Report on value-measure using automated Tool

# Manju’s IDEA

One of the things I wanted to figure out is if there is a way to get developer productivity from the code we check in. We will need to look at multiple attributes such as lines of code, quality (attributes from sonar report), complexity of code and link it to the JIRA ticket and the hours spent. Since we can always tie a checkin to a JIRA ticket we will know the time taken. We can even build a model to predict this too may be. If we can do this, then we have a way to measure what our developers are doing. We will then need a way to add quality ( feature testing ) to this and also design skills ( scale etc.).  Keep this confidential – This could be our IP so do not blog or open source it.

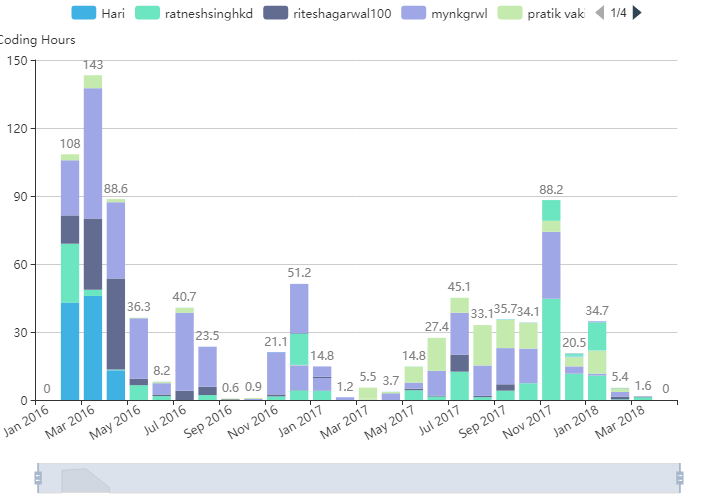
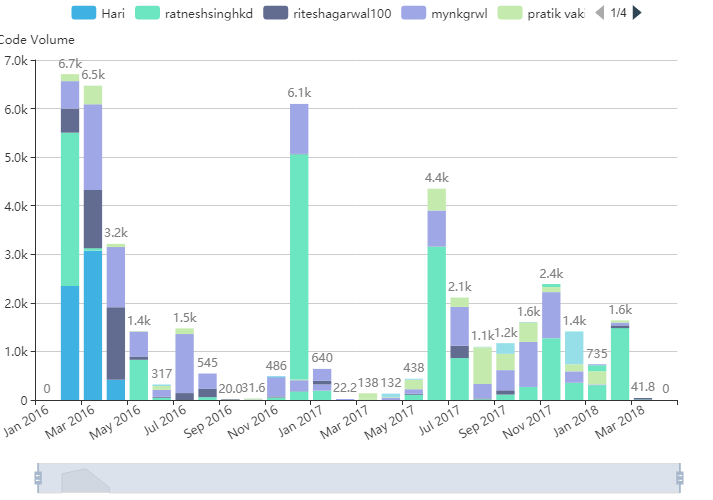
Take a look at gitential. We can discuss this further.

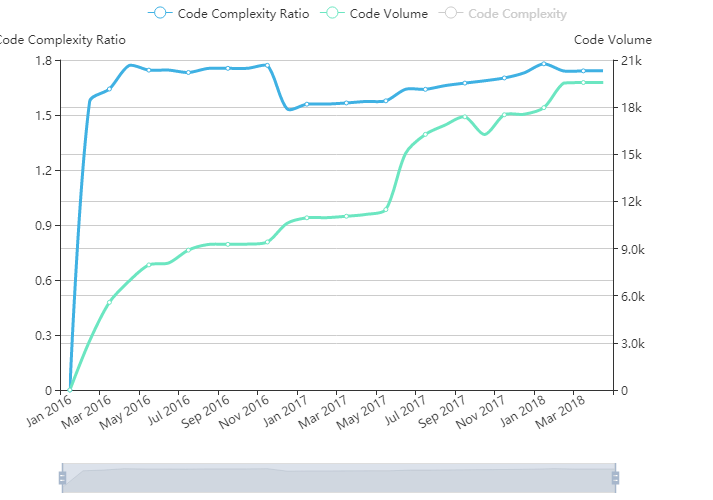
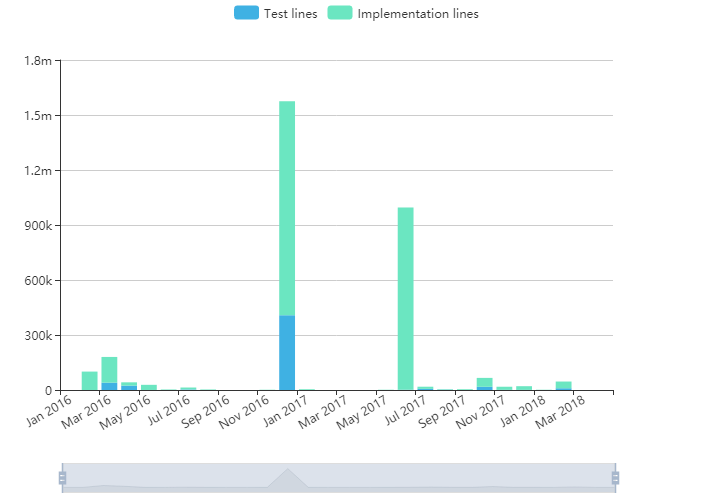
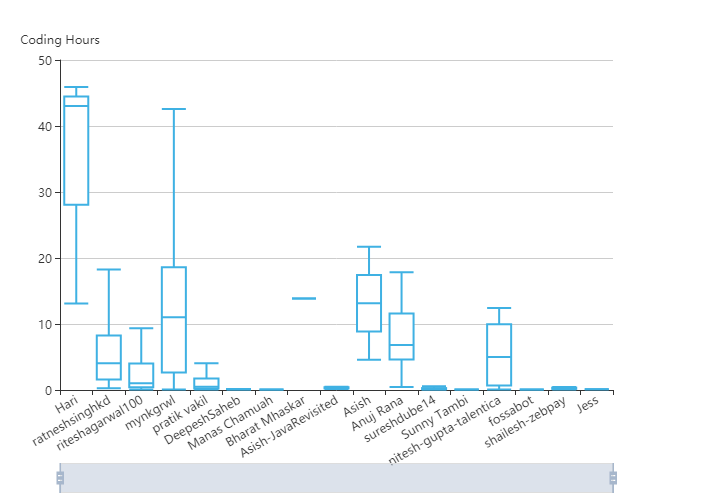
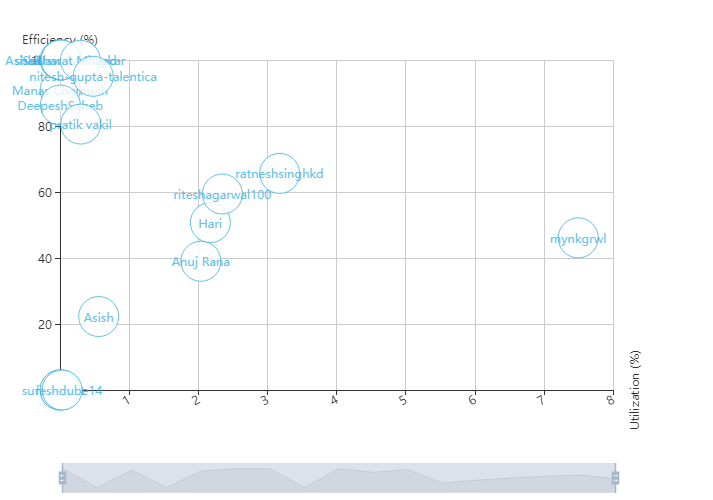
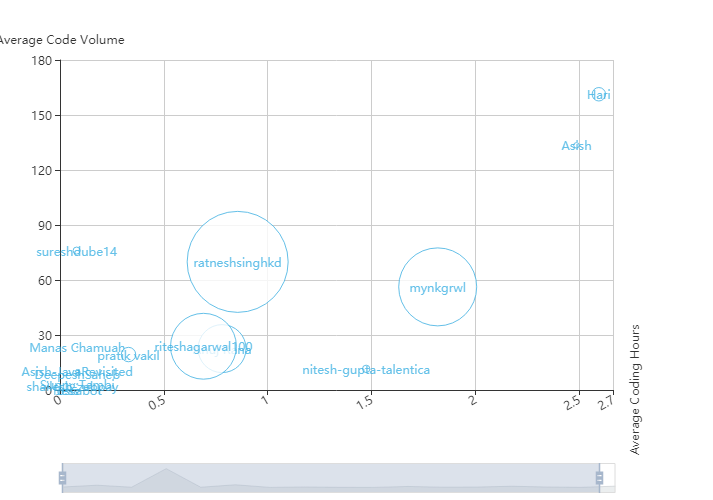
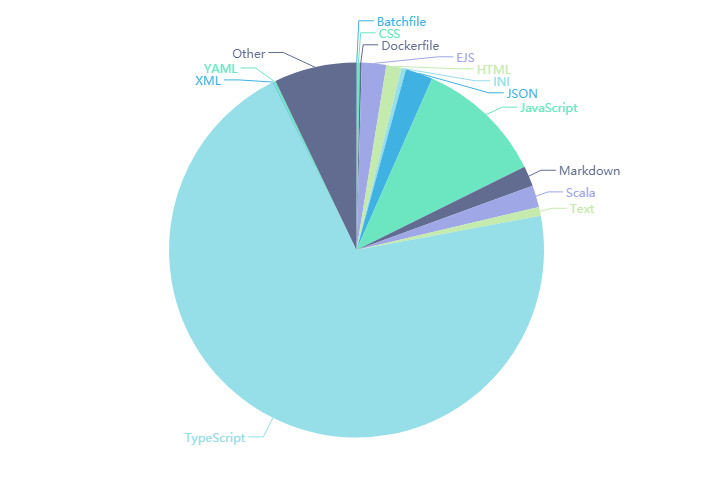
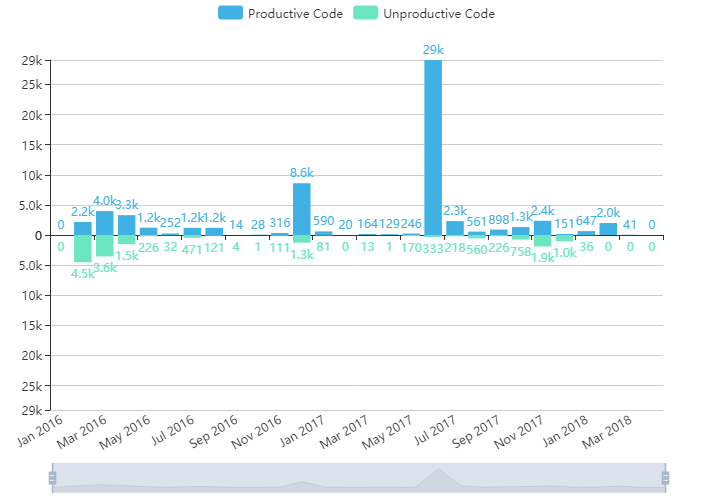
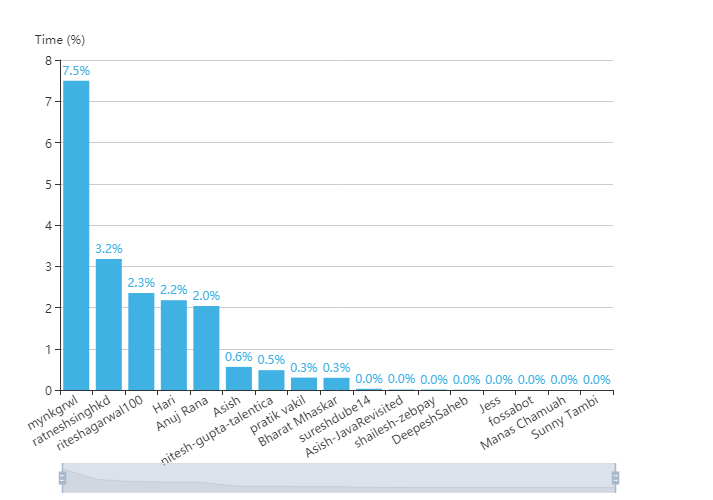
# Ratnesh’s investigation on gitential

# The gitential  collects git commits for a period and provide some matrices on productivity using heuristic logics

1. gitential could be good but if Clients see reports following situation could happen (<http://blog.gitential.com/2017/12/08/in-god-we-trust/>), so there could be problems sharing with client like their coding hours is not known
2. gitential requires pull credentials (though they don't collect code just git commits) , I have checked their data set, they don’t have code.
3. It works on github (public , private ),bitbucket, TFS , support ssh for others
4. For internal use There might be a way for them to create reports just using logs (they need only 4 log files form git which doesn’t include code),
5. I tried out with node-data here you can see reports(attached 10 reports)
6. It clearly identifies the major and minor contributors
7. It works on almost all languages (js, python, java ,etc.)
8. I still don’t know their heuristic logic for calculating coding hours from git commit buts its consistent , It identifies when we put lot of features and when we hardly write any code
9. Their matrices are consistent and matches with commits and productivity
10. Its Handle the duplicates code , duplicate commits
11. It cost 20 $ per developer per month , we can ask them for custom quote
12. We can generate matrices for the VM reports from this ( It might not be against sprints but monthly, quarterly make sense)
13. This blogs tries to give many answers on their methodologies (<http://blog.gitential.com/2017/11/24/measuring-software-development/>)
14. This tool don’t show expertise on developers but I can clearly see the developer’s lacks expertise, training and experience will always score way less than others.
15. This tools works better if you have minimum of 6 months data

Gitential reports (from node-data project using gitentials)





# Ratnesh’s Tool

Here are the updates (06-04-2018)

1. I am able to create data and matrices like genital for following matrices (Development Hours, code churn, utilization, and productivity) (see attached image)
2. Right now the code is a command line tool which can be run over o pulled git repository which predicts the efforts in development hours and code volumn in csv .
3. The report I have created using tableau.
4. Code, data and tableau report is not checked in anywhere.

Next tasks

1. Automating the tool to work on multiple repositories (we will have multiple repositories).
2. Need to work on anomalies (like immediate code churn)

       2. These matrices are as per gitential but we need to create our own matrices as per our VM

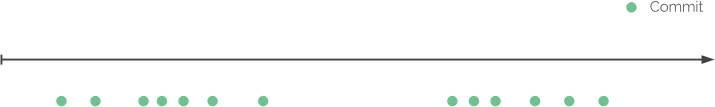
       3. Check feasibility for adding dimensions from sonar and jira

# Sample Git Commit

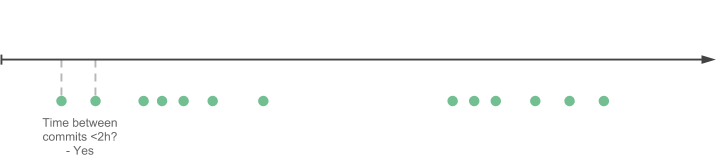


## **Algorithm for estimating hours**

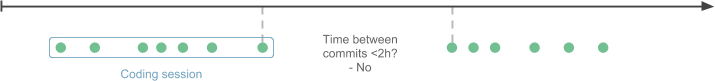
The algorithm for estimating hours is quite simple. For each author in the commit history, do the following:



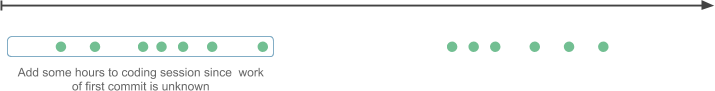
Go through all commits and compare the difference between them in time.



If the difference is smaller or equal then a given threshold, group the commits to a same coding session.



If the difference is bigger than a given threshold, the coding session is finished.



To compensate the first commit whose work is unknown, we add extra hours to the coding session.



Continue until we have determined all coding sessions and sum the hours made by individual authors.

# Anomaly Detection

1. Large no of files added in single commit (first time or bootstrap code committed) (100)

2. Ignore file types (auto generated, protobuff, package.lock.json)

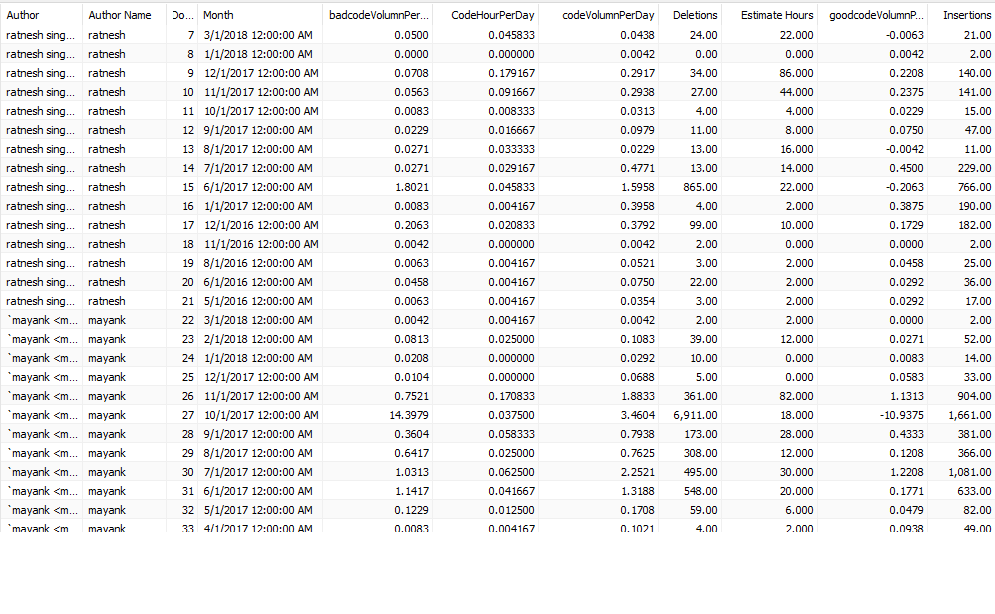
3. Large no of lines added in a file in single commit (1k)

4. Large no of lines added in single commit (all files)(5k)

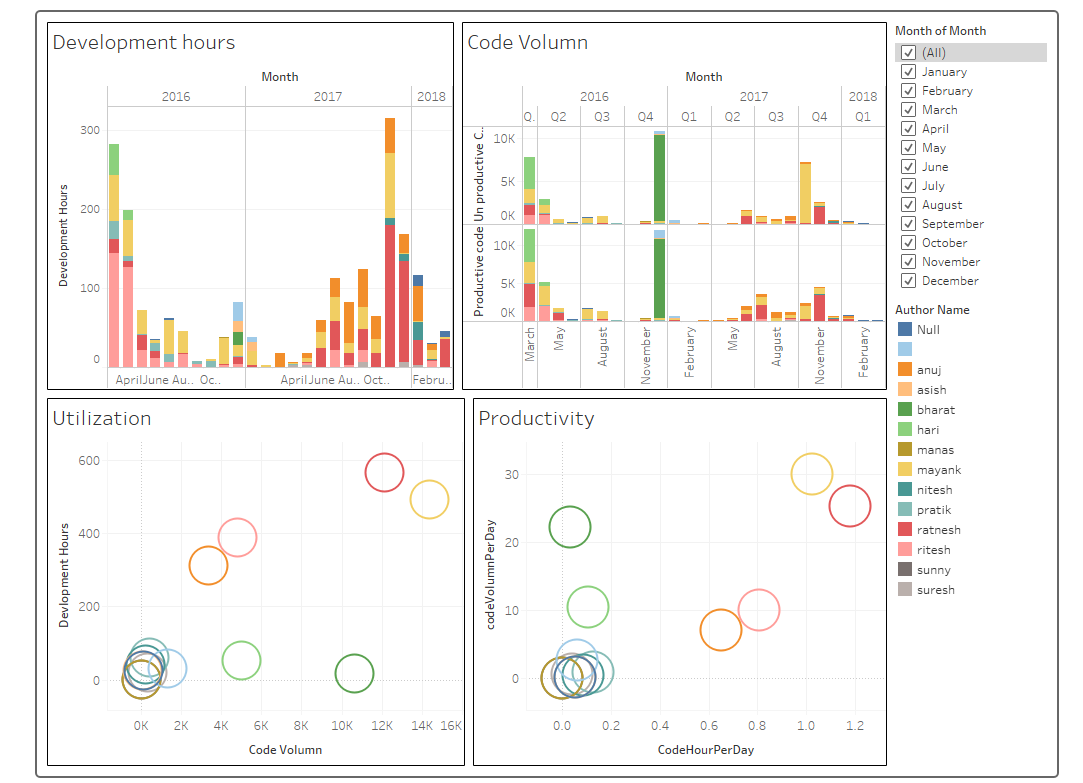
5. Difference between lines added and lines deleted (<5 %) (Reformatting, copy-paste)

6. Large no of commits with in a duration (1 Hr , 100 commits )

# Format of output CSV



# Output of tool



# Code Quality

Code quality is output from sonar-qube .Essentially Its should run beginning of the term and end of the term.

Support for (JS , TS, JAVA , scala , python ,php , html, jsx ,etc)

|  |
| --- |
| **EntityQualityDetail** |
| entity-name |
| lineno |
| noOfLines |
| IssueSeverity(major,minor,crtical , blocker |
| IssueDebt(5min, 1day) |
| IssueType(code-smell, complexity ,design) |

The above detailed data can be summarize into following entity wise data

|  |
| --- |
| **EntityQualityScore** |
| entity-name |
| ScoreBegining |
| ScoreCurrent |
| ScoreImprovement  Loc  Score-per-line |

For a team Overall quality score could be avg(Score-per-line)

For an individual if we know the contribution percentage of author we can provide the score

# Project Cicero

Project Cicero has been created to figure out following Dimensions

Design Score, leadership score, Ownership score, collaboration score . Details can be found in git project(not decided yet)

Combine with productivity, utilization, quality final Data mining output will le look like this.

Color blue represents output of project Cicero

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **EntityAge** |  | **AuthorProductivity** |  |
|  | entity-name |  | author-name |  |
|  | Age-months-start |  | month |  |
|  | Age-months-last |  | year |  |
|  | AgeScore | **EntityAuthorEffort** | development-hours |  |
|  |  | entity-name | productivecode-volume |  |
|  | **EntityChanges** | author-name | unproductive-code-volume |  |
|  | entity-name | author-revs | no-of-days | **AuthorCommunication** |
|  | no-of-authors | total-revs | code-hour-per-day | author-name |
| **EnityCoupling** | no-of-changes |  | code-volume-per-day | peer-name |
| enity-name |  | **AuthorEntityOwnerShip** |  | shared |
| coupled-enity-name | **EntityChurn** | entity-name | **AuthorLeadership** | average |
| degree | entity-name | author-name | author-name | strength |
| avg\_revs | no-of-additions | no-of-additions | no-of-peers |  |
|  | no-of-deletions | no-of-deletions | no-of-subordantes |  |
|  | no-of-commits | no-of-commits | shared |  |
|  |  |  | average | **MainDevelopersByContributing** |
|  |  |  | strength | entity-name |
|  |  |  | LeaderShipScore | main-author-name |
| **EntityQualityDetail** | **EntityQualityScore** |  |  | added |
| entity-name | entity-name |  |  | total-added |
| lineno | ScoreBegining |  | **AuthorOwnerShip** | ownership |
| noOfLines | ScoreCurrent |  | author-name | byrevs |
| IssueSeverity | ScoreImprovement |  | no-of-components-owned | byloc |
| IssueDebt |  |  | total\_ownership\_size |  |
| IssueType |  |  |  | **MainDevelopersByRefactoring** |
|  |  |  |  | entity-name |
|  | **EntityFragmentation** |  | **AuthorChurn** | main-author-name |
|  | entity-name |  | author-name | removed |
|  | fractal-value |  | Added | total-removed |
|  | total-revs |  | Deleted | ownership |
|  |  |  | commits |  |

# Explanation of Parameters

**Design Score: - Stability (Entity Age, AuthorEntityOwnerShip)**

**Churn(EntityChurn , AuthorEntityOwnerShip, EntityChanges)**

**Coupling(EnityCoupling , AuthorEntityOwnerShip)**

Logical coupling refers to modules that tend to change together.

Calculates the degree of logical coupling.

Returns a seq sorted in descending order (default) or an optional, custom sorting criterion.

The calulcation is based on the given coupling statistics.

The coupling is calculated as a percentage value based on

the number of shared commits between coupled entities divided

by the average number of total commits for the coupled entities.

**Ownership Score :- This Score comes out of Author Ownership (which comes out of MainDevelopersByContributing , MainDevelopersByRefactoring)**

**Leadership Score :- Author Leadership (Which comes from Author Communication )**

This Measure attempts to give some heuristics on

the communication needs of a project.

The idea is based on Conway's law - a project

works best when its organizational structure is

mirrored in software.

The algorithm is similiar to the one used for

logical coupling: calculate the number of shared

commits between all permutations of authors.

Based on their total averaged commits, a

communication strength value is calculated.

# Sample Dashboard and report from project Cicero

----coming soon

# Sample Scoring and team Dashboard

----coming soon

# Value Measures

With proper scoring model above data can be crunch into following matrices

Explanation of Higher Level Value Measures for Individual contributors in Last Six months

1. Development Hours
2. Coding Volume
3. Quality Score
4. Design Score
5. Ownership Score
6. Leadership Score

|  |  |  |  |
| --- | --- | --- | --- |
| **Measure** | **What it is** | **How It is measure** | **Expected Values** |
| Development Hours per Day | A Heuristic measurement Indicating no of Hours Spent during durations Per Day. Unit is No Of Hours. Higher the better. | We identify coding sessions (continuous hours of coding) and start time to end time we calculate as Development Hours. For first commit of Session we give extra values. | <1 Bad  1-2 ok  2-3 good  >3 excellent |
| Coding Volume per Day | A statistical measurement about no of lines of code committed Per Day. Higher the better. | Source is git /svn which provided details like no of lines added/deleted/Modified in each commit. | Bootstrap (<100 bad ,100-200 good, >200 excellent ),Scale(<50 bad , 50-100 good , >100 excellent)Mature(<10 bad , 10-50 good, >50 excellent) |
| Quality Score | An aggregated value (avg) of all 5 point quality (Maintainability, security, reliability, complexity, cognitive complexity) measurements, lesser the better. | Sonar Generated 5 point quality for all code commits. We know the commit author and take industry standard average. | <=1 Excellent , 1-2 good/ok, 2-3 bad , >3 very bad |
| Design Score (High I2 and above) | An aggregated value (avg) of all 3 point Design (Coupling, Stability, Churn) measurements, lesser the better. | Our Code Generated 3 point Designs for all code commits. We know the commit author and take standard average. | <=1 Excellent , 1-2 good/ok, 2-3 bad , >3 very bad |
| Ownership Score (I3 and above) | A statistical measurement about no of components owned by an individual. | For Each component in source code, we Identify The Ownership of authors using tree visiting on source code. | >5 Excellent , 3-5 good , 2-3 ok ,<2 Bad |
| Leadership Score (I3 and above) | An aggregated value (Sum) of all 3 point Design (Initiative, sharing, Strength) measurements, Higher the better. | Our Code Generated 3 point Leadership for all code commits. We know the commit author and take summation. | >5 Excellent , 3-5 good , 2-3 ok ,<2 Bad |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| In Last Six month |  |  |  |  |  |  |  |
| **Individuals(1-10)** | Weightage | **Team(1-10)** | bootstrap(<6) | low-Scale(6-12) | medium-scale(12-24) | High Scale(24-36) | mature(>36) |
| Utilization Score |  | Utilization Score |  |  |  |  |  |
| Productivity Score |  | Productivity Score |  |  |  |  |  |
| Bad Qulaity Score |  | Bad Qulaity Score |  |  |  |  |  |
| Bad design Score |  | bad Design Score |  |  |  |  |  |
| LeaderShipScore |  | Collaboration Score |  |  |  |  |  |
| OwnershipScore |  | Architecture Score |  |  |  |  |  |
|  |  | Dev process Score |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| QulityImporvementScore |  | QulityImporvementScore |  |  |  |  |  |
| DesignImprovementScore |  | DesignImprovementScore |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# Timelines of Project

|  |  |  |
| --- | --- | --- |
| Timelines | | |
| **Task** | **Status** | **ETA** |
| gitential analysis | done |  |
| knowledgebase | done |  |
| Data mining | done |  |
| Scoring Algorithms | inprogress |  |
| Automation | not started | 1 week may |
| run for 2 bootrap team |  | 2nd week may |
| run for 2 low scale team |  | 3rd week may |
| run for two high scale team |  | 4th week may |
| run for two mature team |  | 5th week may |
| fix anamolies |  | 1st week jun |
| sonar runner for all team |  | 2nd week jun |
| revise scoring model |  | 3rd week jun |
| presentation |  | 15th June |
| Execution for all teams |  | sanity till 20 sept |
|  |  | execute from 20-25 sept |