SUGISAKA MASANORI

Journal of Robotics, Networking and Artificial Life
Vol. 9(2); September (2022), pp. 115–120
ISSN (Online):2352-6386; ISSN (Print):2405-9021
https://alife-robotics.org/jrnal.html

Research Article

A Proposal of a Software Defect Predication System "FaRSeT-#" for Exploratory Testing

Yoshihiro Kita[1], Kazuki Ueda[2], Kiyotaka Sakurai[2]
[1]*Faculty of Information Systems, University of Nagasaki, 1-1-1 Manabino, Nagayo, Nishisonogi , Nagasaki 851-2195, Japan*
[2]*Nihon knowledge Co.,Ltd. 3-19-5 Kotobuki, Taito, Tokyo 111-0042, Japan*

ARTICLE INFO

Article History

Received 02 November 2021
Accepted 05 May 2022

Keywords

Software testing
Exploratory testing
Defect predication
Self-organizing map (SOM)

ABSTRACT

The goal of software testing is to detect all latent defects. However, it is difficult to know how many latent defects remain and where they are hidden. In this research, we propose a system that analyzes the characteristics and tendencies of already detected defects and predicts where the latent defects are likely to be. Specifically, the system inputs the data of detected defects into a Self-Organizing Map (SOM) and predicts the locations that contain many defects from this map. To confirm the validity of this proposal, we input past defect data into the SOM, analyze the trend of defects, and evaluate the predictability of the latent defects..

1. Introduction

The goal of software testing is to detect all latent defects, but it is impossible to achieve that goal[1]. However, residual defects are causes of serious lost, so it is desirable to remove defects as much as possible.

Exhaustive or exploratory testing methods are needed to find latent defects in software. Especially in exploratory testing, it is difficult to the location of latent defects while deciding where to search next.

Mr. Ueda as the second author, proposed the testing method "FaRSeT (Flexible and Rapid Software Test)"[2] which is an exploratory testing method that utilizes test analysis by mind mapping to cope with projects have short delivery time, and frequent specification changes. This method reduces the time and effort required to prepare test cases in advance and allows us to prioritize important searching points for testing while obtaining the agreement of stakeholders. However, this method has two problems as follows.

- The searching points are not possible to determine as important, cause no defects occurred in the tests that are not executed.
- It is difficult to assume the remaining defects by the number of defects that are found only.

In this paper, we propose a method "FaRSeT-#" which infer the important searching points, and a defect predication system that incorporates this method. In order to infer the important searching points, we use Self-Organizing Map (SOM)[3], and visualize them on a two-dimensional map.

Corresponding Author's E-mail: kita@sun.ac.jp, ueda@know-net.co.jp, sakurai@know-net.co.jp

Function	Function completeness	Function correctness	Function appropriateness	Co-existence	Interoperability	Appropriateness recognizability	Learnability	Operability	User error protection	User interface aesthetics	Accessibility	Confidentiality
A	10	12	9	0	1	1	0	4	5	2	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0
C	5	6	1	0	0	3	0	3	1	0	0	0
D	4	9	1	0	0	0	0	2	0	1	0	0
E	2	7	2	0	0	0	0	2	0	0	1	0
F	3	8	2	0	0	0	0	1	0	3	0	0
G	2	3	2	2	1	1	0	4	1	1	1	0
H	0	3	0	0	0	1	0	0	1	0	0	0
I	9	23	4	0	2	5	0	6	1	2	0	0
J	1	2	0	0	0	0	0	3	3	0	0	0
K	1	2	0	1	0	0	0	0	0	0	0	0

Fig.1. An example of an exploratory testing matrix

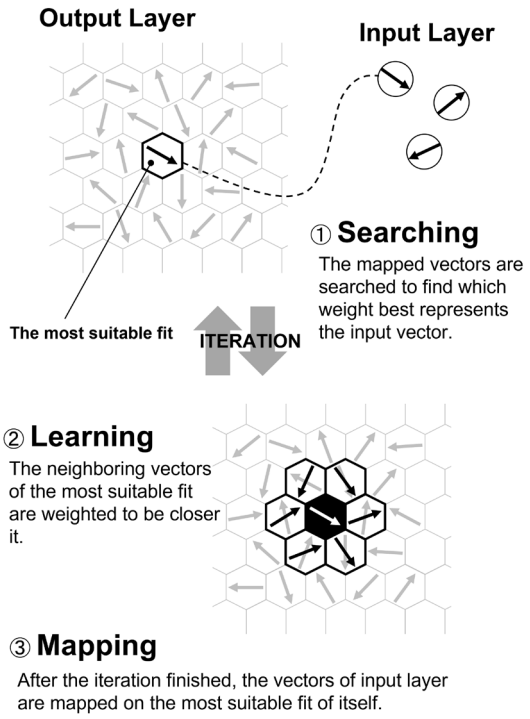


Fig.2. The learning process of SOM

2. Related Works

2.1. *Other tools for supporting the exploratory testing*

PractiTest[4] is a total management tool for testing. This tool manages and visualize the test process includes the exploratory testing. The testers and stakeholders can confirm the results of exploratory testing by the report which is outputted PractiTest.

TESTPAD[5] is a test plan tool likes the spread seat and the check list. This tool manages the test cases that are checked in the past testing and visualize the statistical data of these test cases.

These tools are strong and popular supporting tools for the exploratory testing. However, it is difficult to forecast the next test case for the exploratory testing clearly by these tools.

2.2. *FaRSeT*

FaRSeT (Flexible and Rapid Software Test)[2] is a flexible and rapid testing method that uses a mind-mapped job analysis and an exploratory testing matrix to determine the priority of searching points for testing in a current project. The process of FaRSeT is shown as follows.

(i) Conduct job analysis using a mind map.

(ii) Create the test charters which are broken down from the quality characteristics of software, based on the job analysis in step (i).

(iii) Create the table which is named “the exploratory testing matrix”, with the test charters as the horizontal axis, and the function as the vertical axis.

(iv) Define the intersection as a “session”, of each item in the exploratory testing matrix, and execute the exploratory testing while obtaining the agreement of stakeholders for high importance sessions.

(v) If defects are found in step (iv), the number of defects should be listed in the session of the exploratory testing matrix.

(vi) Repeat step (iv) to step (v) while deciding the next session for the exploratory testing based on the number of defects listed in the session. For example, in Fig.1, the sessions colored in red, are the next session for the exploratory testing.

2.3. *Self-Organizing Map*

Self-Organizing Map (SOM)[3] is an unsupervised competitive neural network. SOM is a machine learning model that represents the similarity of given multidimensional data on a two-dimensional map as shown as Fig. 2.

SOM consists of two layers, an input layer, and an output layer. The input layer contains nodes with input vectors. The number of their nodes is the same as the number of samples of data used as input. The output layer contains nodes which are arranged hexagonal honeycomb, and each node initially holds a random reference vector. The learning process of SOM is shown as follows.

(i) Compare an input vector to all the reference vectors in the output layer.
(ii) The node which has the most similar reference vector to the input vector is defined “The most suitable fitting node”.
(iii) The reference vector of the nodes which are adjoined the most suitable fitting node, is weighted to be close the input vector.
(iv) Iterate the learning process until the learning times is satisfied.
(v) After the iteration finished, the vectors of input layer are mapped on the most suitable fit of itself.

The nodes of input layer are placed close to the nodes that are similar to each other and are placed far from the nodes that are not similar to each other. Therefore, the similarity between the samples is visualized in two-dimensional space as the distance between the nodes of input layer.

3. Software Defect Predication System

In this paper, we propose a method “FaRSeT-#” which infer the important searching points, and a defect predication system that incorporates this method. FaRSeT-# is an extended method of FaRSeT by incorporating SOM. The vectors of all session from the exploratory testing matrix input to SOM. The mapped nodes of session group into colored clusters in SOM. In this way, the sessions belonging to the same cluster are similar, which means that they have the commons: defect tendencies, target areas of search, and effective testing methods. The testers analyze the sessions of each cluster, and predict the priority of searching points, and the location of latent defects. The data to be input to the SOM is the session information consisting of the following items:

- Metrics related to the function
  (e.g., size of the function, importance of the function, skills of the developer, etc.)
- Metrics related to test charter
  (e.g., importance of quality characteristics)
- Metrics related to sessions
  (e.g., number of defects, recent test results, test execution time, skills of testers, etc.)

Our SOM drawing tool can color the background of SOM each input data to SOM. The depth of the color shows the degree of its data. The tester can select the data, if he/she want to color it.

Fig. 3 shows the results of training the exploratory test matrix in Fig. 1 using SOM, mapped to a two-dimensional map. For example, the label "A5" represents the row "Function A" and the column "Test Charter 5 (Interoperability in this case)" in the exploratory test matrix.

The nodes on the SOM are divided into clusters by k-means method[6], and each cluster is color-coded with six colors (RED, BLUE, GREEN, YELLOW, CYAN, and MAGENTA). In Fig.3, the blue area indicates that the scale of the function is large. The upper right area is darker in color, and the YELLOW nodes are gathering in this area. In Fig.4, the green area indicates that the importance of the quality characteristic is high. The lower right area is darker in color, and the BLUE nodes are gathering in this area. The other areas in these maps, (e.g., the center area, upper left area) are lighter in color, and the GREEN, CYAN, and MAGENTA nodes are gathering in these areas. The testers and the stakeholders can analyze what elements affect to the data from the depth of the map background(area) color. And they can decide next search session for exploratory testing from the number or the degree of elements which affect to the data.

we analyze the priority of the clusters from the colors of this map, the priority order is determined as follows

(i) BLUE
(ii) YELLOW
(iii) RED
(iv) CYAN and MAGENTA
(v) GREEN

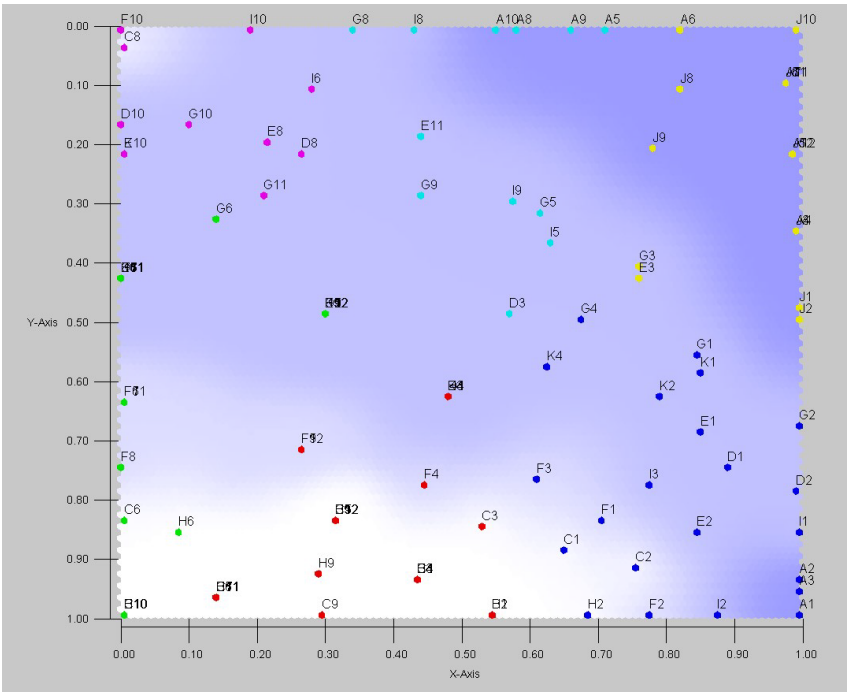


Fig.3. SOM with the session of the exploratory testing matrix in Fig. 1 as input
(The blue area indicates that the scale of the function is large.)

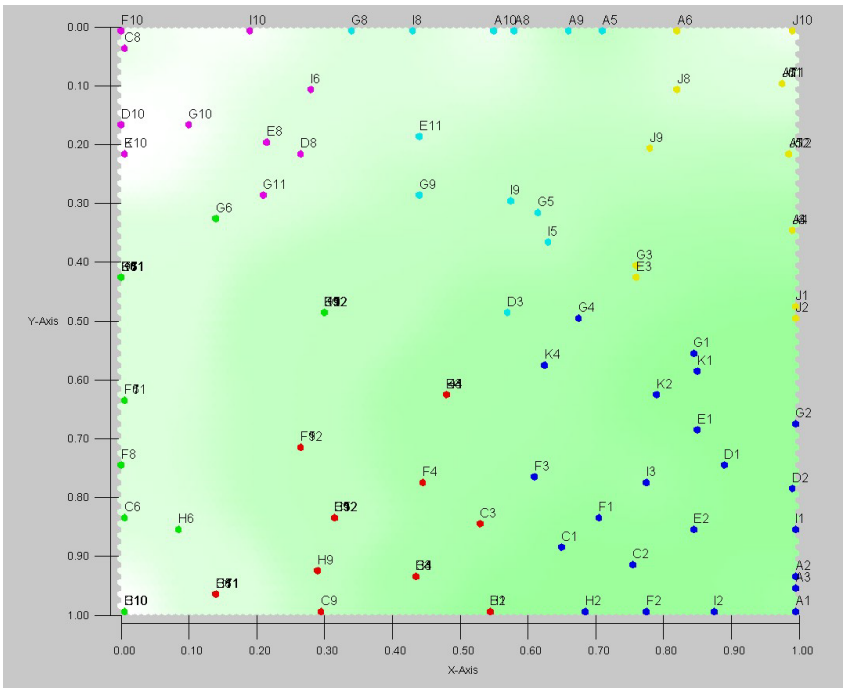
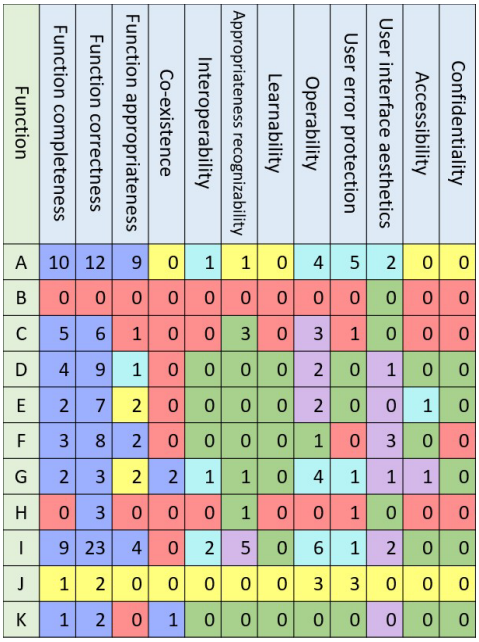
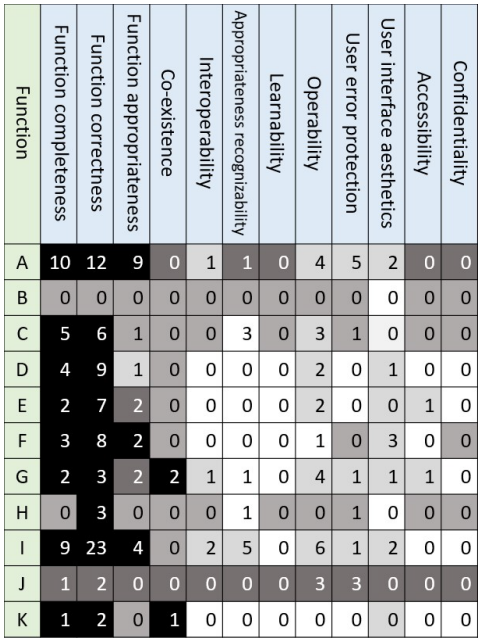


Fig.4. SOM with the session of the exploratory testing matrix in Fig. 1 as input
(The green area indicates that the importance of the quality characteristic is high.)



Function	Function completeness	Function correctness	Function appropriateness	Co-existence	Interoperability	Appropriateness recognizability	Learnability	Operability	User error protection	User interface aesthetics	Accessibility	Confidentiality
A	10	12	9	0	1	1	0	4	5	2	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0
C	5	6	1	0	0	3	0	3	1	0	0	0
D	4	9	1	0	0	0	0	2	0	1	0	0
E	2	7	2	0	0	0	0	2	0	0	1	0
F	3	8	2	0	0	0	0	1	0	3	0	0
G	2	3	2	2	1	1	0	4	1	1	1	0
H	0	3	0	0	0	1	0	0	1	0	0	0
I	9	23	4	0	2	5	0	6	1	2	0	0
J	1	2	0	0	0	0	0	3	3	0	0	0
K	1	2	0	1	0	0	0	0	0	0	0	0

Fig.5. An exploratory testing matrix with sessions colored by cluster.



Function	Function completeness	Function correctness	Function appropriateness	Co-existence	Interoperability	Appropriateness recognizability	Learnability	Operability	User error protection	User interface aesthetics	Accessibility	Confidentiality
A	10	12	9	0	1	1	0	4	5	2	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0
C	5	6	1	0	0	3	0	3	1	0	0	0
D	4	9	1	0	0	0	0	2	0	1	0	0
E	2	7	2	0	0	0	0	2	0	0	1	0
F	3	8	2	0	0	0	0	1	0	3	0	0
G	2	3	2	2	1	1	0	4	1	1	1	0
H	0	3	0	0	0	1	0	0	1	0	0	0
I	9	23	4	0	2	5	0	6	1	2	0	0
J	1	2	0	0	0	0	0	3	3	0	0	0
K	1	2	0	1	0	0	0	0	0	0	0	0

Fig.6. An exploratory testing matrix with a gray scale representing the priority of the searching points.

Next, all sessions colored in each cluster to the exploratory testing matrix in Fig.1, which is shown in Fig.5. Also, Fig.6 shows the priority as a gray scale shading. The darker sessions are the higher the priority for the next test.

Comparing Fig.1 and Fig.6, it can be confirmed that this method enables a finer prediction of the session containing the defect and makes it easier to determine the next area to search.

4. Evaluation of validity

We applied this system to a development project in order to confirm the validity of the proposed method. First, we analyze the sessions that contain defects from the results of the exploratory testing by using our proposal system. Then, we perform the next exploratory testing, and analyze the number of defects for each session. Finally, these results are compared with the results of our proposal system. In this experiment, the development project is real project which is the software of TV system. The parameter values for SOM are much the same items in Section 3.

Table.1 is shown the number of found defects for each testing and priority order to search in next testing. GREEN and MAGENTA were mapped in the area which indicated the importance of the quality characteristics is highest than the other clusters. This priority order was decided by the size of function. GREEN is the most priority, and MAGENTA is the second priority to search in the next testing. RED cluster and CYAN cluster were the next priority because the size of the function is larger than the other clusters. Therefore, we decided the priority in this order: GREEN, MAGENTA, RED, CYAN, BLUE and YELLOW.

Table.1. The number of found defects for each testing and priority order to search in next testing.

Cluster	Number of found defects (Average of sessions)	Priority order to search in next testing	Number of found defect in next testing (Average of sessions)
RED	1.50	Third	0.00
BLUE	0.90	Fifth	0.06
GREEN	3.00	First	0.50
YELLOW	0.00	Fifth	0.00
CYAN	0.00	Fourth	0.09
MAGENTA	1.70	Second	0.11

The results of the next exploratory test showed that the defects were found in GREEN and MAGENTA as predicted. Defects were also found in CYAN, where no defects had been found. This indicates that the proposed method can predict the defects. However, no defects were found in RED, which is the third priority. This point needs to be verified in the future, including the selection of the data to be input to SOM.

5. Conclusion

In this paper, we proposed a method "FaRSeT-#" which infer the important searching points, and a defect predication system that incorporates this method.

We confirmed that it can objectively determine the search location and clearly state the basis for the decision mechanically mapping and clustering the data by the system using FaRSeT-#. In the future, it is necessary to verify the selection of data to be input to the SOM.

References

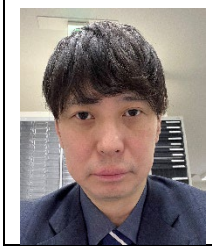
1. "Certified Tester Foundation Level Syllabus Version 2018 V3.1", International Software Testing Qualifications Board, 2018.
2. K. Ueda, J. Tanba, and S. Kudo, "Effort of application of test method "FaRSeT (Flexible and Rapid Software Test)" for short-term delivery development project", Software Quality Symposium 2018 (SQiP2018), pp.1-8, 2018 (in Japanese).
3. T. Kohonen, "Self-Organizing Maps", Springer-Verlag, 2001.
4. PractiTest, "Centralized Test Management for Manual & Automated Testing",
5. TESTPAD/
6. D. Steinley, M.J. Brusco, "Initializing k-means Batch Clustering: A Critical Evaluation of Several Techniques", Journal of Classification, Vol.24, No.1, pp.99-121, 2007.

Authors Introduction

Dr. Yoshihiro Kita

He received his Doctor's degree from the Department of Engineering, University of Miyazaki, Japan in 2011. He belongs to University of Nagasaki, Japan. His research area is biometrics, mobile security, and software testing.

Mr. Kazuki Ueda



He belongs Nihon Knowledge Co.,Ltd. His job is chief executive officer of verification technology. He research interest in development of quality assurance technology for test automation, exploratory testing, and AI products.

Mr. Kiyotaka Sakurai

He belongs the officer of verification technology, Nihon Knowledge Co.,Ltd. He verify the automatical tools, and research the operatable automatical tool by low-code.