SamikshaViz: A Panoramic View to measure Contribution and Performance of Software Maintenance Professionals by Mining Bug Archives

Ayushi Rastogi, Ashish Sureka
Indraprastha Institute of Information Technology, Delhi (IIIT-D)
{ayushir, ashish}@iiitd.ac.in

Abstract—In software maintenance teams, appraisers discern the need to scrutinize underpinning factors involved in performance appraisal and justify performance. However the challenge lies in visually demonstrating distinguished performance based on performance metrics or Key Performance Indicators (KPI) and interplay of environmental factors for a role. The objective of this work is to address the above mentioned requisites of appraiser of software maintenance teams by visual presentations. ‘SamikshaViz’ extends the idea proposed in ‘Samiksha’. ‘Samiksha’ is a framework for contribution and performance assessment of software maintenance professionals by mining software repositories. ‘Samiksha’ proposes 11 role-based contribution and performance assessment metrics and demonstrate the results on real-world data of Google Chromium Issue Tracking System. The requirements gathering of factors influencing performance is based on survey conducted on experienced software maintenance professionals with more than a decade of experience.

In this paper, we propose a framework for visualization techniques to study panoramic view of individual’s contribution and performance in a team. The focus of the study is to demonstrate the application of 6 visualization techniques on role-based contribution and performance assessment metrics proposed in ‘Samiksha’. The inferential power of ‘SamikshaViz’ is established on Google Chromium Issue Tracking System dataset. These visualization techniques reinforce and extend the rationale of the metrics proposed in ‘Samiksha’ and gather insights for appraisers to justify performance. The results are validated by practitioners in industry.

Index Terms— Contribution and Performance Assessment Metrics, Visualization, Mining Bug Archives, Issue Tracking System, Software Maintenance

I. RESEARCH MOTIVATION AND AIM

“What you cannot measure you cannot control” sets the broad motivation of the work presented in this paper. Contribution and performance assessment of employees and workers is a standard human resource management practice followed in organizations world-wide [1]. Previous research shows that contribution and performance assessment of people is fraught with several challenges and imperfections that negatively influences employees’ motivation and organization’s growth [1][2]. Solutions to measure contribution of developers in Software Engineering domain is an area that has attracted several researchers’ attention [3][4]. Rastogi et al. present a framework called as ‘Samiksha’ (a hindi term which means to review or critique) to measure the efficiency and effectiveness of software maintenance professionals such as bug reporters, bug triagers, bug fixers and bug collaborators [3]. They conduct a survey of Software Maintenance Professionals to understand their views, propose several role-based metrics (based on mining activity data from an Issue Tracking System) and conduct a series of experiments on real-world dataset from the Issue Tracking System of Google Chromium open-source project [3]. The work present in this paper is a significant extension of our previous work on ‘Samiksha’ (the extension is called as ‘SamikshaViz’).

The study present in this paper is motivated by the need to enhance the usability and effectiveness of the basic ‘Samiksha’ framework by adding a visualization layer to the existing framework. Organizations store large information in their databases to support informed decision making [5]. However, a prime challenge faced by decision makers (appraisers) is the analysis of large, complex and constraint data for evaluation. Recent researches introduce ‘BizViz’ (Business Visualization) as a buzz word for decision makers to quickly examine large data, analyze trends and outlier behavior and derive actionable insights [2][6][7]. Visualization can support decision makers in comprehending large amounts of data and extracting patterns which are not apparent by viewing the data in raw form. Data visualization is a vast field and several advanced visualization tools are available in the commercial and open-source domain [8][9][10]. The research aim of the work present in this paper is the following:

1) To investigate visualization tools and techniques in context to the problem of visualizing contribution and performance assessment data of software maintenance professionals mined as an output from an existing framework called as ‘Samiksha’.

2) To investigate the effectiveness of the proposed visualization solution (tool support serving as a guide to the target user) by studying the complex interplay of variables, trends, outlier behavior, patterns and regularities and extract actionable information and insights from the perspective of decision maker and get their validation.

II. RELATED WORK AND RESEARCH CONTRIBUTIONS

In this section, we present closely related research studies to the work in this paper, identify research gaps and establish novel contribution of ‘SamikshaViz’. Broadly the related work can be categorized into following lines of research: visualization in business, visualization in software industry by mining
software repositories, and contribution and performance assessment by mining software repositories.

To facilitate visualization in management domain Zhang et al. propose a theoretical, general visualization model. They identify five processes viz. domain problem space analysis, domain data and knowledge collection, pattern discovery and data aggregation, and image construction as theoretical foundations to achieve data comprehension and improve problem solving performance [7]. Schonhage et al. build a prototype for visualization in business to aid managers analyze past and present data and venture with future situations [6].

Visualization ease software understandability and analyze evolution of software ecosystem [9][11]. It also find its application in analyzing software team. Storey et al. propose a framework to describe, compare and understand human centric awareness visualization tool in software development [4]. Taylor et al. introduce the concept of author entropy to characterize authorship. Author entropy in conjunction with other software metrics has the potential to identify areas of concerns within source code [12]. Gilbert et al. explore group dynamics to compare developers and their contribution in a distributed software community by analyzing social visualization code [8]. Robles et al. propose a novel methodology to visually analyze evolution of core team to ensure smooth transitions by identifying breakpoints and unevenness [13]. However, to the best of our knowledge existing literature does not explore the expressive power of visualization to analyze contribution for justified performance appraisal by mining software repositories.

‘SamikshaViz’ extend our previous work ‘Samiksha’ [3] and complements it. In ‘Samiksha’, we conduct a survey on experienced software maintenance professionals to identify real-world organizational needs with the objective of proposing a framework for accurate, reliable and objective measurement of contribution and performance. We formulate 11 role-based contribution and performance assessment metrics for the four roles of software maintenance professionals viz. bug reporter, bug triager, bug owner and bug collaborator. The experimental results are present on real-world data of Google Chromium Issue Tracking System [3]. A brief account of the metrics proposed in ‘Samiksha’ is provided in Table II. Table II state the rationale and the role of software maintenance professionals for which metric is proposed.

In context of existing literature, the work present in this paper makes the following novel contributions:

1) We propose 6 visualization techniques for the four roles of software maintenance professionals viz. bug reporter, bug triager, bug owner and bug collaborator. The unique contribution of this work is a visualization framework (panoramic view) to endow appraisers the ability to justifiably distinguish performance by deriving actionable insights. The visual framework present in this paper builds on the metrics proposed in ‘Samiksha’ (refer Table II). We see application of visualization in software industry to analyze process and generate awareness. However, its application to measure contribution and performance of software maintenance professionals by mining software repositories is unexplored to the best of our knowledge.

2) Implementation of proposed visualization techniques on real world data of Google Chromium Issue Tracking System, infer actionable insights and validate its usability based on survey responses from professionals.

III. RESEARCH METHODOLOGY AND FRAMEWORK

‘SamikshaViz’ use the general visualization model proposed by Zhang et al. [7]. We start by studying variegated sources for unconventional visualizations. We identify approximately 30 unique visualizations. For each visualization, we analyze the rationale behind its proposition and the nature of information that can be inferred from the plot. We then map these visualizations to the metrics to capture multiple information needs of decision makers. Each visualization technique maps to multiple metrics. However, due to limited space availability and for the ease of comprehension we present one metric for a visualization (refer Section IV). All visualizations present in this work are written in R language (GNU project for statistical computing and graphics).

The experimental dataset for analyzing the results is build on Google Chromium Issue Tracking System. The choice of dataset is motivated by the following reasons: 1) publicly available 2) dataset is replicable (can be downloaded using Issue Tracking System API) 3) wide usage and popularity 4) used in ‘Samiksha’ (for experimental analysis of metrics). A detailed description of the dataset is provided in Table I. Unless specified otherwise all analysis is conducted on ‘Closed’ bug reports with status ‘Fixed’ or ‘Verified’.

We assign each developer in the organization a unique identity. The unique identity is of the form ‘dxxxx’ where ‘x’ is any numeric value. This unique identifier masks true identity (anonymous behavior for ethical reasons) and ensures uniformity (in identification) across roles and their visualizations. In two consecutive years (2009-11), 34,056 unique developers contribute towards Google Chromium project (i.e. Issue Tracking System dataset). Developers are assigned unique identities in the alphabetical order of developer identity.

Dataset to create visualization for a role encompass less number of developers compared to the count of developers

\[\text{http://www.r-project.org}\]
for a given role (as mentioned in Table I). It may happen due to exclusion of data points with missing or inconsistent information, considering developers who worked for two years and setting thresholding criteria on selection of developers for analysis.

Post-implementation we conduct survey of practitioners. We receive 10 valid responses from practitioners in large global IT industry. 9 survey respondents have more than 5 years of experience and 1 had more than 1 year of experience. 8 survey respondents played role of Project Manager, 1 was head of R & D initiatives and 2 Bug Owner/Bug Fixer with overlapping roles. 8 out of 10 survey respondents had been appraisers in past or present. Table IV shows usefulness of visualization techniques when compared with existing practices in organization. Approximate Team Size 15-20 members 50-75 members 25-50 members 50-100 members 15-20 members 15-20 members

<table>
<thead>
<tr>
<th>Features</th>
<th>Trellis Plot</th>
<th>Treemap Plot</th>
<th>Bertin’s Hotel</th>
<th>Dart Chart</th>
<th>Hybrid Plot</th>
<th>Nightingale Rose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>SRI</td>
<td>PCLPCI-1,CJ</td>
<td>SBI</td>
<td>PWFI</td>
<td>DMTTR</td>
<td>PWFI</td>
</tr>
<tr>
<td>Data dimensionality</td>
<td>30-75 members</td>
<td>25-50 members</td>
<td>30-100 members</td>
<td>15-20 members</td>
<td>15-20 members</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>Study trends and outlier behavior by systematic arrangement of variables; Temporal analysis of performance; Measure performance relatively</td>
<td>Map data hierarchically; Panoramic view to individual’s contribution in a team; Compare and contrast performance relatively</td>
<td>Pattern analysis from rearrangement of rows and columns based on criteria</td>
<td>Establish qualitative equivalent of quantitative score; Analyze temporal variations in performance; Cumulative statistical data for each class of observations</td>
<td>Study positive and negative contribution</td>
<td>Impact of environmental factors on performance; Compare actual vs expected</td>
</tr>
</tbody>
</table>

TABLE IV

<table>
<thead>
<tr>
<th>Metric</th>
<th>Question Not at all Effective(1)</th>
<th>Slightly Effective(2)</th>
<th>Moderately Effective(3)</th>
<th>Very Effective(4)</th>
<th>Extremely Effective(5)</th>
<th>Total</th>
<th>Mode</th>
<th>PoR</th>
<th>C PoR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dart Chart</td>
<td>RQ1 0</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ2 0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>7</td>
<td>3</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Bertin’s Hotel</td>
<td>RQ1 0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>10</td>
<td>4</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ2 0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>4</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ3 0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ1 2</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ2 2</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>4</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ3 2</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>7.5</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Trekmap Plot</td>
<td>RQ1 2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>3.4</td>
<td>62.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ2 2</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>87.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ3 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ4 1</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ1 0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>85.71</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ2 2</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ4 2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>62.5</td>
<td></td>
</tr>
<tr>
<td>Trellis Plot</td>
<td>RQ1 0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>3</td>
<td>75.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ2 2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>66.96</td>
<td></td>
</tr>
<tr>
<td>Hybrid Plot</td>
<td>RQ1 2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>3</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ2 0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>66.96</td>
<td></td>
</tr>
<tr>
<td>Nightingale Rose</td>
<td>RQ1 0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>3</td>
<td>66.96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ2 0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>66.96</td>
<td></td>
</tr>
</tbody>
</table>

IV. VISUALIZATIONS

In this section, we discuss 6 visualization techniques for the four roles of software maintenance professionals viz. bug reporter, bug triager, bug owner and bug collaborator. This is not an exhaustive list, however it gives a glimpse of decision maker’s needs and requirements. The requirement of each visualization technique is triggered by the information need. The type of information that can be inferred from a visualization is mentioned in Table III. Table III is an analysis of the 6 visualization techniques proposed in ‘SamikshaViz’ and compares them. Each visualization in ‘SamikshaViz’ maps to multiple metrics (proposed in ‘Samiksha’). However, in this work we demonstrate one metric for each visualization.

1) Trellis Plot: Trellis plot is a visualization technique to uncover relationship of variables in a multivariate dataset.
Systematic application of Trellis plot show how a response depends on explanatory variables. We use Trellis plot visualization to analyze and interpret the performance of bug reporters using Status of Reported bug Index (SRI) metric (refer to Table II).

Figure 1 is an arrangement of rows, columns and panels. Each panel in Figure 1 is a status of the bugs reported with state ‘Closed’. These panels are arranged in the non-increasing order of importance of each status towards contribution and performance. Rows in each panel are the bug reporters arranged in non-increasing order of the number of bugs reported annually. Bug reporters can be broadly categorized into five heads based on performance as ‘Excellent’, ‘Very Good’, ‘Good’, ‘Satisfactory’ or ‘Non-Satisfactory’. The criteria for classification is specified in caption of Figure 1. This classification criteria of bug reporters can be fine tuned to specific organizational needs.

Number of bugs reported is a measure of contribution. However, the status with which a bug is ‘Closed’ influence contribution. For instance, reporting bug reports ‘Closed’ as ‘Fixed/Verified’ adds more to the contribution than a ‘Duplicate’ bug report. This standpoint is captured in SRI metric (refer to Table II). Following are interpretations for performance appraiser:

1) Which bug reporters have high efficiency? Efficiency is magnitude of time, effort or cost utilized to achieve target. In Figure 1 we observe that bug reporter ‘d28779’ reports more bug reports than bug reporter ‘d0001’ (bug reporters arranged top to bottom in non-increasing order of total number of bugs reported). However, bug reporter ‘d0001’ has higher SRI score than bug reporter ‘d28779’ (as shown in pink on right). Thus we conclude that bug reporter ‘d28779’ has higher efficiency than bug reporter ‘d0001’. The justification is that bug reporter ‘d28779’ places lot of efforts, however the usefulness of contribution to the organization is substantially less.

2) How bug reporter’s performance vary w.r.t. time? In Figure 1 we see that for large parts blue dots (percentage of bugs reported in year 2010-11) are ahead of red dots (2009-10). However, in panel ‘Fixed/Verified’ we see two colored dots swapped and separated by large distances (w.r.t. other developers) for bug reporter ‘d22575’. We infer that performance of bug reporter ‘d22575’ declined considerably in the year 2010-11 (w.r.t. year 2009-10).

Other interesting questions:

1) Which bug reporters add overhead (in terms of resource) to the organization through their contribution?
2) What is the relative performance of a bug reporter?
3) Which bug reporter’s bugs achieve less attention by team and hence get ‘Closed’ as ‘IceBox’?
4) Which bug reporters report large number of ‘Invalid’ or ‘Duplicate’ bugs?

Fig. 1. Trellis Plot: Horizontal axis is percentage of bugs reported by bug reporter (shown on vertical axis in blue) for a given status (each panel) for two consecutive years 2009-10 and 2010-11 (represented by symbol variables: Red for the year 2009-10; Blue for the year 2010-11). Bug reporters arranged (top to bottom) in the non-increasing order of number of bugs reported followed by their score on Status of Reported bug Index (SRI) metric (in pink). 20 bug reporters are categorized into performance heads as ‘Excellent’, ‘Very Good’, ‘Good’, ‘Satisfactory’ and ‘Non-Satisfactory’ (based on the total number of bugs reported) [‘Excellent’ > 250 bug reports, ‘Very Good’ > 200 and < 250 bug reports, ‘Good’ > 100 and < 200 bug reports, ‘Satisfactory’ > 50 and < 100 bug reports, and ‘Non-Satisfactory’ < 50 bug reports]. Status arranged (top to bottom) in non-increasing order of relevance (in terms of contribution) for the role of bug reporter.

2http://en.wikipedia.org/wiki/Efficiency
A. Treemap Plot

Treemap maps hierarchical information in a space-efficient and space-filling manner. It partitions the display space in rectangles. Each rectangle (sub-partition) is a child node with two distinguishing features: size and color. Size of a child node is proportional to weight of sub-partition (w.r.t. whole) and color represents node-specific property. Thus, a Treemap encodes information as partitions (sub-partitions), size and color [17][18].

In Figure 2, Potential Contribution Index and Potential Contribution Index-1 (refer to Table II) of bug collaborator shown relatively is size of sub-partition of a component indicating expertise and specialization. Also Contribution Index (refer to Table II) in Figure 2 is denoted by the color of sub-partition for each bug collaborator in Treemap. Color of sub-partition indicates initiative, enthusiasm and contribution of bug collaborator (detailed description in caption of Figure 2).

We present contribution of bug collaborators in 5 most prominent components (areas in which maximum number of bugs were reported in year 2010-11, except areas like 'Undefined'). To ease comprehension in limited space of the paper, we include bug collaborators who contribute on more than 50 bug reports in an area for analysis. In Figure 2 bug collaborators are arranged bottom-to-top and left-to-right in the non-increasing order of Potential Contribution Index (refer to Table II).

Color-to-Size ratio indicate bug collaborator’s performance. A higher color-to-size ratio shows performance more than expected (indicator of initiation and enthusiasm) and vice-versa. Following are the inferences:

1) Which bug collaborators are high performers? Figure 2 separate bug collaborator ‘d17756’ from its neighborhood with contrasting color. Bug collaborator ‘d17756’ work significantly in three out of five reported areas in Treemap and is an asset to organization.

2) Which bug collaborator is a critical resource to the organization? In area ‘Feature’ of Treemap plot we see that only bug collaborator ‘d17465’ work on more than 50 bug reports. Thus, bug collaborator ‘d17465’ is a critical resource to the organization.

3) Which bug collaborators contribute more than expected? In area ‘UI’ of Treemap plot we see that size of sub-partition for bug collaborator ‘d29449’ is comparable to size of sub-partitions for neighboring bug collaborators like ‘d06974’, ‘d14528’ etc. It shows similar Potential Contribution Index (refer to Table II) for above mentioned bug collaborators. However, higher color-to-size ratio for bug collaborator ‘d29449’ shows contribution more than expected.
B. Bertin’s Hotel Plot

Bertin’s Hotel is a homogenous structure that use rearrangement of rows and columns to reveal information of interest. Input to the plot can be simple face values or complex derived (transformed) versions of the input. Global scaling is applied to the data to support homogenous nature. The expected outcome of this plot is a decision leading to action [19]. In our context, this visualization is motivated by Specialization and Breadth Index metric (refer to Table II) for the role of bug owner.

Specialization and Breadth (of knowledge) are important for the role of bug owner and relate inversely. A bug with dependencies across components can be very well fixed by bug owners with some understanding of all linked components. Similarly, a bug which deeply embed in a component may require bug owners with specialized knowledge in the area.

Bertin’s Hotel plot is a matrix of areas and bug owners (detailed description in caption of Figure 3). Figure 3 shows the gross performing trends of bug owners for the year 2010-2011. The contribution of bug owners in these areas is range normalized (refer equation 1) to ensure homogeneity for cross-comparison.

\[ \text{RangeScore} = \frac{x - \text{min}}{\text{max} - \text{min}} \]  

In Figure 3, we see that the upper half triangle is a shade of gray. It implies that as we go from top to bottom breadth of knowledge decreases and specialization increases. Following are the inferences relevant to appraisers:

1) **Which bug owners have specialized knowledge and vice-versa?** In Figure 3, we see that bug owner ‘d09728’ work across multiple areas (except area ‘BrowserBackend’). Also the collaboration pattern is almost uniform across areas (as shown by the slightly varying shades of gray). Thus, bug owner ‘d09728’ has breadth of knowledge. Similarly we see in Figure 3 that bug owner ‘d02209’ own bug reports from two areas viz. ‘Internal’ and ‘BrowserBackend’. A striking contrast in shades of gray in Figure 3 shows that bug owner ‘d02209’ has specialized knowledge in area ‘BrowserBackend’.

2) **Which bug owners are critical to my project?** For area ‘ChromeFrame’ in Bertin’s Hotel plot, we see that bug owner ‘d01682’ own maximum number of bug reports (dark shade of gray in column ‘ChromeFrame’ of Figure 3). Apart from bug owner ‘d01682’, no other bug owner contribute to area ChromeFrame. Thus, bug owner ‘d01682’ with expert knowledge in ChromeFrame is a critical resource to the organization.

3) **Which bug owner meet my project specific requirements?**

Bug reports with high dependency across projects need bug owners with breadth of knowledge and vice-versa. For instance, project in which multiple bug reports have dependencies in ‘UI’ and ‘Feature’ will be best solved by bug owner ‘d28504’. We see in Figure 3 that bug owner ‘d28504’ has good knowledge of area ‘UI’ and ‘Feature’ (shown as by dark shades of gray for the two columns) and hence is a best fit for the project.

Fig. 3. Bertin’s Hotel Plot. Horizontal axis shows the ten most popular components (areas) arranged chronologically in the non-increasing order of number of bugs reported in the area. Vertical axis is a list of 20 bug owners arranged in the non-increasing order of Specialization and Breadth Index (SBI) score for the year 2010-11. Contribution of bug owner in an area is measured relatively (w.r.t other bug owners working in the same area). Score scale is color coded. 100% imply distinguished contribution in an area and vice-versa. Shaded upper triangular matrix represent bug owners with breadth of knowledge and vice-versa.
Fig. 4. Dart Chart: Circle is divided into 50 sectors. Each sector is a bug owner arranged chronologically in the non-decreasing order (counter-clockwise) of Priority Weighted Fixed Issues (PWFI) score in the year 2010-11 (second half). Each concentric circle (outside to inside) is categorical classification of performance as ‘Excellent’, ‘Very Good’, ‘Good’, ‘Satisfactory’, or ‘Not-Satisfactory’. This classification is based on quantitative score (PWFI score) [‘Excellent’: Score >0.5, ‘Very Good’: Score >0.25 and <0.5, ‘Good’: Score >0.125 and <0.25, ‘Satisfactory’: Score >0.0625 and <0.125, ‘Not-Satisfactory’: Score <0.0625]. Color of each concentric circle indicates total number of bug owners (as shown in legend ‘count’) that falls in a given performance category. ‘Legend’ shows performance in four halves over two consecutive years (2009-11).

C. Dart Chart

Dart chart is radial equivalent of bullet chart. The name dart chart come from its structural and conceptual resemblance to dart board. Bullet chart establishes a qualitative equivalent of performance from a quantitative measure of contribution score. The choice of classification scale is a function of performance w.r.t. other developers or specific organization’s expectations. Another aspect attached to this chart is performance relative to self [20]. Figure 4 is a visual plot of Priority Weighted Fixed Index (PWFI) score (refer to Table II) for the role of bug owner.

Understanding relationship between score and its interpretation is a crucial task for organizations. However, such demarcations are not intuitive and are subjected to individual’s perception. The aim of this plot is to justify performance ranking (classification) and ensure fairness and uniformity. Based on PWFI score performance is broadly classified as ‘Excellent’, ‘Very Good’, ‘Good’, ‘Satisfactory’, and ‘Not-Satisfactory’ using min-max normalization of contribution of all bug owners for four halves in two consecutive years 2009-11. Other normalization techniques can be applied to meet specific needs of organization. A detailed description of classification criteria is present in caption of Figure 4. Following are few interpretations that a performance appraiser looks for:

1) Which bug owners are high performers relative to the team? In Figure 4 we see color coded concentric circles. Light color of innermost circle shows that majority of bug owners deliver ‘Not-Satisfactory’ performance. Only two bug owners ‘d00184’ and ‘d14019’ achieve ‘Excellent’ performance for their outstanding contribution in second half of 2010-11 (as shown in green colored dots on outermost circle).

2) How consistent is the performance of bug owner? In Figure 4 we observe three overlapping dots for bug owner ‘d00184’ in the outermost circle. These three dots are for three halves: Green for second half of 2010-11, Red for first half of 2010-11 and Blue for second half of 2009-10. The yellow dot for bug owner ‘d00184’ lies in innermost circle (i.e. ‘Not-Satisfactory’ performance for the first half of 2009-10). Thus bug owner ‘d00184’ starts with ‘Not-Satisfactory’ performance and shows marked improvement by consistently performing ‘Excellent’ for the next three halves. The positions of dots for bug owner ‘d19551’ in Figure 4 shows periodic fluctuations in performance. Bug owner ‘d19551’ perform well in first half of the two consecutive years, while in the second half performance observe a steep fall. Similarly, bug owners with marked improvement, slight improvement, consistently average, steep decline etc. can be observed.
D. Hybrid Plot

Hybrid plot is a combination of simple plots to display rich and otherwise complex information. Adjacent plots share axis to link different pieces of information and present it as whole [21]. In Hybrid plot, we study Deviation from Median Time To Repair metric (refer to Table II) for the role of bug owner. For a bug owner, one of the Key Performance Indicator is fixing bug report in/on time. It is a measure of bug owner’s management skills (time management skills in particular). The expected time to fix a bug report varies across priorities i.e. a high priority bug report must be fixed prior to a low priority bug report.

Figure 5 consolidate twelve plots of two types: back-to-back bar chart and simple bar chart. The arrangement of graphs is such that adjacent plots share axis and present same information. Vertical axis of the plot has two level hierarchy of information. At level one, information is presented for all priorities i.e. priority 0, 1, 2 and 3 (according to Google Chromium Issue Tracking System). For each priority, we analyze the contribution of bug owners arranged in the non-increasing order of total number of bugs owned. A detailed description of Hybrid Plot is available in caption of Figure 5.

In Figure 5 we study ‘Security’ bug reports ‘Closed’ by bug owners during 2010-11 (first half). The time to fix is the minimum time between bug entitled ‘Start’ or ‘Assigned’ status and ‘Fixed’ or ‘Verified’ status. Cases in which multiple bug owner reassignments occur are not considered. Also the choice of Expected Time To Repair for a given priority can be organization specific. Following are the inferences for decision maker:

1) Which bug owners have expertise in solving a given priority ‘Security’ bug report? In Figure 5 we see that for each priority bug owners are arranged (top to bottom) in the non-increasing order of value of back-to-back bar chart. The value of back-to-back bar chart shows the total number of bug reports owned with a given priority. Red bar in back-to-back bar chart measures the number of bug reports with positive contribution and vice-versa. In Priority P0 panel of Hybrid plot, we see that bug owner ‘d05167’ own maximum number of bug reports and thus has expertise in solving priority P0 bug reports. Similarly, we can derive for other priorities.

2) Which bug owner deliver good management skills? In Priority P0 panel, we observe that ‘Positive’ bar chart has maximum value for bug owner ‘d31174’. However this maximum value is for one bug report (as shown in blue color bar of back-to-back bar chart). Thus bug owner ‘d31174’ show good management skills. However, one bug report does not establish the desired confidence in result. Combining these two factors, we infer that bug owner ‘d05167’ has good management skills (worked on four bug reports and ‘Fixed’ them 289 hours before expected time). Similarly we can measure bad management skills for bug owner accounting for the two parameters mentioned above.

Fig. 5. Hybrid Plot: Vertical axis is divided into four sections based on priority of bug reports as P0, P1, P2 and P3 (P1 is highest; P3 is lowest). Each section on the vertical axis is a list of bug owners arranged in the non-increasing order of total number of security bug reports owned with a given priority. Horizontal axis measures three different scores. ‘Value’ is count of bug reports owned by a bug owner for a given priority. Red and blue ‘Value’ is the count of bug reports where bug owner exceeded/preceded the time specified respectively. Similarly, ‘Negative’ and ‘Positive’ shows the time exceeded/preceded in hours to fix the bug report (w.r.t. expected time) respectively.
Fig. 6. Nightingale Rose Plot: Each sector of the circle is month of a year. Two radial plots are drawn presenting data for two consecutive years (Regime: Year 1 for 2009 and Regime: Year 2 for 2010). Bug owners are represented by different colors as shown in ‘Legend’. Area of wedge shows the performance of bug owner (for a given month of the year) where radius of wedge is square root of Priority Weighted Fixed Issue (PWFI) score. Bug owners are arranged (inside to outside) in the non-decreasing order of their PWFI Score.

E. Nightingale Rose Plot

Nightingale Rose plot (also known as coxcomb) is a radial plot. It analyzes the gap between actual value and corresponding reference expected value. Nightingale Rose plot display contrast by analyzing data for two consecutive years. However to ensure comparisons and to analyze sophisticated interactions between causative factors the period i.e. time interval between two consecutive observation points must be uniform [22][23]. For this visualization technique, we study Priority Weighted Fixed Issue metric (refer to Table II) to gain detailed insights on performance trends.

Figure 6 is a visual activity track of 10 bug owners for two consecutive years (2009-10 and 2010-11). The selection of these bug owners is based on their PWFI score (detailed description in caption of Figure 6).

The inferential power of Nightingale Rose plot can be harnessed by capturing external environmental factors. In the absence of this data for this work, the inferences stated below are exemplary and must be put into right perspective for analysis by organizations.

1) **Which bug owners are environmental susceptible? or Which bug owners can act as savior for the team?**

We see in Figure 6 a steep decline in quality contribution in the month of December (w.r.t. neighboring months November and January) for years 2009-10 and 2010-11. One possible justification is reduced activity during times of Christmas vacations. However, we observe that bug owner ‘d05167’ contribute significantly in time of general inactivity.

2) **Analyze performance variations within/across years?**

In Figure 6 we see that quality contribution of bug owner ‘d07873’ was overshadowed (wedge disappear) by significant large contributions by other bug owners. However, for the year 2010 we observe that wedge corresponding to bug owner ‘d07873’ appears for all months of the year. It shows quality contribution delivered significantly.

3) **Which bug owner’s contribution dominate in a given time period?**

Assume organization has product release scheduled in April 2009. In times of peak workload we observe that bug owner ‘d13741’ and ‘d00184’ overshadow the contribution of other bug owners. These bug owners deliver quality contribution to ensure timely release and are asset to organization.

Nightingale Rose plot find multitude of applications based on the environmental factor available for analysis.
V. Threats to Validity and Limitations

In this work, we present 6 visualization techniques. To capture a 360 degree view of performance appraisal more visualization techniques (build on diverse role-based contribution and performance assessment metrics) must be devised. Following are the limitations of the work:

1) Limited developer set: In Table III we see that the 6 visualization techniques proposed in ‘SamikshaViz’ analyze a limited set of developers (count of developers calculated approximately). Beyond this limit, visualization seems cluttered and fail to present the intuition for the plot.

2) Fixed data dimensionality: The number of data dimensions that can be analyzed with a visualization technique are fixed. It cannot analyze complex relationships in a higher dimensional dataset (beyond the limit specified for each visualization in Table III.

VI. Discussion and Future Work

We present a visualization framework to analyze contribution and performance of software maintenance professionals by mining bug archives. Motivated by the need to aid decision makers distinguish performance, we propose 6 visualization techniques. Each visualization present a panoramic view of developer’s contribution in a team and analyze performance appraisal with different perspective and goals. These visualizations justify the rationale of metrics proposed in ‘Samiksha’ and give a deeper intuition into contribution patterns of developers.

We implement the proposed visualization on real-world dataset of Google Chromium Issue Tracking System. Experimental results demonstrate the inferential ability of these visualization techniques. These results are validated by survey responses from practitioners in industry. Thus we conclude that visualizations proposed in ‘SamikshaViz’ provide evidence-based and data-driven insights to decision makers. It justify and validate contribution and performance assessment of Software Maintenance Professionals.

Following are future research directions of ‘SamikshaViz’ framework:

1) Build an interactive toolkit to visualize performance.
2) Use inferences from multiple plots to reinforce and interpret complex concepts.
3) Compare inter team, project and organization performance. Also learn and apply key ideas from successful teams.
4) Study evolution of organization from visual framework to facilitate policy framers in strategic decision making.

Acknowledgment

The work present in this paper is supported by TCS Research Fellowship for PhD students awarded to the first author. The author would like to acknowledge Dr. Pamela Bhattacharya (external advisor), Ravindra Naik (industry mentor from TCS) and survey respondents for providing their valuable inputs.