Metrics visualization technique based on the origins and function layers for OSS-based development

Ryosuke Ishizue, Hironori Washizaki
and Yoshiaki Fukazawa
Department of Computer Science and Engineering
Waseda University Tokyo, Japan
Email: ishizue@ruri.waseda.jp,
{washizaki, fukazawa}@waseda.jp

Sakae Inoue, Yoshiiku Hanai,
Masanobu Kanazawa and Katsushi Namba
Fujitsu Connected Technologies Limited
Kanagawa, Japan
Email: {inoue.sakae, hanai.yoshiiku,
kanazawa.masano, nanba}@jp.fujitsu.com

***Abstract*—OSS (Open Source Software)-based software developments tend to have a lot of defects when editing program source code files that other organizations created. Developments with complex origins and functional layers are increasing in OSS-based development. As an example, here we focus on an Android smart phone development project and propose new visualization techniques for product metrics based on the file origin and functional layers. One is the Metrics Area Figure, which can express duplication of edits by multiple organizations intuitively using overlapping figures. The other is Origin City, which was inspired by Code City. It can represent the scale and other measurements, while simultaneously stacking functional layers as 3D buildings. The contributions of our paper are to propose new techniques, implement them as web applications, and share the results of our questionnaire experiment. Our proposed techniques are useful not only to visualize measured metrics, but also to improve product quality.**

Keywords—metrics; origin; functional layer;

I. INTRODUCTION

Herein we focus on OSS (Open Source Software)-based software developments because their origins and functional layers of are becoming more complex. For example, Android smart phone developments often incur problems because the base of Android is provided by Platformer as an OSS, which some developers then use to create their own products. In a development involving multiple organizations, it is important to focus on which organization created each program source code file as files edited by multiple organizations tend to have more defects than ones edited by a single organization [1]. Additionally, it is important to know which functional layer the file belongs (e.g., Kernel, Driver, Framework, Application, etc.).

A previous study defined the *origin* as its creation and modification history [1]. If the functional layers have different origins, then development becomes more challenging. Hence, knowing the origin of a file and its functional layers is useful to avoid defects. Moreover, this information leads to improved product quality as product metrics are based on the origin and functional layers.

As a motivating example, our project involving Android smart phone development has three development organizations: Platformer, Chipset Vendor, and Fujitsu Connected Technologies (Fig. 1). For example, O_1, O_2, and O_3 mean a file was created by Platformer, Chipset Vendor, and Fujitsu Connected Technologies, respectively, whereas O_{12} means a file was created by Platformer and edited by Chipset Vendor. We use the seven functional layers known as Android Architecture: (1) Linux Kernel, (2) Library, (3) Android Runtime, (4) Library (external OSS), (5) Application Framework, (6) Applications and (7) Others.

When reviewing software, the metrics of each file are extremely difficult to understand using tables, bar charts, bar graphs, or other primitive methods, making it hard to analyze the origin and functional layer. For example, it is easy to find the origin of the largest metric with a simple table, but it is challenging to determine the percentage of the total value of the organization (e.g., Total of Platformer=O_1+O_{12}+O_{13}+O_{123}). In another example Chipset Vendor edited the files that originated from Platformer. Moreover, adding a functional layer further complicates the origin. To help address these issues, we propose a visualization method called the *Metrics Area Figure (MAF)*, which shows the measured metrics of each origin using overlapping circles, rectangles, etc.

Many previous studies have visualized metrics as 2D or 3D objects. Some even used 3D software visualization as 3D visualizations can provide more information than 2D visualizations. An especially famous metrics visualization technique is *Code City*, which represents the scale and other measurements as 3D buildings in a city [2][3]. Inspired by Code City, we also propose a 3D visualization method called *Origin City (OC)*, which shows the measured metrics of each origin as well as shows the functional layers as color-coded stacks.

The contributions of our paper are that we propose : (1) two new visualization techniques, which are useful to improve the product quality, (2) MAF, which is useful to indicate the origins in a development, and (3) OC, which is useful to show the functional layers in a development.

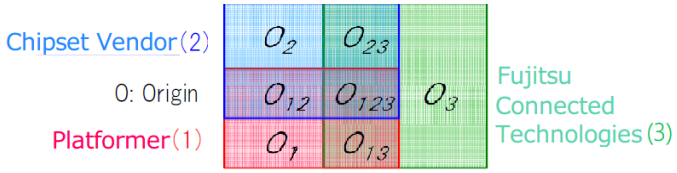


Fig 1: Origins in our Android smart phone project

II. VISUALIZATION METHOD

We implemented two visualization techniques with canvas of HTML5 and JavaScript. Figure 2 overviews the system. First, a user prepares the measurement result as a csv file, which contains measured metrics (e.g., defects, number of public methods/fields, global variable, etc.). For visualization, these metrics must be measurable by a static analysis tool such as *Understand* as well as contain a functional layer that can be classified by the file path (e.g., kernel/*.* is layer of kernel) and the origin.

To determine the origin of each file, as many directories as the number of organizations are prepared. The files of each organization are placed into each directory. The *diff* command is used to find the edited and added files. Then the target file is chosen, and the metrics are measured to visualize the selection form of the visualization tool. Finally, the results of the visualized data are shown upon pushing the exec button.

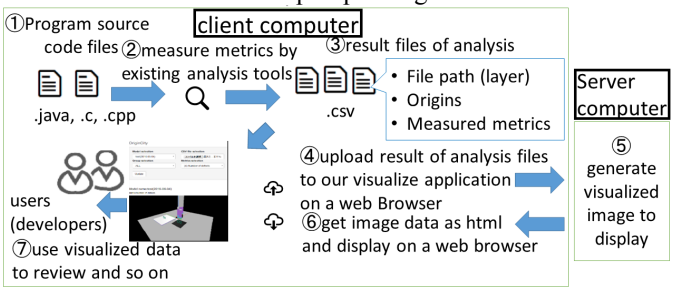
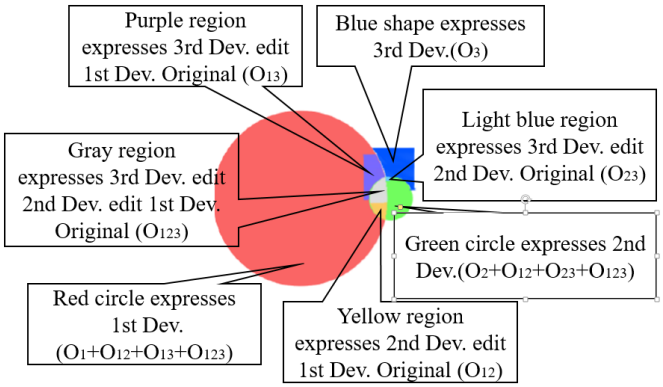


Fig 2: Overview of visualization



Dev. : development organization

Fig 3 : Metrics Area Figure

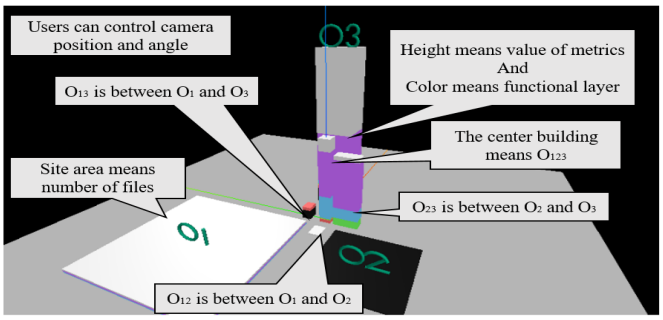


Fig 4 : Origin City

The procedure to generate MAF and OC with N organization is as follows. First, there are 2^N-1 regions or buildings in MAF or OC. Each region or building corresponds to the origins represented by $O_{x_1 \cdots x_n \cdots x_N}$, where N is the total number of organizations and n represents a specific organization. The value of x_n = 0 or n, and is ignored when x_n=0. For example, if there are five organizations, O_{124} is defined as

N=5: $O_{x_1 x_2 x_3 x_4 x_5}$ $\{x_1=1, x_2=2, x_3=0, x_4=4, x_5=0\} = O_{124}$

A. Metrics Area Figure

MAF expresses the duplication of edits by multiple organizations intuitively with overlapping figures. To generate MAF of N organizations, a circle, which has an area equal to the sum of the measured metrics of origins for x_1=1, is initially painted. Then another circle, which has an area equal to the sum of the measured metrics of the origins for x_2=2, is drawn. The overlapping area of the two circles must equal the sum of the measured metrics of the origins for x_1=1 and x_2=2. Next, if 3 ≤ N, the following process is repeated for n ≤ N.

1. Choose an intersecting point, which is surrounded by most of the regions. The surrounding regions can be up to $O_{x_1 \cdots x_{(n-1)}}$ (e.g., n=3, O_1, O_2, O_{12} and outside of figure).
2. From the intersecting point toward each region whose origin is $O_{x_1 \cdots x_{(n-1)}}$, new regions, which have areas equal to the sum of the measured metrics of the origins for x equal to the origin of base region plus x_n = n.(e.g., n=3, O_{13}, O_{23}, O_{123} and O_3), are painted.
3. When all regions, which total 2^n-1, are painted, n is incremented.

As a result of repeating the above process, MAF of N organizations is generated

Figure 3 shows an example of MAF for three organizations. The red circle denotes the sum of the measured metrics of the files created by the 1st Dev. organization (O_1). The green circle means the sum of the measured metrics of files created and edited by the 2nd Dev. (O_2), while the blue shape represents the measured metrics of the files created and edited by the 3rd Dev. (O_3). Note that the blue shape is not a circle because it is mathematically impossible

To generate this figure, first a red circle is painted. Then a green circle is developed. The overlapping area of the two circles is painted in yellow. Next, from one of the intersecting points, a blue shape is painted outward. Finally, a purple shape, a light blue shape (O_{23}), and a gray shape (O_{123}) are painted toward the each region.

B. Origin City

OC represents the scale, the measured metrics, and the stack 3D buildings simultaneously. OC is depicted by 2^N-1 buildings in 3D space. Each building corresponds to an origin metric. Additionally, a building's colors denote the ratio of the measured metrics values of the functional layers.

The most important thing is the building position. First, N buildings whose origins are O_1, $O_{2,..}$, O_n are placed on a concentric circle. After that, buildings corresponding to $O_{x_a x_b \cdots x_n}$ ($1 \leqq a<b<n \leqq N$, x_a=a, x_b=b, x_n=n) are placed on the

center of gravity of the figure connecting buildings with x_a=a, x_b=b, x_n=n. For example, O_{12} is placed in the middle of the line connecting O_1 and O_2. O_{123} is placed at the center of gravity of a triangle connecting O_1, O_2, and O_3. O_{1234} is placed on the gravity of a rectangle connecting O_1, O_2, O_3, and O_4. After all the positions are determined, the buildings are drawn from the center. The radius from the center of the buildings is decided to avoid intersecting each other toward the outside. By repeating the above process, OC of N organizations is generated.

Figure 4 shows example of OC with three organizations. The center building corresponds to O_{123}, and it is surrounded by the buildings of O_1+O_2+O_3. The O_{12} building is between the O_1 and the O_2 buildings. The building of O_{13} and O_{23} are similarly placed.

III. Experiment

We verified the effectiveness of our techniques using Android smart phone projects.

A. Target and Visualization Results

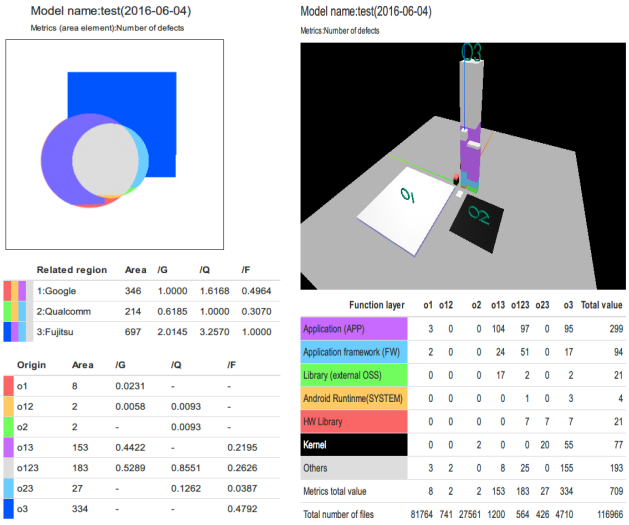
We speculated that our techniques are useful for visualizing a situation at a particular stage of development or comparing stages as a development process progresses. In this experiment, there are three organizations. 1st Dev. is Platformer, 2nd Dev. is Chipset Vendor, and 3rd Dev. is Fujitsu Connected Technologies.

Figure 5 (left) is an example of MAF, which shows the number of defects. The big blue region indicates that the number of defects in the files created by Fujitsu Connected Technologies is large. Moreover, the red and green circles almost completely overlap, demonstrating that Fujitsu Connected Technologies edited many defects. Such analysis makes it easy to intuitively grasp the origins with many defects. Consequently, developers know to carefully edit such files.

Figure 6 shows additional examples of MAF. They show the results of visualizing a series of data for an Android smart phone project involving the lines of code (LOC). The right example visualizes newer data than the left. The green circle in the right is larger than that in the left because the Android version is updated in the right. Moreover, the overlapping area between Platformer and Chipset Vendor is smaller in the right than the left, but the size of the red circle is about the same. Thus, MAFs help developers review previous projects.

Figure 5 (right) provides an example of OC where the number of defects is given as measured metrics. Platformer (O_1) and Chipset Vendor (O_2) created many more files (site areas) than Fujitsu Connected Technologies. Moreover, the site area of O_{123} the same as O_{13}, but its height is much higher, indicating that the ratio of defects is greater in O_{123} than in O_{13}. The building color of O_{13} is almost black and red, whereas that of O_{123} is almost purple and blue, indicating that the defects in O_{13} are in low layers (Kernel and HW Library) whereas those in O_{123} are in high layers (APP and FW).

By such an analysis, it is easier to grasp intuitively the origin and functional layer with many defects. As a result, developers are aware to carefully edit such files.



Related region	Area	/G	/Q	/F
1:Google	346	1.0000	1.6168	0.4964
2:Qualcomm	214	0.6185	1.0000	0.3070
3:Fujitsu	697	2.0145	3.2570	1.0000

Origin	Area	/G	/Q	/F
o1	8	0.0231	-	-
o12	2	0.0058	0.0093	-
o2	2	-	0.0093	-
o13	153	0.4422	-	0.2195
o123	183	0.5289	0.8551	0.2626
o23	27	-	0.1262	0.0387
o3	334	-	-	0.4792

Function layer	o1	o12	o2	o13	o123	o23	o3	Total value
Application (APP)	3	0	0	104	97	0	95	299
Application framework (FW)	2	0	0	24	51	0	17	94
Library (external OSS)	0	0	0	17	2	0	2	21
Android Runtinme(SYSTEM)	0	0	0	0	1	0	3	4
HW Library	0	0	0	0	7	7	7	21
Kernel	0	0	2	0	0	20	55	77
Others	3	2	0	8	25	0	155	193
Metrics total value	8	2	2	153	183	27	334	709
Total number of files	81764	741	27561	1200	564	426	4710	116966

Fig 5 : Example of MAF (left) and OC (right)

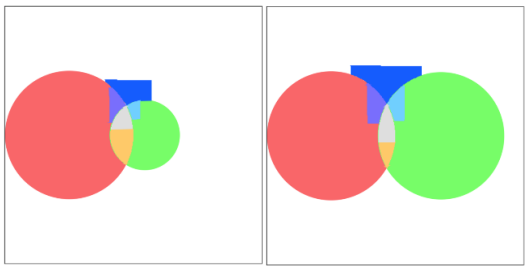


Fig 6 : Example of MAF in an old (left) and new (right) model

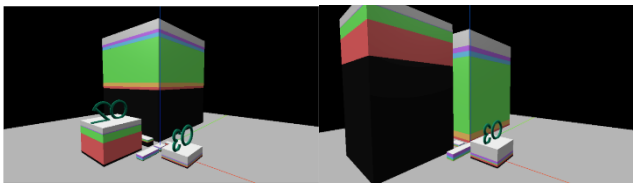


Fig 7 : Example of OC in an old (left) and new (right) model

Figure 7 shows additional examples of OC using the same data as Fig. 6. The building of O_2 on the right is larger and higher than that on the left because the Android version on the right has been updated. Moreover, the black layer of O_1 on the right almost disappears compared to the large black layer of O_2 on the left. Consequently, OC can help developers review and analyze previous projects.

It is much easier for developers to analyze the measured metrics based on the origin and the functional layer with MAF and OC. MAF solves problems that are difficult to grasp using the measured metrics of an origin that includes multiple organizations due to overlapping figures. Similarly, the stacks of colored layers in OC address the issue of complex origins and functional layers.

B. Experimental Setting for Usefulness

We asked developers about the usefulness of our visualization techniques for Android smart phone projects. We speculated that both visualization techniques are useful for senior development personnel like project managers and team leaders to review previous projects and to compare the progress

of a current project. We conducted an experiment to verify this hypothesis.

This work investigated the following research questions:

RQ1 Is MAF useful to determine the origins in a development?
RQ2 Is OC useful to determine the functional layers in a development?
RQ3 Are our new visualization techniques useful to improve product quality?
RQ4 What is the purpose of our new visualization techniques?
RQ5 Who would find our visualization techniques useful?

To answer these research questions, we implemented a questionnaire about our new visualization techniques.

Table 1 shows the 13 questions about the awareness of metrics, origins, and functional layers as well as the 4 questions about which method provides the best visualization. Q1 to Q13 are evaluated on a four-level scale: Very much, Somewhat, A little, and Not at all. Q14 to Q17 are also answered on a four-point scale: Table, Pie/bar Chart, MAF, and OC.

We implemented a questionnaire, and 18 people belonging to an Android smart phone development project completed it. They were managers, leaders, and various other project members (designer, reviewer, tester, programmer, etc.).

TABLE I. QUESTIONS

Q	Question sentence
1	Are you usually aware of the origin?
2	Are you usually aware of the metrics?
3	Are you usually aware of the metrics for each origin?
4	Are you usually aware of the metrics for each functional layer?
5	Do you feel that MAF is useful for awareness of the origin?
6	Do you feel that MAF is useful for awareness of the metrics?
7	Do you feel that MAF is useful for awareness of the metrics for each origin?
8	Do you feel that OC is useful for awareness of the metrics?
9	Do you feel that OC is useful for awareness of the metrics of each functional layer?
10	Do you feel that MAF is useful to improve product quality?
11	Do you feel that OC is useful to improve product quality?
12	Do you feel that MAF is useful for your work?
13	Do you feel that OC is useful for your work?
14	Which method is the best to grasp the detailed value?
15	Which method is the best to grasp the trend of the value?
16	Which method is the best to grasp the value by functional layer?
17	Which method is the best to grasp the value by origin?

"Q" means "Question number". Including the unanswered

C. *Experimental Results of Usefulness*

Figure 8 (left) and Table 2 summarize the results. Figure 8 (right) and Figure 9 show the results by the role of the participants.

RQ1: Q5 to Q7 indicate that about 70-80% of the respondents feel our new visualization techniques are useful for improving the awareness of metrics based on the origin. As expected, most feel that MAF is useful to understand the origin (Q7).

RQ2: Q4 shows that less than 50% of developers are aware of the metrics and the functional layers. However, Q8 to Q9 show that about 70-80% of the people feel our new visualization techniques are useful for improving the awareness of the functional layers.

RQ3: Q10 and Q11 confirm that more than half of the participants feel that both methods are useful to improve product quality.

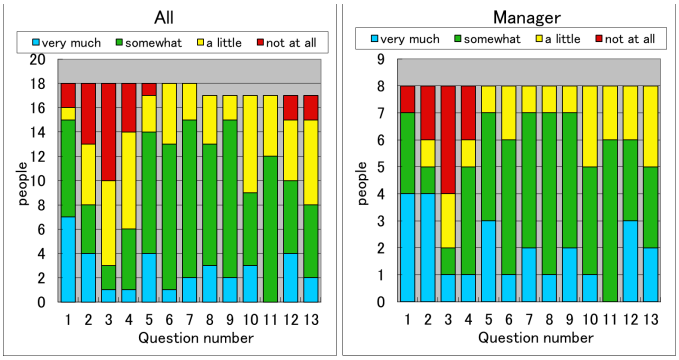


Fig 8 : All responses to the questionnaire (left), Managers' responses to the questionnaire (right)

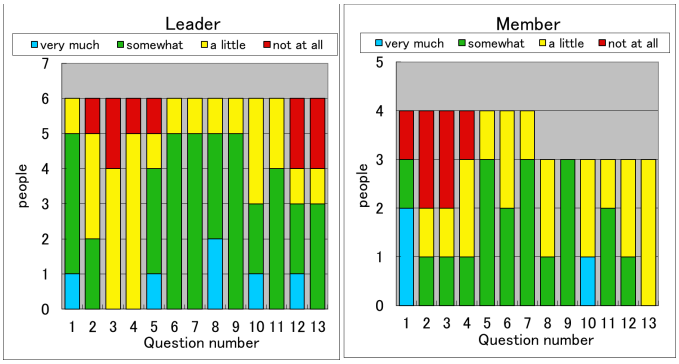


Fig 9 : Leaders' responses to the questionnaire (left), Members' responses to the questionnaire (right)

TABLE II. RESULTS OF THE PURPOSE OF THE EXPRESSION METHOD

Q	Answerer [people]			
	Table	*pie・bar chart*	*MAF*	*OC*
14	11	5	0	0
15	1	5	7	3
16	1	2	3	10
17	1	1	6	6

"Q" means "Question number", and includes unanswered questions.

RQ4: According to Table 2, tables, pie and bar charts are more useful than our method to grasp the detailed value (Q14), but more people think that MAF is the best method to grasp the trend of a value (Q15). Moreover, as expected, 63% participants feel that OC is better to grasp the value by functional layer (Q16). Surprisingly, OC is as good as MAF to grasp the values of the origins (Q17). These results indicate that our proposed techniques are very effective for visualizing the measured metrics based on the origin and functional layers.

RQ5: Figure 8 (right) and Figure 9 show the questionnaire results by position of the participant. Managers have more positive answers than leaders, while leaders have slightly more positive responses than members. Thus, it can be inferred that our methods are more useful for management because managers generally have more management tasks than leaders, while leaders have more tasks than members.

D. Threats to Validity

The questionnaire was carried out after a demonstration for developers. Therefore, the results may change after developers actually use our visualization tool in their development works. This is a threat to the internal validity.

These results are from one group in one company. Some of participants were already familiar with metrics, origins and functional layers. If we repeated this experiments with another group or organization unfamiliar with these concepts, the results may differ. This is a threat to the external validity.

IV. RELATED WORK

In this section, we describe related works about origins, Code City, and metrics visualization software in general.

S. Sato et al. looked at the effects of organizational changes on product metrics and defects [1]. They analyzed an open source projects to investigate the relationships between the file's creation and modification history and metrics. Then they defined a file's *origin* as its creation and modification history. Our proposed visualization techniques, especially MAF, are based on this concept.

Richard Wettel and Michele Lanza presented a 3D visualization approach, which gravitates around the city metaphor [2][3]. Their goal was to give the viewer a sense of locality to ease program comprehension. OC is inspired by Code City. Buildings or city-lots in Code City represent the context of software, such as packages and classes. The building height expresses its measured metrics. In contrast, buildings in OC represent the origins. Moreover, the building is composed of a color-coded stack, which denotes the functional layers.

There are many studies on the visualization of software. P. Caserta et al. surveyed 2D and 3D visualization techniques based on statistical aspects of software and its evolution [4]. 3D software visualization like a city has a decent history. C. Knight et al. proposed *Software World* to create graphical abstractions of Java source code, which was the starting point of 3D software visualization [5]. J. I. Maletic et al. visualized UML class diagram in the form of layered 3D objects [6]. Similarly, OC is based on layered objects in 3D; however, it is dedicated to the visualization of origins. T. Panas et al. presented *3D City*, which uses a city metaphor for software maintenance costs. As a result, their visualization technique expresses dangerous source code as an old or collapsed building [7]. Such methods can be applied to our techniques for additional expressions to represent the severity of defects. Although we implemented our tool as a web application, F. Fittkau et al. presented *ExplorViz*, which models live program traces with a web-application utilizing WebGL [8]. K. Kobayashi et al. proposed *SarF Map* to visualize features and layers [9]. G. Langelier et al. proposed a metrics visualization technique named VERSO in which the graphical properties of buildings are height, color, and angle [10]. The idea of using colors to represent properties is similar to our approaches; if we need to represent more properties at the same time, we could also use angles. Similar to MAF, H. Byelas et al. developed the *area of interest*, which expresses UML elements of shared properties by overlapping figures, which are colored like a contour diagram [11][12].

V. CONCLUSIONS

We propose two visualization techniques: MAF and OC. The former helps grasp the measured metric of each origin, while the latter helps grasp the measured metrics of each functional layer. These new methods are more efficient than primitive methods, such as tables, bar charts, and pie charts. Additionally, many developers indicated that the proposed method improve the product quality.

In the future, we intend to investigate whether these methods can improve product quality by a continuous experiment. We also plan to refine our visualization tool. For example, our visualize application currently supports the function layer for the Android architecture only, but we would like to support additional architectures. Furthermore, as we improve and introduce more visualization methods, we would like to create a tool that collects all information about quality management like a dashboard.

REFERENCES

[1] S. Sato, H. Washizaki, Y. Fukazawa, S. Inoue, H. Ono, Y. Hanai, & M. Yamamoto, "Effects of organizational changes on product metrics and defects," APSEC 2013, 2013.

[2] R. Wettel and M. Lanza, "Visualizing Software Systems as Cities," VISSOFT 2007, 2007.

[3] R. Wettel, M. Lanza, and R. Robbes, "Software systems as cities: a controlled experiment," ICSE 2011, 2011.

[4] P. Caserta, O. Zendra, "Visualization of the Static Aspects of Software: A Survey," Visualization and Computer Graphics, IEEE Transactions on, On page(s): 913 - 933 Volume: 17, Issue: 7, 2011

[5] C. Knight and M. Munro, "Virtual but visible software," IEEE Int. Conf. on Info. Visualisation, 2000.

[6] J. I. Maletic, J. Leigh, and A. Marcus, "Visualizing software in an immersive virtual reality environment," ICSE 2001, 2001.

[7] T. Panas, R. Berrigan, and J. Grundy, "A 3D metaphor for software production visualization," IEEE Int. Conf. on Info. Visualisation, 2000.

[8] F. Fittkau, J. Waller, C. Wulf, and W. Hasselbring, "Live trace visualization for comprehending large software landscapes: The ExplorViz approach," VISSOFT 2013, 2013.

[9] K. Kobayashi, M. Kamimura, K. Yano, K. Kato, & A. Matsuo, "SArF map: Visualizing software architecture from feature and layer viewpoints," ICPC 2013, 2013.

[10] G. Langelier, H. Sahraoui, and P. Poulin, "Visualization-Based Analysis of Quality for Large-Scale Software Systems,"ASE 2005, 2005.

[11] H. Byelas and A. Telea, "Visualization of Areas of Interest in Software Architecture Diagrams," Symp. Software Visualization, 2006.

[12] H. Byelas and A. Telea, "Visualizing Metrics on Areas of Interest in Software Architecture Diagrams," Proc. Pacific Visualization Symp., 2009.