NTT DATA

CRESTA_Software Test Acceleration Robotic Engine Requirements

Global R&D initiative 2016

Doc V 1.3 | October, 2016

REVISION HISTORY AND APPROVAL

Version	Effective Date	Brief Description of Change	Affected Section(s)	Prepared By	Reviewed By	Approved By
0.1	22-Jul-16	Initial Version		Sameer Bagade	Amod Bhalerao	
0.2	29-Jul-16	Initial Version		Sameer Bagade	Amod Bhalerao	
0.3	1-Aug-16	Initial Version		Sameer Bagade	Amod Bhalerao	
0.4	03-Aug-16	Review comments by Amod		Sameer Bagade	Hirofumi Ota & Imai Katsutoshi	
0.5	08-Aug-16	Review comments given by Ota & Imai	5. Architecture 3.3 Benefits of CRESTA 7. Roles & Users	Sameer Bagade	Amod Bhalerao	
0.6	12-Aug-16	Review comments given by Amod	6. Requirements 14. Appendix	Sameer Bagade	Hirofumi Ota & Imai Katsutoshi	
0.7	16-Aug-16	Review comments given by Ota & Imai	4. Overview 5. Architecture 14. Appendix	Sameer Bagade	Amod Bhalerao	
0.8	16-Aug-16	Review comments given by Amod	4. Overview 5. Architecture 14. Appendix	Sameer Bagade	Amod Bhalerao	
0.9	26-Aug-16	New sections added	3.4 Actor list added Section 6.6 & 6.7 added Section 10 Message	Ranjeet Mohanty	Sameer Bagade	

Version	Effective Date	Brief Description of Change	Affected Section(s)	Prepared By	Reviewed By	Approved By
			Matrix added Row no. 16 in Defect Metrics.xlsx was added			
1.0	30-Aug-16	Review comments given by Sameer	7. Users & Roles 6. Requirements	Ranjeet Mohanty	Amod Bhalerao	
1.1	6-Sep-16	ETL Requirements Metrics Parameters	6. Requirements 15. Appendix	Sameer Bagade		
1.1	8-Sep-16	Test Metrics Parameters Wireframes	15. Wireframes 16. Appendix	Ranjeet mohanty	Sameer Bagade	
1.2	13-Sep-16	Added Use Case. Added following requirements 1. Trend Parameter 2. Prediction Model	6.1 Generic Requirements	Sameer Bagade		
1.3	10-Nov-16	Added use cases Added defect metrics.	5. Use cases 17.1 Defect Measures & Metrics			

Table of Contents

1 Glossary

Sr. No.	Short Form	Long Form
1	AI	Artificial Intelligence
2	ALM	Application Life cycle Management tool
3	API	Application program interface
4	CRESTA	Comprehensive Robotic Engine for Software Test Acceleration
5	DD	Defect Density
6	DL	Defect Leakage
7	DRE	Defect Removal Efficiency
8	JIRA	A proprietary bug tracking, issue tracking, and project management tool
9	LCL	Lower Control Limit LCL = mean value - standard deviation
10	ML	Machine Learning
11	NA	Not Applicable
12	Shift Left	Finding defects earlier in the lifecycle to save the cost of fixing defects
13	SOW	Statement of Work
14	TC	Test Cases
15	TCD	Test Case Density
16	UCL	Upper Control Limit UCL = mean value + standard deviation

2 Purpose

The purpose of this document is to capture and describe the requirements of **CRESTA** - A web based solution to predict the future state and quality of the software using ML (Machine Learning) and AI (Artificial Intelligence).

3 CRESTA – An Overview

3.1 Introduction

CRESTA is a web based solution for analyzing historical test artifacts such as test cases, defects and defect metrics of the project. It integrates with project and defect management tools like Redmine, ALM, JIRA, Bugzilla to collect project defects data, metrics and other artifacts. It then leverages intelligent robotic automation technologies such as ML, AI and Text analytics for predictive defects analysis and test optimization. It presents the results to the user in highly comprehendible dashboards, graphs, charts and tables.

3.2 System Overview Diagram

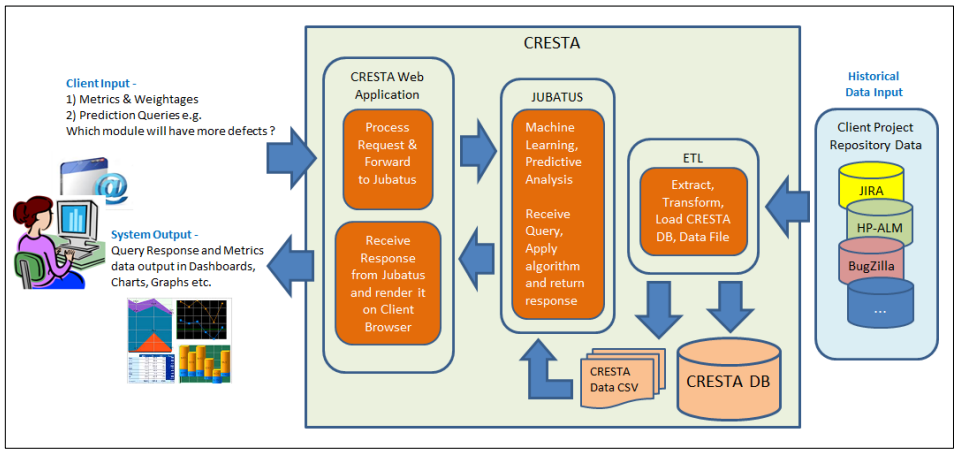

Figure 4.1 System Overview Diagram

3.3 System Description

CRESTA has 3 main components as shown in the diagram above.

1. Web Application –
   This is the user interface of CRESTA.
   1.1. Web application allows user to define various metrics for the project and their weightages.
   1.2. It receives various queries from user e.g. "Which module will have more defects in future?", "Are we under-testing some part of the system?" and so on.
   1.3. Web application passes these query requests to next component JUBATUS which processes these requests.
   1.4. It renders the responses from JUBATUS in the form of various dashboards, graphs, and charts.

2. ETL –
   This component extracts Historical defects data from repositories like Redmine, JIRA, QC etc. (These are just examples. For actual repository list which CRESTA supports refer to ETL section later in this document). ETL collates the gathered data and loads it in CRESTA Database and also to CSV files for JUBATUS component to read.

3. JUBATUS –
   This is the core component or brain of CRESTA which carries out all predictions. It applies ML and AI algorithms to user defined metrics and the collated historical data by ETL component to predict future values of various metrics and quality trend e.g.
   3.1. Find out mean value and standard deviation
   3.2. Calculate and predict UCL & LCL
   3.3. Plot normal distribution and so on…

   Example algorithms -
   Machine Learning - Vector Machines (SVM) for regression and classification
   Naive Bayes - For classification
   k-Nearest Neighbours (KNN) - For regression and classification
   Cluster Management - For Joining (Tree Clustering)
   K-Means Clustering and EM (Expectation Maximization) Clustering

3.4 *Salient Features*

- Predictive analysis on historical data to provide insights on quality gaps, defect targets/trends
- Dashboard & reporting to provide visibility to stakeholders based on historical and release data, various metrics and predictions.
- Visual coverage maps to intuitively uncover coverage issues prior to release
- Analyze test cases in real-time or offline and identify redundancy and reusability
- Analyze test execution logs and identify untested areas of software proactively
- Data aggregation from commonly used industry standard tools

3.5 *CRESTA User Roles (Actors)*

Role Abbreviation	Role Name	Description
TL	Test Lead	Carry out predictions on various metrics based on various prediction questions.
PM	Project Manager	Carry out predictions on various metrics same as TL but has more rights to modify the controlling parameters e.g. UCL, LCL.
DM	Delivery Manager	Comparative analysis of predictions of multiple projects
CA	CRESTA Admin	1. Create/Edit/Delete User Roles. 2. Roles and Function Access Rights management. 3. User to Project access rights management.
GU	Guest User	View prediction data in read-only mode.

Note: The detail Roles and Rights assignment i.e. Role wise Function wise Read/Modify access rights matrix, refer to High Level Design.

4 Generic Requirements

4.1. CRESTA should provide web based user interface and reporting dashboard
4.2. CRESTA should integrate with Redmine
4.3. This supervised machine learning will predict defect metrics. These defect matrices will help in making release decisions
4.4. CRESTA should read historical defects data
4.5. CRESTA should read\calculate metrics
4.6. CRESTA should predict UCL & LCL for each metric
4.7. CRESTA should predict future trend of the metrics
4.8. CRESTA should use future trend of these metrics & weightages to predict quality of product
4.9.

Refer Appendix section for Metrics information

5 Use Cases

For business flows refer CRESTA_Wireframe_Field Validation.xlsx

No.	Role	Question	Use Case Name
1	Test Lead	What will be the defect density in future?	Predict defect metrics
1A	Test Lead	What will be the defect acceptance rate in future?	Predict defect metrics
1B	Test Lead	What will be the defect deferral rate in future?	Predict defect metrics
1C	Test Lead	What will be the overall defects count in future?	Predicted All Defects vs. Actual All Defects
1D	Test Lead	What will be the functional defects count in future?	Predicted Functional Defects vs. Actual Functional Defects.
2	Test Manager	How to manage Regression testing within limited resources?	Test Optimization
3	Project Manager	Which modules will have difficulties during and after release?	Predict defective modules

Refer CRESTA_Use Cases_rev01.pptx and CRESTA_Usecase_Specification.docx for detailed description.
SVN Path - \03_SDLC\01_Requirements Development\Use Cases

6 Users and Roles

Commented [JR1]: We can create tabular roles and rights matrix where we can map roles to the functions with Read/Write access.
Make use of function list defined above during mapping.

Role	Metrics Definition	UCL & LCL	Use Case Name
Test Lead	View	Modify	Predict defect metrics
Project Manager	Modify	Modify, Approve	Predict defective modules
Delivery Manager	Modify	Modify, Approve	Project comparative analysis
CRESTA Admin	Delete	NA	Project setup, User setup
Guest User	View	View	Ready only access to all pages

Note - Guest User (GU) should have View\Ready only access to all pages

7.1. CRESTA should provide interface to setup users & roles.
7.2. CRESTA should provide four static roles(refer section 3.4 for actors list):
  7.2.1. Project Manager(PM)
  7.2.2. Test Lead(TL)
  7.2.3. CRESTA Admin(CA)
  7.2.4. Guest User(GU)
7.3. These 4 roles should be 'READ ONLY'. The Permissions of those CANNOT be modified (Edit/Remove) even by CRESTA Admin
7.4. Project Manager
  7.4.1. Project manager should have permissions to define metrics for a project
  7.4.2. Project manager should have permissions to configure UCL & LCL for various metrics
  7.4.3. Project manager should have permissions to monitor metrics & predictions given by CRESTA
7.5. Test Lead
  7.5.1. Test lead should have permissions to configure UCL & LCL for various metrics
  7.5.2. Test lead should have permissions to monitor metrics & predictions given by CRESTA
7.6. CRESTA Admin
  7.6.1. Only one user should be assigned to CRESTA Admin role
  7.6.2. CRESTA admin should have permissions to create new project
  7.6.3. CRESTA admin should have permissions to create new user for the project
  7.6.4. CRESTA admin should have permissions to assign project to the user
  7.6.5. CRESTA admin should have permissions to assign predefined role to the user except CRESTA Admin

7.6.6. CRESTA admin should have permissions to modify project
7.6.7. CRESTA admin should have permissions to modify user roles
7.6.8. CRESTA admin should have permissions to delete user
7.6.9. CRESTA admin should have permissions to delete project
7.6.10. CRESTA admin should have permissions to reset password of all the users except CRESTA Admin

7.7. Guest User

7.7.1. Guest user should have Read Only permissions to configuration page & reports dashboard

7 Business Rules

Refer CRESTA_CommonBusinessRules.xlsx

8 Functional Requirements

8.1 Login Page

8.2 Metrics Page - Selection of Metrics and Weightages

6.4.1. CRESTA should provide interface to select Defect and Test case metrics
6.4.2. CRESTA should predict & populate UCL value using historical data
6.4.3. CRESTA should predict & populate LCL values using historical data
6.4.4. CRESTA should provide interface to view CRESTA predicted UCL & LCL and define custom UCL & LCL for selected metric
6.4.5. Interface to define UCL & LCL should follow interface for metrics
6.4.6. Upper Control Limit (UCL) should be floating point numbers
6.4.7. Lower Control Limit (LCL) should be floating point numbers
6.4.8. Custom UCL & LCL should be non-mandatory fields
6.4.9. If custom UCL & LCL are kept blank, they should not be used for prediction
6.4.10. Upper Control Limit (UCL) should be definable at module, sub module & page\screen levels.
6.4.11. Lower Control Limit (LCL) should be definable at module, sub module & page\screen levels

8.3 Prediction Summary Page

8.4 Prediction Details Page

9 ETL Requirements

9.1 Structured Data

To be elaborated.

9.2 Unstructured Data

CRESTA should read test execution logs generated in excel format.
CRESTA should read system logs generated in text format. System logs are generated by application under test.

10 Supporting OS

CRESTA should be developed as a web based solution to support all operating systems.

11 Browser Support

CRESTA should support Internet Explorer 11.0

12 Assumptions

1. Historical defects data follow Normal distribution pattern.
2.

To be updated based on findings from PoC.

13 Non-functional Requirements

To be updated based on findings from PoC.

14 Out of Scope

1. Accessibility of CRESTA on mobile phones is out of scope for current phase

15 References

1. CRESTA_Wireframe_Field Validation.xlsx
2. CRESTA_Use Cases_rev01.pptx
3. CRESTA_Usecase_Specification.docx
4. CRESTA_CommonBusinessRules.xlsx
5. Defect_Metrics_Parameters.xlsx
6. TestCase_Metrics_Parameters.xlsx

16 Wireframes:

Refer CRESTA_Wireframe_Field Validation.xlsx
SVN path - 03_SDLC\01_Requirements Development\Wireframe & Field Validation Documents

17 Appendix

17.1 ***Defect Measures & Metrics***

Sr. No.	Defect Metric	Formula
1	Defect Density (DD)	No. of Defects identified / size Number of defects identified per requirement OR Number of Defects identified per 1000 lines of code OR Number of defects identified per module
2	Defect Leakage (DL)	DL = (No. of Defects found in UAT / No. of Defects found in QA testing) * 100 [How many defects are missed / slipped during the QA testing.]
3	Defect Rejection Rate (DRR)	DRR = [No. of QA Testing Defects Rejected by the Development Team] / [No. of Testing Defects Raised by the Testing Team] * 100
4	Mean Time to Bug	Average time to discover new bug
5	Defects by Severity	% Critical Defects = (No. of Critical Defects identified / Total no. of Defects identified) * 100 % High Defects = (No. of High Defects identified / Total no. of Defects identified) * 100 % Medium Defects = (No. of Medium Defects identified / Total no. of Defects identified) * 100 % Low Defects = (No. of Low Defects identified / Total no. of Defects identified) * 100
6	Defect Removal Efficiency (DRE)	DRE = (No. of Defects found during QA testing / (No. of Defects found during QA testing +No. of Defects found by End user)) * 100
7	Defect Acceptance Rate (DAR)	DAR = (No. of defects accepted by development team / Total No. of defects found by QA testing) * 100
8	Defect Deferral Rate (DDR)	DDR = (No. of defects deferred / No of defects accepted) * 100

17.2 ***Test Measures & Metrics***

Sr. No.	Test Case Metric	Formula
1	Automation ROI	ROI = (Cost of manual testing - Cost of automated testing)/Investment on automation
2	Test Case Density	Test Case Density = Number of Test Case/KLOC (kilo lines of code) No. of test cases written / size Number of test cases per requirement OR Number of test cases per 1000 lines of code OR Number of test cases per module
3	Schedule Variance (SV)	Schedule Variance = (Actual Duration - Planned Duration) / (Planned Duration) * 100
4	Effort Variance (EV)	Effort Variance = (Actual Efforts - Planned Effort)/Planned Efforts In general the Effort Variance Should be in between +/- 5%
5	% TC Passed	% Test cases Passed = (No. of Test cases Passed / Total no. of Test cases Executed) * 100.
6	% TC Failed	% Test cases Failed = (No. of Test cases Failed / Total no. of Test cases Executed) * 100.
7	% TC executed	% Test cases Executed = (No. of Test cases executed / Total no. of Test cases written) * 100.
8	% TC blocked	% Test cases Blocked = (No. of Test cases Blocked / Total no. of Test cases Executed) * 100
9	% TC Not Run	% Test cases Not Run = (No. of Test cases Not Run due to time constraint / Total no. of Test cases Executed) * 100.
10	TC development productivity	Number of test cases developed per day
11	TC execution productivity	Number of Test Cases executed per day OR (Total number of Test Cases executed) / Actual Effort of Testing

17.3 ***Predictions using Defect & Test Metrics***

- CRESTA will read or calculate above metrics with applicable parameters
- UCL & LCL for all the metrics will be derived from historical data
- Further it will analyze interrelation and impact of above metrics on each other
- Based on this analysis & statistics, CRESTA will predict

Sr. No.	Predictions
1	What will be defect metrics for next release?
2	Which modules may have issues ….pre-release or post-release?
3	Is there any scope to narrow down permissible range of defect metrics?
4	Have we tested enough?
5	Were the Test cases really executed or did we fudge them?
6	Are we over testing some modules and under testing some?
7	Did we really “Shift Left” in this release?

17.4 ***Impact of Defect Metrics***

Metrix \ Parameter	Defects & Issues Post Release	Efficiency in Finding Defects
Defect Density (DD)	↑	↓
Defects by severity	↑	↑
Defect Leakage (DL)	↑	↓
Defect Removal Efficiency (DRE)	↓	↑
Defect Rejection Rate (DRR)	↑	↓
Complexity of Module	↑	NA
Usage of Module by end users	↑	NA

Legend	
Direct Proportion	↑
Indirect Proportion	↓
Not Applicable	NA

17.4.1 Defects & Issues Post Release

- When defect density is outside UCL & LCL, there are more chances of finding defects post release.
- Probability of finding post release issues is high with higher number of critical & high severity defects.
- When defect leakage is outside UCL & LCL, there are more chances of finding defects post release.
- When defect removal efficiency is outside UCL & LCL, there are more chances of finding defects post release.
- When defect rejection by development team is outside UCL & LCL, there are more chances of finding defects post release.
- Complexity of module depends on complexity in development \customization of module. Complexity of business rules for that module. Higher the complexity, higher the probability of getting defects & issues post release.
- Higher the usage by end users, higher the probability of getting defects & issues post release.

17.4.2 Efficiency in Finding Defects

- Efficiency is inversely proportional to Defect Leakage. Higher the defect leakage, lower the quality of QAT efforts.
- Efficiency is directly proportional to Defect Removal Efficiency. Higher the DRE, higher the efficiency.
- Efficiency is inversely proportional to Defect Rejection. Higher the defect rejection rate, lower the efficiency.

Refer Defect_Metrics_Parameters.xlsx, for defect metrics & applicable parameters.

*17.5 **Impact of Test Metrics***

Metrix \ Parameter	Over Testing\ Under Testing	Improved Automation ROI	Shift Left in current release
Test Case Density (TCD)	↑	NA	NA
Schedule Variance (SV)	↑	NA	↑
Effort Variance (EV)	↑	NA	↑
Automation ROI	NA	↑	NA
Finding Defects Early	NA	NA	↑

Legend	
Direct Proportion	↑
Indirect Proportion	↓
Not Applicable	NA

17.5.1 Over Testing & Under Testing

- When TCD is above upper control limit for a module that module will be over tested
- When TCD is below lower control limit for a module that module will be under tested
- When EV is above UCL that module\sprint is over tested
- When EV is below LCL that module\sprint is under tested
- When SV is above UCL that module\sprint is over tested
- When SV is below LCL that module\sprint is under tested

17.5.2 Shift Left in current release

- *Shift Left - Finding defects earlier to save cost of fixing defects*
- *When defect is found in requirements or design phase, cost of fixing it is significantly low as compared to fixing it in test or production phase*

Refer TestCase_Metrics_Parameters.xlsx, for test case metrics & applicable parameters.