwal	4.0	Low
TC0022	30.0	Low			

Utilization Chart of-Name	Effort Left	Effort Utilized	Capability Level
Prem Kumar	1.0	39	High
Abhinav Mittal	10.0	30	High
Madhusudan Iyengar	4.0	36	Medium
Sonia Amte	4.0	36	Medium
Sourbh Agarwal	4.0	36	Low
Kesav Rao	16.0	24	Low

Figure 15: Test case Priority based Resource Allocation Summary

On click of Pre-Risk Zones Identification Chart (Uses TC, BP, Effort for Test case) in Figure 10, various test cases technical complexities vs. Business Priority vs. execution effort details is presented as showed in Figure 16. This summary helps to identify the Pre-Risk zones and to deploy resources accordingly.

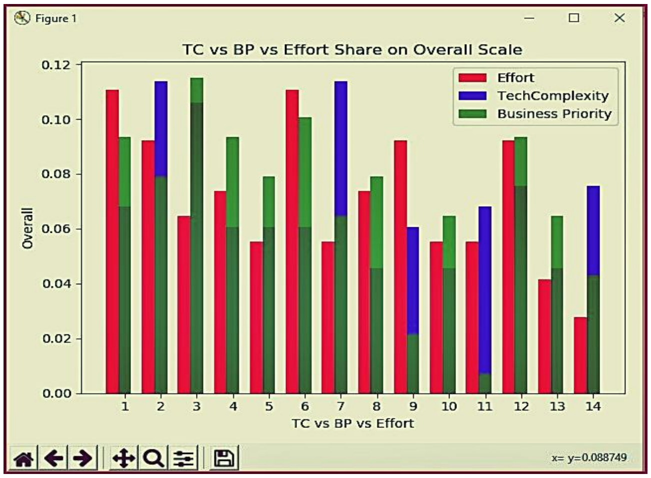
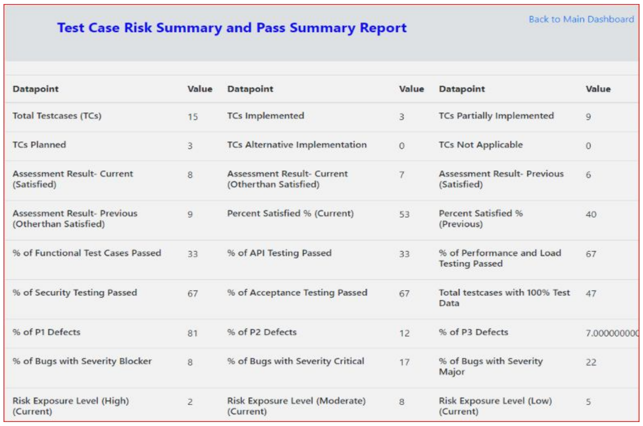


Figure 16: Pre-Risk Zones Identification Chart

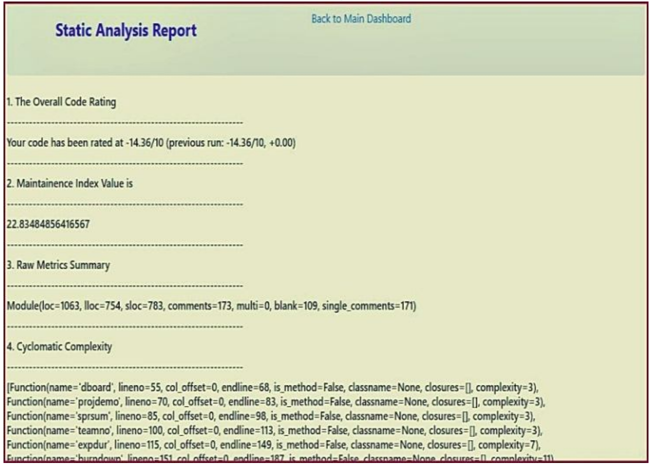
On click of Test Case Risk Summary and Pass Summary Report in Figure 10, Test Case Risk Summary and Pass Summary Report is presented as showed in Figure 17. It contains Total Test cases (TCs), TCs Implemented, TCs Partially Implemented, TCs Planned, TCs Alternative Implementation, TCs Not Applicable, Assessment Result- Current (Satisfied),Assessment Result- Current (Other than Satisfied), Assessment Result- Previous (Satisfied),Assessment Result- Previous (Other than Satisfied), Percent Satisfied % (Current),Percent Satisfied % (Previous),% of Functional Test Cases Passed, % of API Testing Passed, % of Performance and Load Testing Passed, % of Security Testing Passed, % of Acceptance Testing Passed, Total testcases with 100% Test Data, % of P1 Defects, % of P2 Defects, % of P3 Defects, % of Bugs with Severity Blocker , % of Bugs with Severity Critical, % of Bugs with Severity Major , Risk Exposure Level (High) (Current), Risk Exposure Level (Moderate) (Current), Risk Exposure Level (Low) (Current), Risk Exposure Level (High %) (Current), Risk Exposure Level % (Moderate), Risk Exposure Level (Low) % (Current), Risk Exposure Level (High) (Previous),Risk Exposure Level (Moderate) (Previous),Risk Exposure Level (Low) (Previous),Risk Exposure Level (High) % (Previous),Risk Exposure Level (Moderate) % (Previous), Risk Exposure Level (Low) % (Previous) etc. The same is presented in Figure 17.


Test Case Risk Summary and Pass Summary Report

Datapoint	Value	Datapoint	Value	Datapoint	Value
Total Testcases (TCs)	15	TCs Implemented	3	TCs Partially Implemented	9
TCs Planned	3	TCs Alternative Implementation	0	TCs Not Applicable	0
Assessment Result- Current (Satisfied)	8	Assessment Result- Current (Otherthan Satisfied)	7	Assessment Result- Previous (Satisfied)	6
Assessment Result- Previous (Otherthan Satisfied)	9	Percent Satisfied % (Current)	53	Percent Satisfied % (Previous)	40
% of Functional Test Cases Passed	33	% of API Testing Passed	33	% of Performance and Load Testing Passed	67
% of Security Testing Passed	67	% of Acceptance Testing Passed	67	Total testcases with 100% Test Data	47
% of P1 Defects	81	% of P2 Defects	12	% of P3 Defects	7.000000000
% of Bugs with Severity Blocker	8	% of Bugs with Severity Critical	17	% of Bugs with Severity Major	22
Risk Exposure Level (High) (Current)	2	Risk Exposure Level (Moderate) (Current)	8	Risk Exposure Level (Low) (Current)	5

Figure 17: Test Case Risk Summary and Pass Summary Report

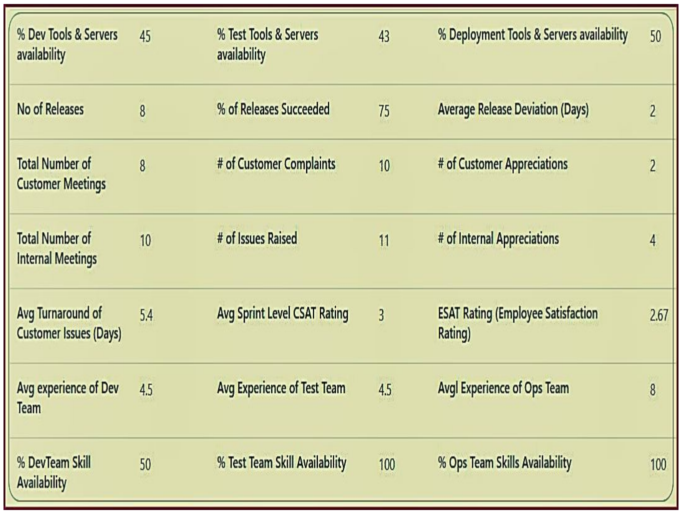
On click of Static Code Analysis Report in Figure 10, it reads the entire code base connected to business logic and presents the metrics like Overall Code Rating, Maintenance Index Value, Raw Metrics Summary (illustration -loc=1063, lloc=754, sloc=783, comments=173, multi=0, blank=109, single comments=171), Cyclomatic Complexity, Halstead's Software Metrics (Halstead Program Length, Halstead Vocabulary, Program Volume, Potential Minimum Volume, Program Level, Program Difficulty, Programming Effort, Language Level, Intelligence Content, Programming Time), Conventions, Warnings, Refactoring details, etc.


Static Analysis Report

```
1. The Overall Code Rating
---------------------------------------------------------------
Your code has been rated at -14.36/10 (previous run: -14.36/10, +0.00)
---------------------------------------------------------------
2. Maintainence Index Value is
---------------------------------------------------------------
22.83484856416567
---------------------------------------------------------------
3. Raw Metrics Summary
---------------------------------------------------------------
Module(loc=1063, lloc=754, sloc=783, comments=173, multi=0, blank=109, single_comments=171)
---------------------------------------------------------------
4. Cyclomatic Complexity
---------------------------------------------------------------
[Function(name='dboard', lineno=55, col_offset=0, endline=68, is_method=False, classname=None, closures=[], complexity=3),
Function(name='projdemo', lineno=70, col_offset=0, endline=83, is_method=False, classname=None, closures=[], complexity=3),
Function(name='sprsum', lineno=85, col_offset=0, endline=98, is_method=False, classname=None, closures=[], complexity=3),
Function(name='teamno', lineno=100, col_offset=0, endline=113, is_method=False, classname=None, closures=[], complexity=3),
Function(name='expdur', lineno=115, col_offset=0, endline=149, is_method=False, classname=None, closures=[], complexity=7),
Function(name='burndown', lineno=151, col_offset=0, endline=187, is_method=False, classname=None, closures=[], complexity=11)
```

Figure 17: Static Code Analysis Report

The final set of metrics are general project execution related. They are percentage of Dev Tools & Servers availability, No of Releases, Total Number of Customer Meetings, Total Number of Internal Meetings, Average Turnaround of Customer Issues (Days), Average experience of Dev Team, Percentage of DevTeam Skill Availability, Percentage Test Tools & Servers availability, Percentage of Releases Succeeded, No of Customer Complaints, No of Issues Raised, Average Sprint Level CSAT Rating, Average Experience of Test Team, Percentage of Test Team Skill Availability, etc. These metrics are calculated from the database and presented as showed in the Figure 18.



% Dev Tools & Servers availability	45	% Test Tools & Servers availability	43	% Deployment Tools & Servers availability	50
No of Releases	8	% of Releases Succeeded	75	Average Release Deviation (Days)	2
Total Number of Customer Meetings	8	# of Customer Complaints	10	# of Customer Appreciations	2
Total Number of Internal Meetings	10	# of Issues Raised	11	# of Internal Appreciations	4
Avg Turnaround of Customer Issues (Days)	5.4	Avg Sprint Level CSAT Rating	3	ESAT Rating (Employee Satisfaction Rating)	2.67
Avg experience of Dev Team	4.5	Avg Experience of Test Team	4.5	Avgl Experience of Ops Team	8
% DevTeam Skill Availability	50	% Test Team Skill Availability	100	% Ops Team Skills Availability	100

Figure 18: Dashboard- Part 3

4.THREATS TO VALIDITY

We attempted to simulate real-time projects execution parameters and implemented them using Django-Python Web Framework. These metrics can be further fine-tuned while implementing real-time projects. This paper covers exhaustive list of metrics for in the context of DevOps continuous testing. However, project managers need to select relevant metrics suiting to their project requirements and customize real-time dash board. We created datasets using Excel and implemented this dashboard. However, this can be further improvised by introducing database management software tools. Authors and Affiliations

5.CONCLUSION

Continuous Testing (CT) promotes automated tests as part of software delivery so that feedback on functional, technical and business risks is real-time and continuous. Project Communication, Technology adoption, Team Collaboration, Tools and Processes, etc are critical factors driving CT process. The probability of project success is high when metrics are applied systematically, methodologically and results are published real-time. CT project health requires the design of progressive metrics/measures which brings-out the adaptive project culture. It improves the collaboration between all project stakeholders. It requires a well-designed system. CT Metrics becomes the tone of organization culture and abilities for effective testing in DevOps phenomena.

REFERENCES

[1] O. Aktunc. **Entropy Metrics for Agile Development Processes,** in 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 7–8, 2012. https://doi.org/10.1109/ISSREW.2012.36

[2] J. Angara, S. Gutta, S. Prasad. **DevOps with Continuous Testing Architecture and Its Metrics Model**. In: Sa P., Bakshi S., Hatzilygeroudis I., Sahoo M. (eds) Recent Findings in Intelligent Computing Techniques. Advances in Intelligent Systems and Computing, vol 709. Springer, Singapore, 2018.

[3] W Ariola. **DevOps: Are You Pushing Bugs to Your Clients Faster?** Thirty-Third Annual Pacific Northwest Software Quality Conference, World Trade Center Portland Portland, Oregon, October 12-14 ,2015.

[4] W. M. Farid and F. J. Mitropoulos. **NORPLAN: Non-functional Requirements Planning for agile processes,** in 2013 Proceedings of IEEE Southeastcon, pp. 1–8, 2013. https://doi.org/10.1109/SECON.2013.6567463

[5] D. Hartmann and R. Dymond. **Appropriate agile measurement: using metrics and diagnostics to deliver business value**, in Agile Conference, p. 6 pp.– 134, 2006.

[6] InfoStretch, http://www.qmetry.com/casestudy-stanford/

[7] N. Kerzazi, and F. Khomh. **Factors Impacting Rapid Releases: An Industrial Case Study**, in Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, New York, NY, USA, pp. 61:1–61:8, 2014. https://doi.org/10.1145/2652524.2652589

[8] A. Kushwaha, S. K. Verma, and C. Sharma. **Analysis of the Concerns Associated with the Rapid Release Cycle**, Int. J. Comput. Appl., vol. 52, no. 12, 2012.

[9] Measuring DevOps Success, https://www.microfocus.com/media/white-paper/measuring_devops_success_wp.pdf, accessed on 03-Apr-2019

[10] S. Misra and M. Omorodion. **Survey on Agile Metrics and Their Inter-relationship with Other Traditional Development Metrics**, SIGSOFT Softw Eng Notes, vol. 36, no. 6, pp. 1–3, Nov. 2011. https://doi.org/10.1145/2047414.2047430

[11] D. Saff, and. M. Ernst, **An experimental evaluation of continuous testing during development**. In Proceedings of the 2004 International Symposium on Software Testing and Analysis, pages 76–85, Boston, MA, USA, July 2004.

[12] M. Schur, A. Roth, A. Zeller. **Mining Workflow Models from Web Applications**, IEEE Transactions on Software Engineering 41(12):1-1(2015)

[13] Spirent. **A Solution Blueprint for DevOps**. http://www.spirent.com/Assets/WP/WP_A_Solution_Blueprint_for_DevOps

[14] Tricentis 2019. **What is Continuous Testing?,** https://www.tricentis.com/products/what-is-continuous-testing/, accessed on 03-Apr-2019

[15] Wayne A. and Cynthia D (2016). **Continuous Testing for IT Leaders**, https://alm.parasoft.com/ continuoustestingbook. accessed on 14-06-2016

[16] N. Yahya, S.S.Maidin, A.B.Soomro.**Factors Influence Novice Programmers toward Test First Approach**, International Journal of Advanced Trends in Computer Science and Engineering, Vol 8, No.4, July-Aug 2019. https://doi.org/10.30534/ijatcse/2019/39842019

[17] L. Rajamanickam, N. A. Mat Saat, and S. N. Daud, **Software Testing: The Generation Tools**, International Journal of Advanced Trends in Computer Science and Engineering, vol. 8, pp. 231-234, 2018. https://doi.org/10.30534/ijatcse/2019/20822019

[18] M.H.Molawade, S.D. Joshi, **Software reliability prediction using Knowledge Engineering approach**, International Journal of Advanced Trends in Computer Science and Engineering,Vol 8, No 6, Nov-Dec 2019 https://doi.org/10.30534/ijatcse/2019/14862019