GUI Test Case Prioritization using Social Network Analysis

Miss Chawannut Maitrikul

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2019

การจัดลำดับกรณีทดสอบจียูไอโดยใช้การวิเคราะห์เชิงเครือข่ายทางสังคม

น.ส.ชวัลนุช ไมตรีกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

Thesis Title	GUI Test Case Prioritization using Social Network Analysis
By	Miss Chawannut Maitrikul
Field of Study	Computer Science
Thesis Advisor	Associate Professor Yachai Limpiyakorn, Ph.D.

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

..... Dean of the FACULTY OF ENGINEERING

(Professor SUPOT TEACHAVORASINSKUN, Ph.D.)

THESIS COMMITTEE

..... Chairman

(Assistant Professor SUKREE SINTHUPINYO, Ph.D.)

..... Thesis Advisor

(Associate Professor Yachai Limpiyakorn, Ph.D.)

..... External Examiner

(Paskorn Apirukvorapinit, Ph.D.)

ชวัลนุช ไมตรีกุล : การจัดลำดับกรณีทดสอบจียูไอโดยใช้การวิเคราะห์เชิงเครือข่ายทางสังคม. (GUI Test Case Prioritization using Social Network Analysis) อ.ที่ปรึกษาหลัก : รศ. ดร.ญาใจ ลิ่มปิยะกรณ์

ส่วนต่อประสานกราฟิกผู้ใช้ หรือจียูไอ ได้รับความนิยมใช้กันอย่างแพร่หลายในผลิตภัณฑ์ซอฟต์แวร์ปัจจุบัน อันเนื่องมาจากข้อดีของความเป็นมิตรกับผู้ใช้ อย่างไรก็ตาม การทดสอบจียูไอมีต้นทุนค่าใช้จ่ายสูงและประสบกับปัญหากรณีทดสอบขนาดใหญ่ไร้ขอบเขต ยิ่งไปกว่านั้นวิวัฒนาการของซอฟต์แวร์เป็นเหตุให้จียูไอเปลี่ยนแปลงอย่างหลีกเลี่ยงไม่ได้ ทำให้การทดสอบยากยิ่งขึ้น การทดสอบถดถอยถูกใช้อย่างแพร่หลายสำหรับการทดสอบจียูไอเมื่อซอฟต์แวร์มีการเปลี่ยนแปลง แต่การทดสอบดังกล่าวนับเป็นขั้นตอนที่มีค่าใช้จ่ายมากสุดในกระบวนการทดสอบซอฟต์แวร์ ระหว่างหลายๆ คำตอบทางเลือกเพื่อช่วยแก้ปัญหาข้อจำกัดของเวลาและต้นทุนในขั้นตอนดังกล่าว เทคนิคการจัดลำดับกรณีทดสอบอาจช่วยระบุเซ็ตกรณีทดสอบในชุดทดสอบถดถอยให้มีขนาดเล็กลง แต่ยังคงรักษาเกณฑ์ทดสอบเท่าเดิม งานวิจัยนี้ได้นำเสนอการประยุกต์ใช้ค่าความเป็นศูนย์กลางในระบบเครือข่ายจากการวิเคราะห์เครือข่ายทางสังคมเพื่อจัดอันดับความสำคัญของกรณีทดสอบและค้นหาพารามิเตอร์ที่เหมาะสมสำหรับการจัดลำดับกรณีทดสอบจียูไอ แนวทางดังกล่าวจะมีผลทำให้ทรัพยากรที่ใช้ระหว่างการประมวลผลทดสอบลดลง และเพิ่มอัตราการตรวจจับข้อบกพร่องได้เร็วขึ้นตั้งแต่ในระยะแรกๆของการทดสอบ

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2562 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6170916121 : MAJOR COMPUTER SCIENCE
KEYWORD: test case prioritization, social network analysis, network centrality

Chawannut Maitrikul : GUI Test Case Prioritization using Social Network Analysis. Advisor: Assoc. Prof. Yachai Limpiyakorn, Ph.D.

Graphical User Interfaces (GUIs) has become popular in today software products due to its user-friendly merit. However, GUI tests are costly and suffer explosive test cases. Moreover, software evolution causes the GUIs inevitably changed resulting in harder testing. Regression test has been widely used for GUIs software testing when software changes. It is considered to be the most expensive phase in the software testing process, though. Among several alternatives, the technique of test case prioritization would help identify the reduced set of test cases in the regression testing suite, still maintain the same criteria as the original number of test cases in order to solve the limitation of time and cost in this phase. This research thus proposes applying network centrality from social network analysis for ranking test case importance and finding suitable parameter(s) for GUI test case prioritization. The approach would lead to reduced resources during test execution and increased fault detection rate in early stages.

Field of Study: Computer Science Student's Signature .....
Academic Year: 2019 Advisor's Signature .....

ACKNOWLEDGEMENTS

The completion of this thesis could not have been possible without my thesis advisor, Assoc. Prof. Dr. Yachai Limpiyakorn for her invaluable help and constant encouragement throughout this research. I am most grateful for her teaching and advice, not only the research methodologies but also many other methodologies in life. I would not have achieved this far, and this thesis would not have been completed without all the support that I have always received from her.

In addition, I am grateful for the teachers, Asst. Prof. Dr. Sukree Sinthupinyo and Dr. Paskorn Apirukvorapinit and other people for suggestions and all their help.

Finally, I most gratefully acknowledge my parents and my friends for all their support throughout the period of this research.

Chawannut Maitrikul

TABLE OF CONTENTS

Page

List of Tables

Page

List of Figures

Page

Chapter 1

Introduction

1.1 Statement of the Problems

Many of the software products have been developed daily and rolled out to support customer needs. The development process includes adding new features and modifying existing features. But before all the features can be introduced to the market, developers need to ensure the correctness of code modification and avoid the side effects caused by code modification to other modules. One of the processes to make sure that such a case will not happen is to test the feature before rolling it out. Existing test suites aim to avoid detecting regressions, validating software changes, and avoiding the introduction of new defects by using regression tests. However, according to the size of the software and number of features, regression tests would be both time consuming and account for a huge amount of the software maintenance cost. Research shows regression testing is an expensive process that may require more than 33% of the cumulative expenses of the software [1]. Thus, the measures to save both time and budgets are necessary. In the work of Shin Y, and Harman M [2] divided the optimization technology into three domains: test case reduction, test case selection, and test case prioritization (TCP). In this Thesis, we mainly discuss the TCP technology on GUI.

Test case prioritization (TCP) [1] aims to order a set of test cases to achieve and early optimization based on preferred properties. It gives an approach the ability to execute highly significant test cases first according to some measure, and produce the desired outcome, such as revealing faults earlier. It also helps testers to manage risks plan tests, consider cost value, and be analytical about which test to run in the context of the specific project.

In social network analysis, centrality has been one of the important methods, there several ways to measure network centrality including Degree centrality, Betweenness centrality, Closeness centrality, Eigenvector centrality and Page rank.

In summary, this thesis aims to apply network centrality methods from social network analysis to prioritize GUIs test case scenarios and then apply the method to real-world applications.

1.2 Objective

To apply the centrality method from network analysis for prioritizing GUI test cases, compare with the existing method and apply the approach to real-world GUI tests.

1.3 Scope of Study

1. Use graph theory as the base of test case scenarios.

2. Use test cases from a Wongnai application (Restaurant Management System Application) for evaluation.

3. The importance of the function and test case is ranked by the measurement value of network centrality.

1.4 Contribution

1. More alternative approach of GUI test case prioritization with social network analysis based on graph theory.

2. Empirical results contributed to network centrality as a measure for ranking test case importance that would lead to reduced resources during test execution and increased fault detection rate in early stages.

1.5 Thesis Publication

This thesis had been published in the conference proceedings.

Maitrikul, C., & Limpiyakorn, Y. (2020). GUI Test case Prioritization using Social Network Analysis. *In 13th International Conference on Computer and Electrical Engineering.* Beijing, China.

Chapter 2

Literature Review

2.1 Graphical User Interface (GUI)

GUI is a system of interactive visual components for computer software. GUI was first developed by Xerox Corporation's Palo Alto Center in the 1970s [3] and today has become widely applied in many software products because this type of operating system is easy to understand and gives a friendly environment to users more than command-line interface (CLI), Graphical interface is flexible to make interactions between users and software products. The user can work on it easily by clicking the icons and open the file, using the keyboard shortcut of holding down the Ctrl key and pressing the letter "P" to print, etc. without writing any command.

GUI is what the user sees. In Figure 1 if you visit wongnai.com, the user sees the homepage and he or she does not see the source code. The interface is visible to the user and they simply focus on the design structure, images that they are working properly or not.

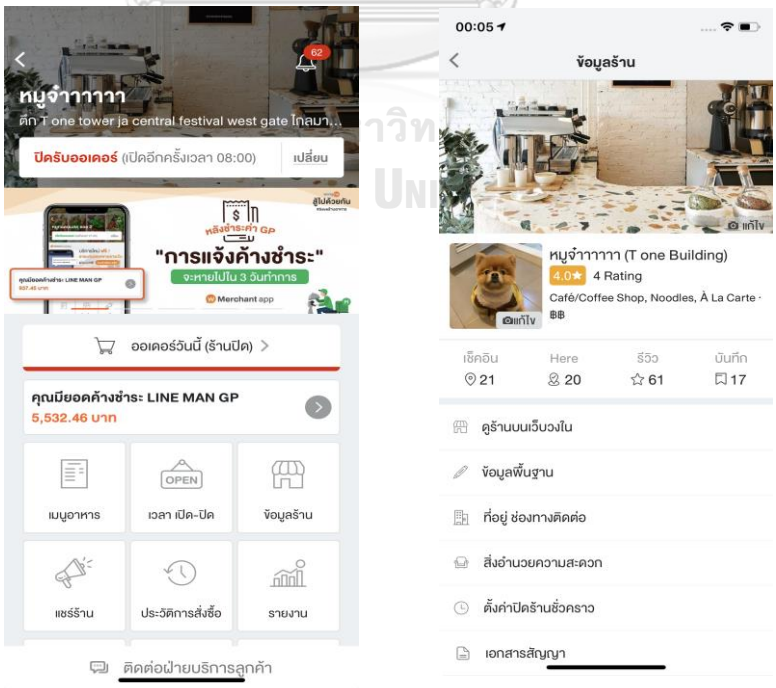

Figure 1 Restaurant Management System Application

2.2 Software Product Testing

Software testing is the basic activity to check the actual results match the expected results (Software requirements). Testing is aimed to detect and solve technical issues in the software source code and assess the overall product usability, performance, security, and experience to ensure that the software product is defect-free. There are two types of software testing approaches [4]. Their details are introduced as follows.

1) White Box Testing: This testing is highly effective in detecting and resolving problems. White box testing is based on a system internal code structure. So programming skills are required to design test cases.
2) Black Box Testing: This testing software based on output requirements and without any knowledge of the internal structure or coding in the program. The goal is to test how well the component conforms to the published requirement for the component.

2.3 Graphical User Interface Testing (GUI Testing)

GUI Testing [5] is a type of testing that checks the Graphical User Interface of the software product. GUI Testing required checking the screens with the controls like icons, menus, buttons, text boxes, type of toolbar, and menu bar, etc. The purpose of GUI Testing is to ensure that User Interface functionality can work specifications and correctly.

There are three approaches to GUI Testing

1) Manual Testing: In Figure 2 Testers check the graphical screen manually by creating and executing test cases from a requirement document. This approach is convenient to use where the UI is unstable and has a lot of changes.

Figure 2 Manual Testing

2) Record and Replay Testing: As shown in Figure 3, this testing can be done by using automation tools. Usually used in regression tests, Testers run an application and record the user interaction with the system. Testers write a script for GUI testing and run to track and save the user actions, including step movement, which can be replayed several times to find the defect in the interface.

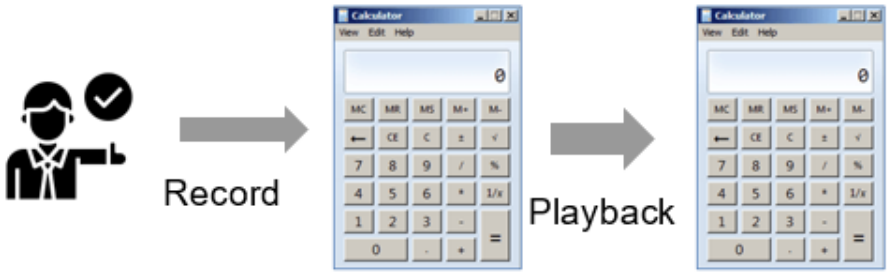

Figure 3 Record and Replay Testing

1) Model-Based Testing: This testing describes a system's behavior. It can help testers to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements. For example, using a Genetic algorithm, Sharma A, Patani R, and Aggarwal A [6] use a Genetic algorithm to evaluate the fitness function for selecting the best possible test method. These methods take the test populations as an input and then evaluate the test cases for the system.

The challenges of GUI Testing while doing Regression Testing is that the User interface changes frequently, it is difficult to test and identify.

2.4 Test case Prioritization (TCP) with GUI

In the software testing process, Testers must ensure that the GUI of the system is correct by using testing techniques to generate test cases. Before starting to test, we should determine the criteria for knowing the sufficiency of testing software. When the criteria are committed, the next step is to create a test case. During the testing process, testing the defects is an expensive and time-consuming activity. So, many researchers applied testing techniques to solve a problem such as Multi-objective test case prioritization [7], Test case prioritization for GUI application [8] to solve the limitation of time and resource. Test case reduction [9] can reduce the number of test cases in the testing phase but the remaining test cases can maintain the same criteria as the original number of test cases.

2.5 Event Handler Tree (EHT)

An Event Handler of a GUI application is the function written by the user but invoked automatically when the corresponding event is performed. There is a mapping relationship between events and Event Handlers, but the relation is not one-to-one.

Bin Wang et al. [10] proposed an EHT model to assist the test case generation process. They can automatically combine the cell events into runnable test cases, the result is their test cases are easy to repair when GUI is changed or modified.

2.6 Graph Theory and Centrality Measurement

2.6.1 Graph is a mathematical representation of a network and describes the relationship between points and lines, each point represented by Node in the graph while edges represent the relation between 2 nodes.

2.6.2 Centrality Measures [11] [12] [13] are based on graph theory and social network analysis. So far, briefly discussed the method and usability of each Centrality Type including Degree Centrality, Closeness Centrality, Betweenness Centrality, Eigenvector Centrality and Page Rank. Their details are introduced as follows.

Degree Centrality uses the degree of node to measure the centrality. In real-world interactions, we consider people who have many connections to be important. Degree Centrality has the same idea to measure, The degree centrality measure rank nodes with more connections higher in term of centrality as Equation (1).

$$C_d(V_i) = d_i \tag{1}$$

Where d_i is the degree of node v_i

Betweenness Centrality defined as the proportion of all the shortest paths passing through the node in the network as Equation (2).

$$b_i(g) = \sum_{(j,k),j\neq k\neq i} \frac{v_g\,(i{:}j,k)}{v_g(j,k)} \tag{2}$$

Where $v_g(j,k)$is the number of shortest paths between nodes j to nodes k, $v_g(i{:}\,j,k)$ is the number of the shortest paths between node j and nodes k through node i.

Closeness Centrality is based on the network distance between a node and each other node in the network as Equation (3).

$$C_c(v_i) = \frac{1}{l_{vi}} \tag{3}$$

Where $\underline{l}_{vi} = \frac{1}{n-1}\sum_{v_i\neq v_j} \quad l_{i,j}$ is node v'_i average shortest path length to other nodes. The smaller the average shortest path length, the higher the centrality for the node.

Eigenvector Centrality is considered nodes with more connections to be more important. However, in the real world, having more friends does not by itself guarantee that someone is important: having more important friends provides a stronger signal as Equation (4).

$$C_e(v_{\,i}) \;=\; \frac{1}{\lambda}\sum_{j=1}^{n} A\;i,j\;c_e(v_j) \tag{4}$$

Page Rank is like eigenvector centrality but not everyone known by a well-known person is well known. To mitigate this problem, one can divide the value of passed centrality by the number of outgoing links from the node such that each connected neighbor gets a fraction of the source node's centrality as Equation (5).

$$C_p(v_i) = \alpha \sum_{j=1}^{n} A\ i,j\ \frac{c(v_i)}{d_j^{out}} + \beta \quad (5)$$

This research focused on four metrics: Closeness Centrality, Betweenness Centrality, Eigenvector Centrality and Page Rank.

2.7 Use-Case Weight Estimation

The basic concept of the use case models is used in object-oriented analysis for capturing and describing the functional requirements of the system.

Anda B and team [14] describe that the estimation method requires to count the number of transactions in each use case. A transaction is an event occurring between an actor and the system. They define actors in the use case model as simple, average, and complex. A weighting factor is assigned to each use case category.

Nageswaran S [15] determine the number of use cases in the system and assigned weights depending on the number of transactions/scenarios. They categorized the use case as simple, average, and complex. A simple use case has 3 or fewer transactions; an average use case has 4 to 7 transactions; and complex use case has more than 7 transactions. A weighting factor is assigned to each use case category;

- Simple: Weighting factor 1
- Average: Weighting factor 2
- Complex: Weighting factor 3

Chapter 3

Methodology

In this chapter, we will drive deep into the technical details of each component that contributed to the final outcomes of the research. The overall process is briefly introduced as Figure 4.

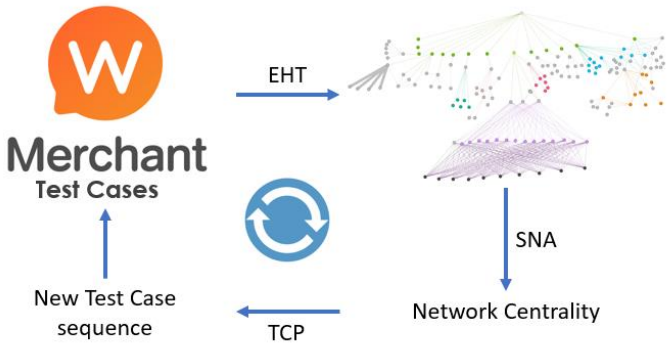

Figure 4 Test cases Prioritization workflow

3.1 Dataset

A test case dataset from Wongnai Restaurant Management System Application (RMS) is selected. There are around 100 primary test cases that have been tested in every test cycle, we use those test cases data set as an example in this experiment.

For example, Table 1 shows 3 Test case scenarios Add a new menu, Edit the existing menu and Add new category, all path attribute values are transformed to numeric. The Function paths are labelled 1 and No-Function path are labelled 0.

Table 1 Test case scenario

Function / Test Case	Main	Menu	Add new menu	Edit menu	Create category	Menu image	Menu name	Manu price	Manu category	Add category	Edit category	Category list
Add new menu	1	1	1	0	0	1	1	1	1	0	0	0
Edit Existing menu	1	1	0	1	0	1	1	1	1	0	0	1
Add new category	1	1	0	0	1	0	0	0	1	1	1	1

Example.

Testcase: Add new menu

Action:

Main page ⟶ Menu page ⟶ Add new menu ⟶ Add menu image ⟶ Add menu name ⟶ Add menu price ⟶ Add menu category

Another factor that has been considered in this experiment is the Test Estimation Effort. Test Estimation Efforts are assigned weights depending on the number of expected results and complex class. We classify the test complexity in three classes i.e. Complex, Average, and Simple based upon the total number of expected results. A complex class has more than 18 expected results given weight to 5; an average class has 9 to 17 expected results given weight 3; and a simple class has 9 or fewer expected results given weight 2.

An example of a function with expected results, complex class, and test estimation effort as shows In Table2.

Table 2 Example Test Estimation Effort

Function	Expected results	Classes	Test points	Test Estimation Effort
Menu	4	simple	2	8
Add new menu	24	complex	5	120
Edit menu	27	complex	5	135
Add menu category	15	average	3	45
Menu menu photo	1	simple	2	2
Menu name	1	simple	2	2
Menu price	1	simple	2	2
Menu category	1	simple	2	2
Add category	1	simple	2	2

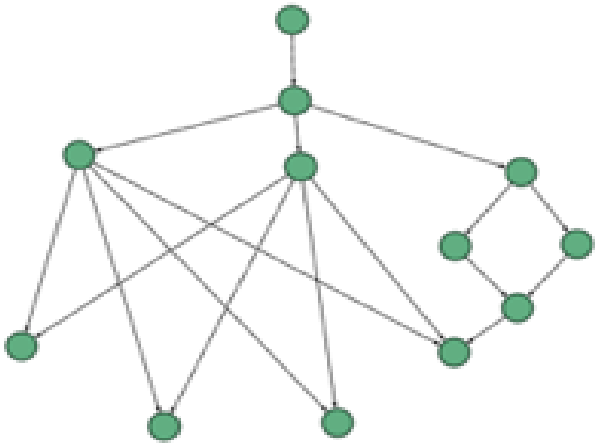

Figure 5 Test cases prioritization workflow

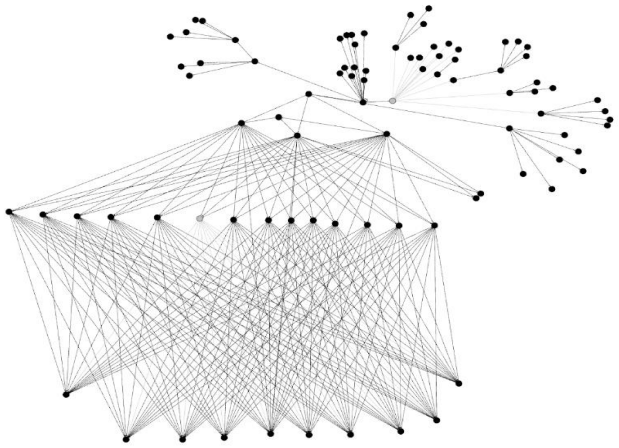

Figure 6 Test case scenario from Wongnai Application

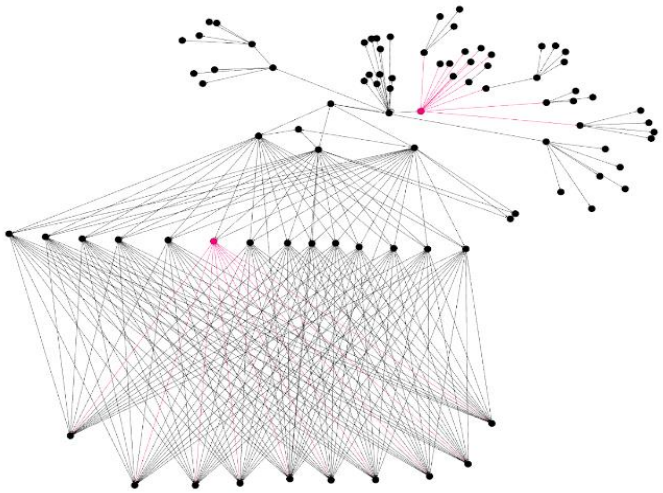

Figure 7 EHT from Wongnai Scenario with modify and new Function

3.3 Social network analyst

We calculate each Function consequence by using social network analysis measurement. In our research, we consider Closeness centrality, Betweenness centrality, Eigenvector centrality, and PageRank to measure the importance of each node. We use Gephi application to do the calculation in this step.

3.4 Test case prioritization in each cycle

We used the result to prioritize the test case with a focus on version-specific prioritization. We keep records for 10 versions of the Wongnai application release. Follow with we test, the test case using a new sequence in each criterion and measure the error in the system, then compare the earliness of error finding between each measurement in the social network analysis to find the best measurement that suits our test cases.

3.5 Overall evaluation

We recorded 10 test cycles along the experiment period. In each cycle, the modified features, and new features are added to the network. We briefly discuss the experimental process and outcome in the following section.

Chapter 4

Experiments

The experiments demonstrated in the section are based on the previous setup aimed to answer the objectives of the thesis.

This section presents two levels of an experiment, Experiment 4.1 focuses on version-specific prioritization, while Experiment 4.2 consider fault detection.

4.1 Version-specific prioritization

In this experiment section, we focus on version-specific prioritization of the release's version of Wongnai RMS application, information of version 1 to version 10 as shown in Table 3.

Table 3 Number of functions in each version

Version	*Total Function*	*Modified Function*	*New Function*
1	*150*	*-*	*-*
2	*150*	*1*	*0*
3	*156*	*6*	*6*
4	*158*	*5*	*2*
5	*166*	*0*	*8*
6	*177*	*0*	*11*
7	*177*	*7*	*0*
8	*188*	*1*	*11*
9	*200*	*5*	*12*
10	*200*	*2*	*0*

4.1.1 Version 1

Figure 8 shows the Version 1 Test case graph generate using Gephi application, Node with blue color is complex classes, pink is average classes, and green is simple classes.

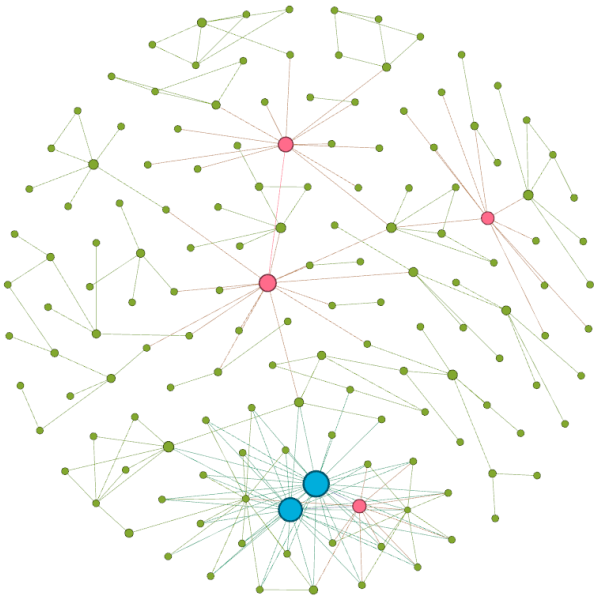

Figure 8 Version 1 Test case graph

Next, Closeness centrality, Betweenness centrality, PageRank, and Eigenvector centrality are calculated using function in Gephi application. In this version, it is the first version of the experiment, no new function, and modify the function in this version, the reason is that researchers want to understand the most importance Function in the network system before adding another factor into the calculation. Table 4, table 5, table 6, and Table 7 shows the top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. After combining the result from each centrality, the top 10 functions having the highest value represented in Table 8

Table 4 Top 5 Closeness centrality measurement of version 1

Rank	Function	Closeness centrality
1	main	0.3725
2	menu	0.319742
3	Setting	0.312369
4	Reward card	0.311065
5	Today order	0.281132

Table 5 Top 5 Betweenness centrality measurement of version 1

Rank	Function	Betweenness centrality
1	main	0.786595
2	menu	0.405734
3	Setting	0.294214
4	Reward card	0.260249
5	Reward card setting	0.213677

Table 6 Top 5 Eigenvector centrality measurement of version 1

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Add menu	0.939391
3	save menu	0.689427
4	Add new menu option	0.672558
5	save menu option value	0.568154

Table 7 Top 5 PageRank centrality measurement of version 1

Rank	Function	PageRank
1	main	0.036307
2	Edit existing menu	0.03374
3	Setting	0.032691
4	Reward card setting	0.028842
5	Add menu	0.027673

Table 8 Top 10 overall centrality of version 1

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.269	0.178	0.034	1.000	1.481
Add menu	0.267	0.137	0.028	0.939	1.370
main	0.373	0.787	0.036	0.133	1.329
menu	0.320	0.406	0.010	0.287	1.022
save menu	0.218	0.013	0.020	0.689	0.940
Add new menu option	0.217	0.001	0.014	0.673	0.905
Save menu option value	0.216	0.001	0.012	0.568	0.797
Setting	0.312	0.294	0.033	0.086	0.725
Reward card	0.311	0.260	0.017	0.068	0.656
Add menu category	0.260	0.068	0.012	0.278	0.618

4.1.2 Version 2

Figure 9 shows the second version test case graph generate using Gephi application. In this version, there is a 1 modified function which shows as a pink node in Figure 10 and detail in Table 9 Table 10, Table 11, Table 12, and Table 13 shows the top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, and PageRank respectively. Even though there is no new function in this version, but the modified function is affected by other functions more than version 1, so the centrality values have been changed. However, there is no significant change among the ranking score, which leaves us the same top function priorities order.

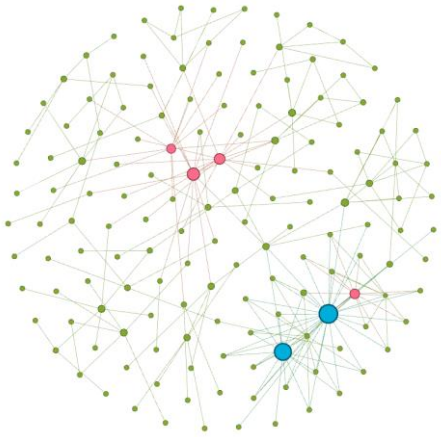

Figure 9 version 2 test case graph

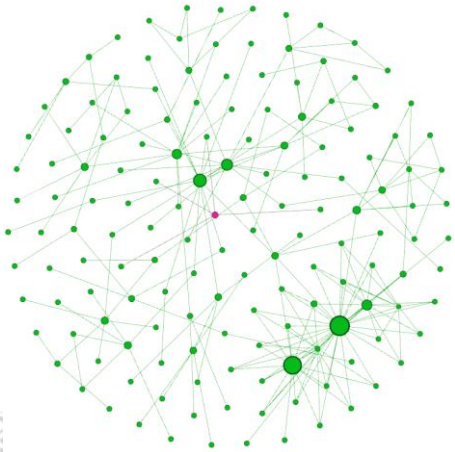

Figure 10 version 2 Modified and Added function

Table 9 Modify and Add function of version 2

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
Order history report	Modified	0.218659	0.243841	0.014384	0.0131

Table 10 Top 5 Closeness centrality measurement of version 2

Rank	Function	Closeness centrality
1	main	0.373134
2	menu	0.321199
3	Setting	0.3125
4	Reward card	0.311203
5	Today order	0.281426

Table 11 Top 5 Betweenness centrality measurement of version 2

Rank	Function	Betweenness centrality
1	main	0.785369
2	menu	0.411167
3	Setting	0.292438
4	Reward card	0.258747
5	Edit existing menu	0.241749

Table 12 Top 5 Eigenvector centrality measurement of version 2

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	save menu	0.71699
3	Add menu	0.565821
4	Add new menu option	0.523019
5	save menu option value	0.40799

Table 13 Top 5 PageRank centrality measurement of version 2

Rank	Function	PageRank
1	Edit existing menu	0.038046
2	main	0.03618
3	Setting	0.032491
4	Reward card setting	0.028655
5	save menu	0.020812

Table 14 Top 10 overall centrality of version 2

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.27	0.24	0.04	1.00	1.55
main	0.37	0.79	0.04	0.16	1.36
menu	0.32	0.41	0.01	0.27	1.02
Save menu	0.22	0.00	0.02	0.72	0.96
Add menu	0.26	0.09	0.02	0.57	0.93
Add new menu option	0.22	0.00	0.01	0.52	0.76
Setting	0.31	0.29	0.03	0.11	0.75
Reward card	0.31	0.26	0.02	0.09	0.67
Save menu option value	0.22	0.00	0.01	0.41	0.64
Add existing menu option	0.22	0.00	0.01	0.37	0.59

4.1.3 Version 3

Figure 11 shows the third version test case graph generate using Gephi application. In this version, there are 6 modified functions and 6 added functions. Which shows as a pink node and yellow node respectively in Figure 12 and detail provided in Table 15. Table 16, Table 17, Table 18, and Table 19 shows the Top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. We analyze results from each centrality and found that orders are swapping between each rank compared to version 2. However, the list of top functions that got highest remain the same.

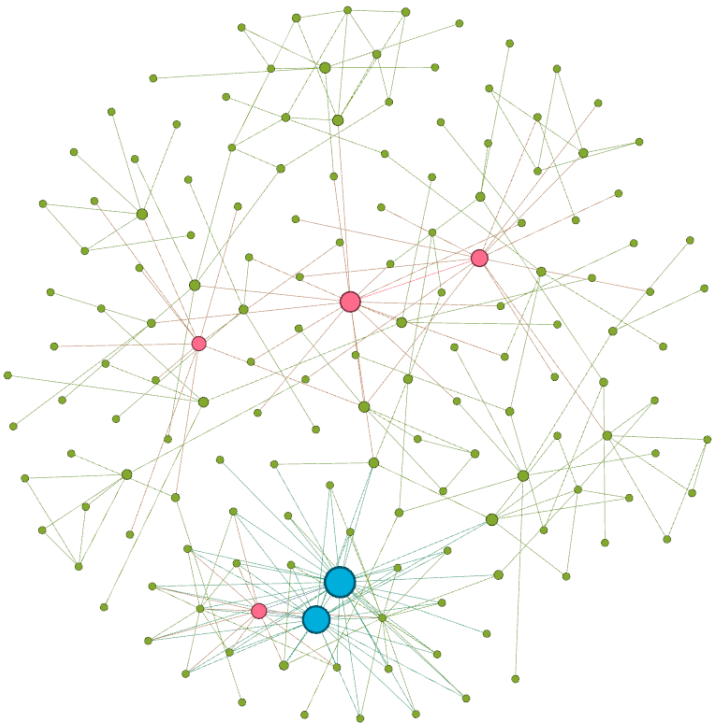

Figure 11 version 3 test case graph

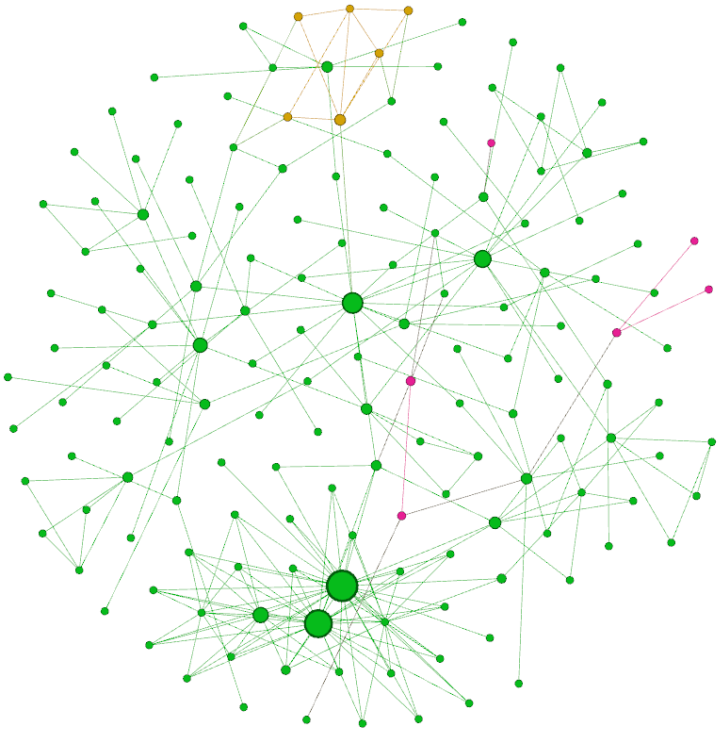

Figure 12 version 3 Modified and Added function

Table 15 Modify and Add of version 3

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
Transaction	Modified	0.183215	0	0.004243	0.003419
New order	Modified	0.187198	0.025723	0.009873	0.009515
Working order	Modified	0.206667	0.025639	0.015227	0.007821
accept order	Modified	0.157841	0	0.003561	0.003656
popup time to create order	Modified	0.157841	0	0.003561	0.003656
edit order detail	Modified	0.252443	0.051571	0.04309	0.008578
Working hour main	Add	0.281307	0.0687	0.05263	0.010844
Add restaurant working hour	Add	0.222701	0.0003	0.023162	0.005705
Add delivery working hour	Add	0.221429	0.004182	0.021516	0.005717
Edit restaurant working hour	Add	0.222701	0.0003	0.023162	0.005705
Edit delivery working hour	Add	0.221429	0.004182	0.021516	0.005717
Save working hour button	Add	0.222063	0.000357	0.031163	0.00897

Table 16 Top 5 Closeness centrality measurement of version 3

Rank	Function	Closeness centrality
1	main	0.378049
2	menu	0.321577
3	Setting	0.314402
4	Reward card	0.313131
5	Today order	0.283883

Table 17 Top 5 Betweenness centrality measurement of version 3

Rank	Function	Betweenness centrality
1	main	0.798872
2	menu	0.394927
3	Setting	0.283871
4	Reward card	0.251487
5	Reward card setting	0.205949

Table 18 Top 5 Eigenvector centrality measurement of version 3

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Add menu	0.918637
3	save menu	0.686891
4	Add new menu option	0.658532
5	save menu option value	0.555696

Table 19 Top 5 PageRank centrality measurement of version 3

Rank	Function	PageRank
1	main	0.035469
2	Edit existing menu	0.033393
3	Setting	0.031195
4	Reward card setting	0.027682
5	Add menu	0.026412

Table 20 Top 10 overall centrality of version 3

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.269565	0.180135	0.033393	1	1.483093
main	0.378049	0.798872	0.035469	0.145167	1.357557
Add menu	0.266323	0.128264	0.026412	0.918637	1.339636
menu	0.321577	0.394927	0.009071	0.284631	1.010206
save menu	0.218003	0.003725	0.018588	0.686891	0.927207
Add new menu option	0.21648	0.001099	0.013322	0.658532	0.889433
Save menu option value	0.215877	0.000597	0.011428	0.555696	0.783598
Setting	0.314402	0.283871	0.031195	0.086517	0.715985
Reward card	0.313131	0.251487	0.015824	0.068765	0.649207
Add menu category	0.259631	0.061482	0.011299	0.275021	0.607433

4.1.4 Version 4

Figure 13 shows the fourth version test case graph generate using Gephi application. In this version, there are 5 modified functions and 2 added functions. Which shows as a pink node and yellow node respectively in Figure 14 and detail provided in table 21. Table 22, Table 23, Table 24, and Table 25 shows top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. 2 added function that adding into the network are added at the end the line which does not affect other function in the application. The result shows that top prioritize function remain the same.

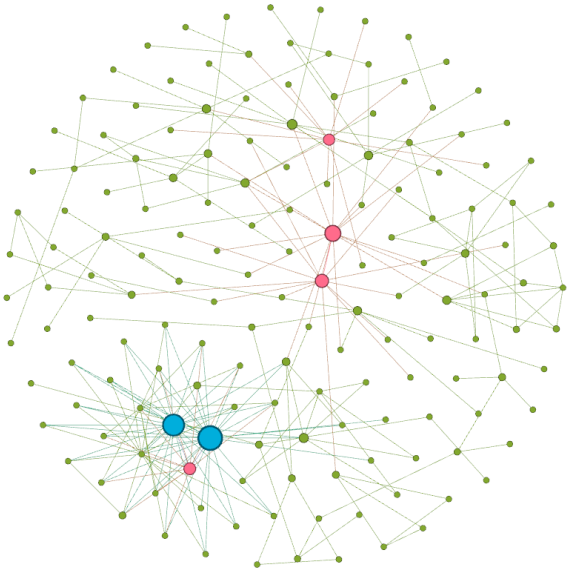

Figure 13 version 4 test case graph

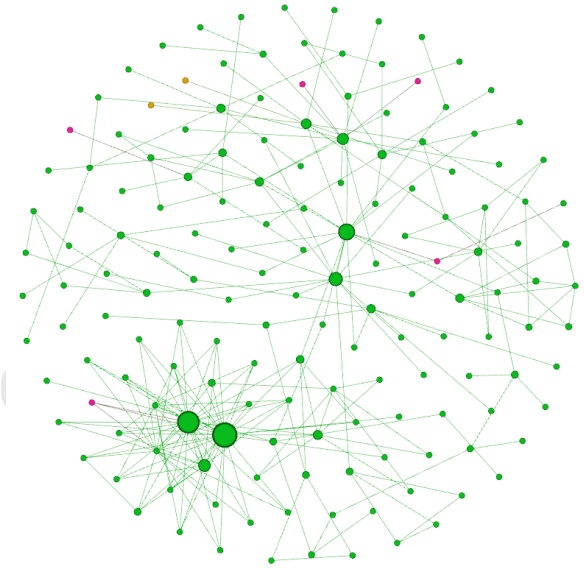

Figure 14 version 4 Modified and Added function

Table 21 Modify and Add function of version 4

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
Add menu photo	Modified	0.213315	0.264945	0.278122	0.003688
upload restaurant logo photo	Add	0.221751	0.241143	0.010601	0.002953
upload restaurant cover photo	Add	0.221751	0.241143	0.010601	0.002953
add photo	Modified	0.182984	0.198446	0.005061	0.003585
edit card cover photo	Modified	0.20154	0.22569	0.010635	0.003274
edit card logo photo	Modified	0.20154	0.22569	0.010635	0.003274
Announcement	Modified	0.276408	0.300743	0.027144	0.005676

Table 22 Top 5 Closeness centrality measurement of version 4

Rank	Function	Closeness centrality
1	main	0.379227
2	menu	0.321721
3	Setting	0.314629
4	Reward card	0.313373
5	Restaurant information	0.28442

Table 23 Top 5 Betweenness centrality measurement of version 4

Rank	Function	Betweenness centrality
1	main	0.802156
2	menu	0.39143
3	Setting	0.280581
4	Reward card	0.248693
5	Reward card setting	0.203495

Table 24 Top 5 Eigenvector centrality measurement of version 4

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Add menu	0.918637
3	save menu	0.686867
4	Add new menu option	0.658508
5	save menu option value	0.555676

Table 25 Top 5 PageRank centrality measurement of version 4

Rank	Function	PageRank
1	main	0.035172
2	Edit existing menu	0.032975
3	Setting	0.030821
4	Reward card setting	0.027336
5	Add menu	0.026082

Table 26 Top 10 overall centrality of version 4

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.269297	0.178092	0.032975	1	1.480364
main	0.379227	0.802156	0.035172	0.147346	1.363901
Add menu	0.266102	0.126885	0.026082	0.918637	1.337706
menu	0.321721	0.39143	0.008966	0.284948	1.007065
save menu	0.217753	0.00363	0.018354	0.686867	0.926604
Add new menu option	0.216253	0.001071	0.013155	0.658508	0.888987
save menu option value	0.215659	0.000581	0.011285	0.555676	0.783201
Setting	0.314629	0.280581	0.030821	0.086906	0.712937
Reward card	0.313373	0.248693	0.015639	0.069135	0.64684
Add menu category	0.259504	0.060737	0.011158	0.275052	0.606451

4.1.5 Version 5

Figure 15 shows the fifth version test case graph generation using Gephi application. In this version, there are 8 added functions related to the Covid-19 situation added in this cycle. Which shows as a pink node in Figure 16 and detail provided in Table 27. Table 28, Table 29, Table 30, and Table 31 shows the top 5 function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. Although there is a big function added into the application. But due to the lack of relation between other existing functions in the application, the result shows that it did not impact the prioritize function in the network.

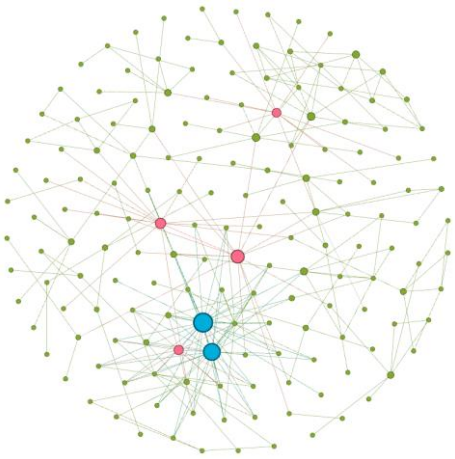

Figure 15 version 5 test case graph

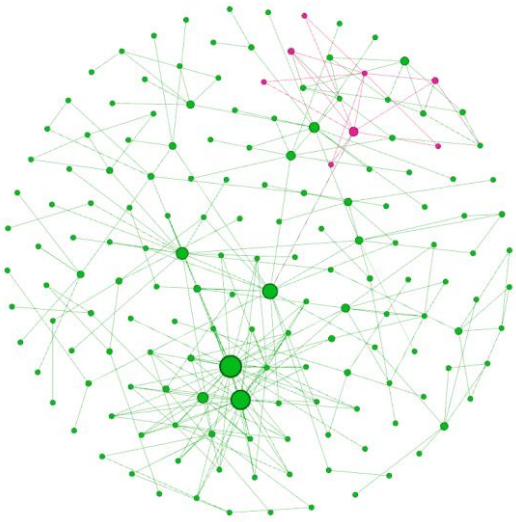

Figure 16 version 5 Modified and Added function

Table 27 Modify and Add function of version 5

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
Be safe setting	Add	0.286213	0.115821	0.06854	0.012947
Add restaurant working hour	Add	0.22905	0.012913	0.036951	0.00643
Edit restaurant working hour	Add	0.227147	0.028625	0.034695	0.006379
Dine in allow	Add	0.223129	0	0.022719	0.003645
Message to customer	Add	0.223129	0	0.022719	0.003645
Covid policy	Add	0.223129	0	0.022719	0.003645
Add restaurant photo Covid	Add	0.223129	0	0.022719	0.003645
Save Covid policy	Add	0.227462	0.001965	0.047408	0.011203

Table 28 Top 5 Closeness centrality measurement of version 5

Rank	Function	Closeness centrality
1	main	0.374429
2	menu	0.315992
3	Setting	0.309434
4	Reward card	0.308271
5	Restaurant information	0.286213

Table 29 Top 5 Betweenness centrality measurement of version 5

Rank	Function	Betweenness centrality
1	main	0.809377
2	menu	0.379579
3	Setting	0.269639
4	Reward card	0.239376
5	Reward card setting	0.195346

Table 30 Top 5 Eigenvector centrality measurement of version 5

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Add menu	0.918641
3	save menu	0.686947
4	Add new menu option	0.658589
5	save menu option value	0.555741

Table 31 Top 5 PageRank centrality measurement of version 5

Rank	Function	PageRank
1	Edit existing menu	0.031557
2	main	0.031061
3	Setting	0.029441
4	Reward card setting	0.026172
5	Add menu	0.024961

Table 32 Top 10 overall centrality of version 5

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.26409	0.171287	0.031557	1	1.466934
main	0.374429	0.809377	0.031061	0.143754	1.358621
Add menu	0.261146	0.122275	0.024961	0.918641	1.327023
menu	0.315992	0.379579	0.008548	0.284202	0.988321
save menu	0.214099	0.003326	0.017569	0.686947	0.921941
Add new menu option	0.212711	0.000982	0.012591	0.658589	0.884873
save menu option value	0.21216	0.000533	0.010802	0.555741	0.779236
Setting	0.309434	0.269639	0.029441	0.085871	0.694385
Reward card	0.308271	0.239376	0.014916	0.068187	0.63075
Add menu category	0.255054	0.058266	0.010676	0.274944	0.59894

4.1.6 Version 6

Figure 17 shows the sixth version test case graph generation using Gephi application. In this version, 11 functions added to the application related to payment and a new function from the Lineman application, which shows as a pink node in Figure 18 and details provided in Table 33-37 show the top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. The result from the new function is as same as version 5 which is it does not affect the overall measurement value in the application and lead to the remainder of function prioritize orders

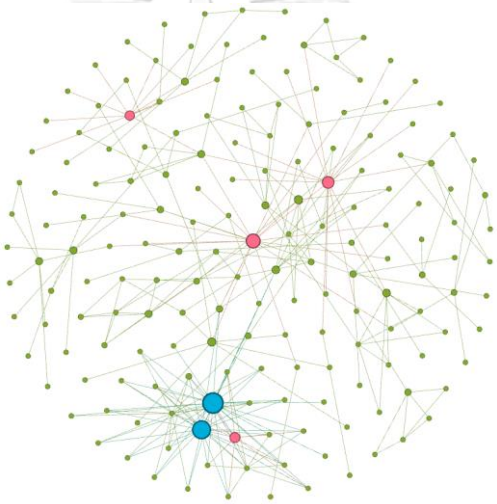

Figure 17 version 6 test case graph

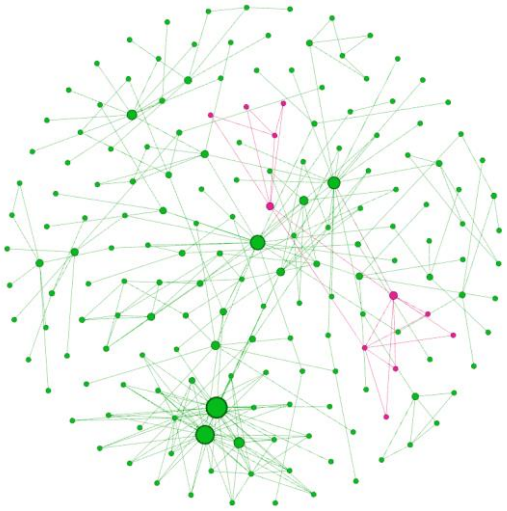

Figure 18 version 6 Modified and Added function

Table 33 Modify and Add function of version 6

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
self-delivery	Add	0.250354	0.108051	0.053531	0.011486
Add line Official account	Add	0.20068	0	0.019007	0.003625
Add Payment Account	Add	0.20068	0	0.019007	0.003625
Delete Payment Account	Add	0.20068	0	0.019007	0.003625
Add delivery fee condition	Add	0.202517	0.044524	0.035154	0.009858
Description for customer	Add	0.20068	0	0.019007	0.003625
Enable self-delivery service	Add	0.202517	0.000835	0.036689	0.009797
Delivery fee	Add	0.168732	0	0.013184	0.003724
Distance	Add	0.168732	0	0.013184	0.003724
Minimum spending	Add	0.168732	0	0.013184	0.003724
Save delivery fee condition	Add	0.169054	0.000096	0.019094	0.006987

Table 34 Top 5 Closeness centrality measurement of version 6

Rank	Function	Closeness centrality
1	Main	0.378205
2	Setting	0.321234
3	Menu	0.314947
4	Reward card	0.31383
5	Be safe setting	0.283654

Table 35 Top 5 Betweenness centrality measurement of version 6

Rank	Function	Betweenness centrality
1	Main	0.800813
2	Menu	0.359107
3	Setting	0.351695
4	Reward card	0.223774
5	Reward card setting	0.181818

Table 36 Top 5 Eigenvector centrality measurement of version 6

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Add menu	0.918635
3	Save menu	0.686626
4	Add new menu option	0.658269
5	Save menu option value	0.555484

Table 37 Top 5 PageRank centrality measurement of version 6

Rank	Function	PageRank
1	Main	0.03203
2	Edit existing menu	0.029234
3	Setting	0.027939
4	Reward card setting	0.024205
5	Add menu	0.023124

Table 38 Top 10 overall centrality of version 6

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.261448	0.15992	0.029234	1	1.450602
Main	0.378205	0.800813	0.03203	0.169073	1.380121
Add menu	0.258772	0.114524	0.023124	0.918635	1.315055
Menu	0.314947	0.359107	0.007887	0.288007	0.969948
Save menu	0.211976	0.002854	0.016279	0.686626	0.917735
Add new menu option	0.210714	0.000842	0.011667	0.658269	0.881492
Setting	0.321234	0.351695	0.027939	0.104972	0.80584
Save menu option value	0.210214	0.000457	0.010009	0.555484	0.776164
Reward card	0.31383	0.223774	0.013635	0.075116	0.626355
Add existing menu option	0.208726	0.000024	0.004906	0.381145	0.594801

4.1.7 Version 7

Figure 19 shows the seventh version test case graph generate using Gephi application. In this version, 11 functions are modified, which shows as a pink node in Figure 20 and detail provided in Table 39. Table 40, Table 41, Table 42, and Table 43 shows the top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. The overall function prioritize order remain the same.

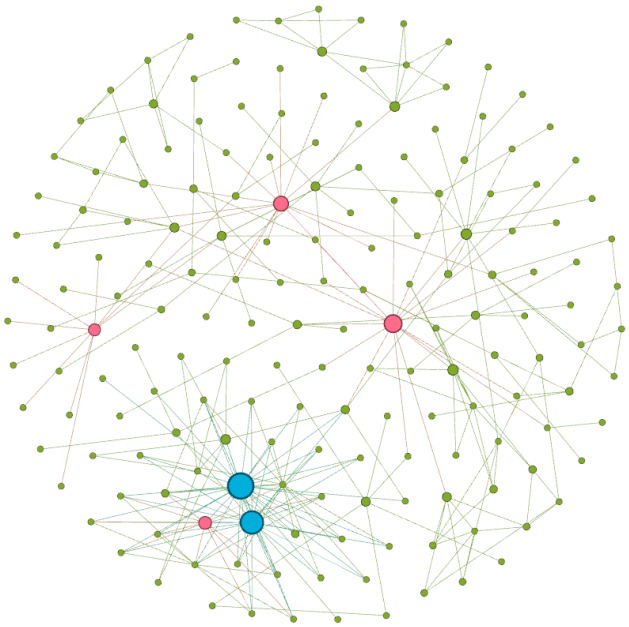

Figure 19 Version 7 test case graph

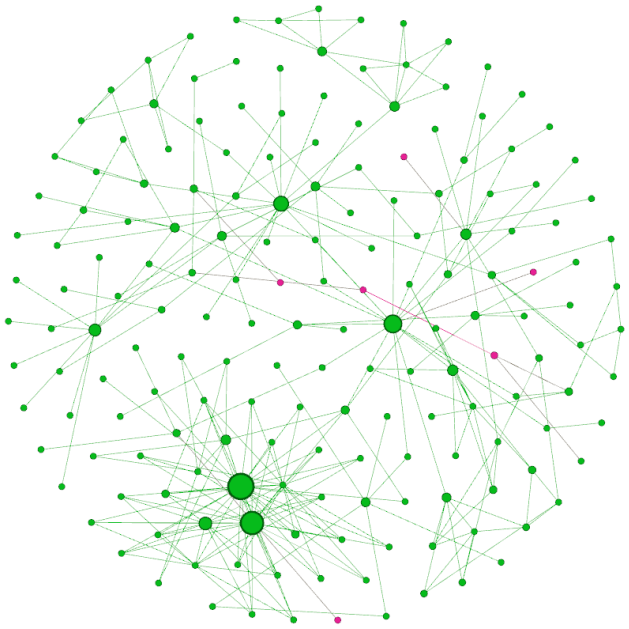

Figure 20 Version 7 Modified and Added function

Table 39 Modify and Add function of version 7

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
Temporary suspended	Modified	0.221106	0	0.011267	0.002565
Order history start date / end date	Modified	0.180328	0	0.004024	0.003186
Sale report start date / end date	Modified	0.183333	0	0.004398	0.002968
Invoice	Modified	0.158702	0.001688	0.006863	0.00529
View invoice	Modified	0.185654	0.01776	0.009021	0.007863
Force close/open delivery	Modified	0.275	0	0.028165	0.002441
Invoice	Modified	0	0	0	0

Table 40 Top 5 Closeness centrality measurement of version 7

Rank	Function	Closeness centrality
1	main	0.378495
2	Setting	0.321755
3	menu	0.315412
4	Reward card	0.314286
5	Restaurant information	0.283414

Table 41 Top 5 Betweenness centrality measurement of version 7

Rank	Function	Betweenness centrality
1	main	0.798277
2	menu	0.360614
3	Setting	0.353442
4	Reward card	0.224903
5	Reward card setting	0.182792

Table 42 Top 5 Eigenvector centrality measurement of version 7

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Add menu	0.918635
3	save menu	0.68662
4	Add new menu option	0.658263
5	save menu option value	0.55548

Table 43 Top 5 PageRank centrality measurement of version 7

Rank	Function	PageRank
1	main	0.031868
2	Edit existing menu	0.029389
3	Setting	0.028055
4	Reward card setting	0.024322
5	Add menu	0.023246

Table 44 Top 10 overall centrality of version 7

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.261905	0.160741	0.029389	1	1.452035
main	0.378495	0.798277	0.031868	0.170117	1.378757
Add menu	0.259205	0.115087	0.023246	0.918635	1.316173
menu	0.315412	0.360614	0.007912	0.288111	0.972049
save menu	0.212304	0.002887	0.016367	0.68662	0.918178
Add new menu option	0.211031	0.000852	0.01173	0.658263	0.881876
Setting	0.321755	0.353442	0.028055	0.105093	0.808345
save menu option value	0.210526	0.000462	0.010063	0.55548	0.776531
Reward card	0.314286	0.224903	0.013683	0.075233	0.628105
Add existing menu option	0.209026	0.000024	0.004933	0.381143	0.595126

4.1.8 Version 8

Figure 21 shows the eighth version test case graph generation using Gephi application. In this version, 11 functions are added to the application and 1 modified function, which shows as a pink node and yellow node in Figure 22 respectively, and detail provided in table 45. Table 46, Table 47, Table 48, and Table 49 shows top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. The result of the new function is as same as version 5, 6, and 7 which it is does not affect the overall measurement value in the application and lead to the remainder of function prioritize orders.

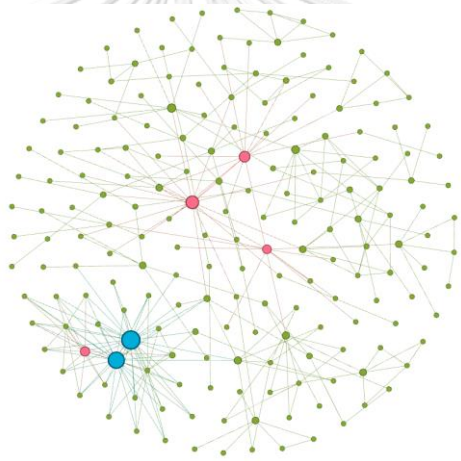

Figure 21 Version 8 test case graph

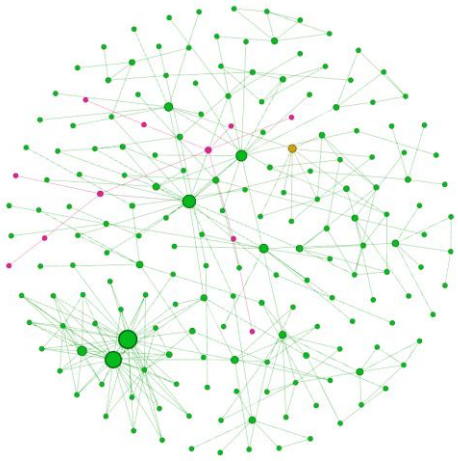

Figure 22 Version 8 Modified and Added function

Table 45 Modify and Add function of version 8

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
Restaurant safety	Modified	0.228433	0.061046	0.05519	0.011477
Deal	Add	0.279346	0.102571	0.038132	0.009005
Redeem deal	Add	0.219114	0.010638	0.009473	0.005293
Report deal	Add	0.221698	0.062863	0.017697	0.012394
Scan qr code	Add	0.218605	0	0.007636	0.002706
View redeem deal	Add	0.179904	0	0.003026	0.003044
Select date deal report	Add	0.181643	0	0.005214	0.002902
Select branch deal report	Add	0.181643	0	0.005214	0.002902
Redeemed	Add	0.181994	0.010638	0.007037	0.0056
See all others deal	Add	0.181994	0.010638	0.007037	0.0056
Redeemed information	Add	0.154098	0	0.00281	0.003172
Other deal detail	Add	0.154098	0	0.00281	0.003172

Table 46 Top 5 Closeness centrality measurement of version 8

Rank	Function	Closeness centrality
1	Main	0.371542
2	Setting	0.313333
3	Menu	0.307692
4	Reward card	0.306688
5	Restaurant information	0.285281

Table 47 Top 5 Betweenness centrality measurement of version 8

Rank	Function	Betweenness centrality
1	Main	0.816873
2	Menu	0.343239
3	Setting	0.333542
4	Reward card	0.212055
5	Reward card setting	0.171749

Table 48 Top 5 Eigenvector centrality measurement of version 8

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Add menu	0.918635
3	Save menu	0.686669
4	Add new menu option	0.658311
5	Save menu option value	0.555519

Table 49 Top 5 PageRank centrality measurement of version 8

Rank	Function	PageRank
1	Main	0.031009
2	Edit existing menu	0.027559
3	Setting	0.026415
4	Reward card setting	0.022809
5	Add menu	0.021798

Table 50 Top 10 overall centrality of version 8

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.255088	0.151406	0.027559	1	1.434053
Main	0.371542	0.816873	0.031009	0.164386	1.38381
Add menu	0.252688	0.108678	0.021798	0.918635	1.301799
Menu	0.307692	0.343239	0.007481	0.287397	0.945809
Save menu	0.207506	0.002529	0.015341	0.686669	0.912045
Add new menu option	0.206367	0.000746	0.010995	0.658311	0.876419
Setting	0.313333	0.333542	0.026415	0.104227	0.777517
Save menu option value	0.205915	0.000405	0.009432	0.555519	0.771271
Reward card	0.306688	0.212055	0.012923	0.074408	0.606074
Add existing menu option	0.20457	0.000021	0.004624	0.381167	0.590382

4.1.9 Version 9

Figure 23 shows the ninth version test case graph generation using Gephi application. In this version, 12 functions are added to the application and 5 modified functions, which shows as a pink node and yellow node in Figure 24 respectively, and detail provided in Table 51. Table 52, Table 53, Table 54, and Table 55 shows the top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. The result shows that there are orders swapping between each rank compared to the version before. However, the list of top functions that got the highest remains the same.

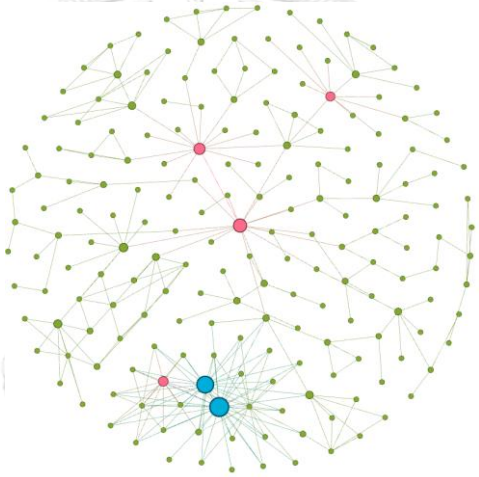

Figure 23 Version 9 test case graph

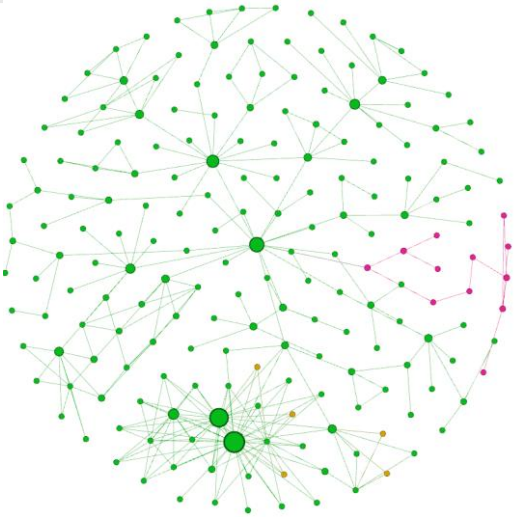

Figure 24 Version 9 Modified and Added function

Table 51 Modify and Add function of version 9

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
Add menu name (Primary)	Modified	0.199586	0.000006	0.278062	0.003003
Add menu name (TH)	Modified	0.199586	0.000006	0.278062	0.003003
Add menu name (EN)	Modified	0.199586	0.000006	0.278062	0.003003
Input name for new group	Modified	0.194949	0	0.043414	0.002791
Edit existing menu group	Modified	0.194949	0	0.043414	0.002791
Menu promotion	Add	0.272984	0.109348	0.034061	0.006359
Create menu promotion	Add	0.218079	0.070272	0.009279	0.004575
See menu promotion	Add	0.215642	0.020671	0.010576	0.007501
Edit menu promotion	Add	0.177553	0	0.003493	0.0029
Set menu promotion as inactive	Add	0.177553	0	0.003493	0.0029
Menu discounts	Add	0.181221	0.060557	0.005679	0.00471
Select menu	Add	0.154771	0.050734	0.006112	0.004684
Menu to discount	Add	0.134871	0.040803	0.009552	0.006742
Search menu to discount	Add	0.119136	0	0.007498	0.004674
Create discount on menu	Add	0.119283	0.020671	0.010545	0.009369
Start/end date promotion discount on menu	Add	0.10663	0	0.003956	0.002764
Save create promotion discount on menu	Add	0.10663	0	0.003956	0.002764

Table 52 Top 5 Closeness centrality measurement of version 9

Rank	Function	Closeness centrality
1	Main	0.359404
2	Setting	0.303459
3	Menu	0.2983
4	Reward card	0.297381
5	Restaurant information	0.277698

Table 53 Top 5 Betweenness centrality measurement of version 9

Rank	Function	Betweenness centrality
1	Main	0.823834
2	Menu	0.336434
3	Setting	0.325885
4	Reward card	0.207119
5	Reward card setting	0.16753

Table 54 Top 5 Eigenvector centrality measurement of version 9

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Add menu	0.918635
3	Save menu	0.68666
4	Add new menu option	0.658302
5	Save menu option value	0.555512

Table 55 Top 5 PageRank centrality measurement of version 9

Rank	Function	PageRank
1	Main	0.030269
2	Edit existing menu	0.026851
3	Setting	0.025741
4	Reward card setting	0.022223
5	Add menu	0.021238

Table 56 Top 10 overall centrality of version 9

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.248072	0.147826	0.026851	1	1.422749
Main	0.359404	0.823834	0.030269	0.165	1.378507
Add menu	0.24586	0.106209	0.021238	0.918635	1.291942
Menu	0.2983	0.336434	0.007292	0.287505	0.929531
Save menu	0.202731	0.002399	0.014946	0.68666	0.906736
Add new menu option	0.201672	0.000708	0.010712	0.658302	0.871394
Save menu option value	0.201251	0.000384	0.009189	0.555512	0.766336
Setting	0.303459	0.325885	0.025741	0.104365	0.75945
Reward card	0.297381	0.207119	0.012596	0.074537	0.591633
Add existing menu option	0.2	0.00002	0.004505	0.381162	0.585687

4.1.10 Version 10

Figure 25 shows the tenth version test case graph generate using Gephi application. In this version, there are 5 modified functions, which shows as a pink node in Figure 26 and detail provided in Table 57, Table 58, Table 59, Table 60, and Table 61 shows the top 5 Function that got the highest centrality from Closeness centrality, Betweenness centrality, Eigenvector centrality, PageRank, respectively. The overall function prioritize order remains the same.

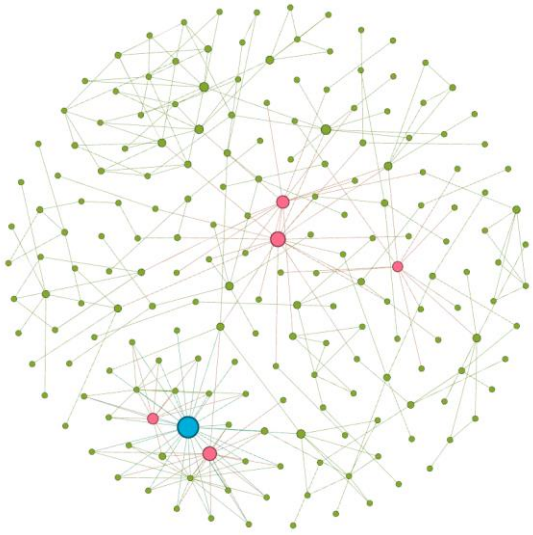

Figure 25 Version 10 test case graph

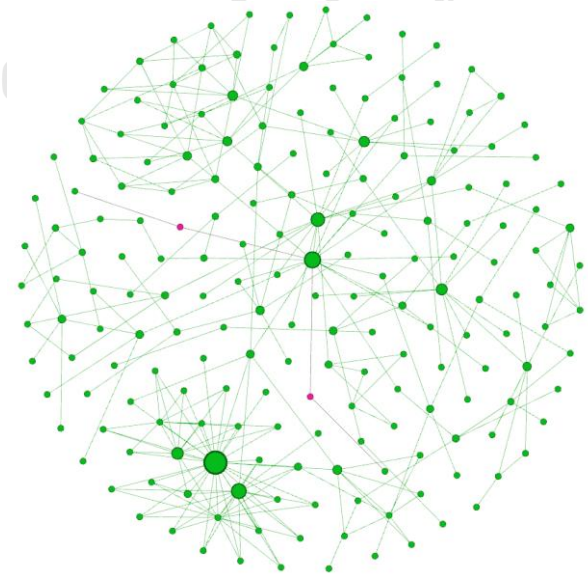

Figure 26 Version 10 Modified and Added function

Table 57 Modify and Add function of version 10

Function	Add / Modified	Closeness centrality	Betweenness centrality	Eigenvector centrality	PageRank
Announcement	Modified	0.26539	0.010309	0.039067	0.004593
Banner	Modified	0.26539	0.010309	0.039067	0.004593

Table 58 Top 5 Closeness centrality measurement of version 10

Rank	Function	Closeness centrality
1	Main	0.359259
2	Setting	0.303125
3	Menu	0.298921
4	Reward card	0.29709
5	Restaurant information	0.277539

Table 59 Top 5 Betweenness centrality measurement of version 10

Rank	Function	Betweenness centrality
1	Main	0.82322
2	Menu	0.341798
3	Setting	0.324395
4	Reward card	0.206159
5	Edit existing menu	0.191312

Table 60 Top 5 Eigenvector centrality measurement of version 10

Rank	Function	Eigenvector centrality
1	Edit existing menu	1
2	Save menu	0.716377
3	Add menu	0.566153
4	Add new menu option	0.522475
5	Save menu option value	0.407578

Table 61 Top 5 PageRank centrality measurement of version 10

Rank	Function	PageRank
1	Main	0.030203
2	Edit existing menu	0.029411
3	Setting	0.02562
4	Reward card setting	0.022111
5	Restaurant information	0.016844

Table 62 Top 10 overall centrality of version 10

Function	Closeness centrality	Betweenness centrality	PageRank	Eigenvector centrality	total
Edit existing menu	0.248399	0.191312	0.029411	1	1.469122
Main	0.359259	0.82322	0.030203	0.210619	1.423301
Save menu	0.203354	0.002485	0.016097	0.716377	0.938313
Menu	0.298921	0.341798	0.007644	0.280184	0.928547
Add menu	0.239802	0.070936	0.015946	0.566153	0.892837
Setting	0.303125	0.324395	0.02562	0.137207	0.790347
Add new menu option	0.201873	0.001092	0.011433	0.522475	0.736873
Save menu Option value	0.201036	0.000499	0.009761	0.407578	0.618874
Reward card	0.29709	0.206159	0.01254	0.097935	0.613724
Add existing menu option	0.20062	0.000259	0.004987	0.365184	0.57105

4.2 Test case fault detection

In this section of the experiment, the Average Percentage of Faults Detected (APFD) is introduced as a measurement. Which measures the weighted average of the percentage of faults detected over the life of the test suite. APFD values range from 0 to 1; higher numbers imply better fault detection rates. The method is constructed as shown in Equation (6).

$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{nm} + \frac{1}{2n} \quad (6)$$

Let T be a test suite containing n test cases and let F be a set of m faults revealed by T. Let TF_i be the first test case in ordering T' of T which reveals fault i.

Table 63 shows number of defects (m) in each version and number of test case (n) which will be use in our experiment in the following section.

Table 63 Number of defects in each version

Version	Number of defects (m)	Number of Test case (n)
1	0	78
2	2	78
3	1	83
4	4	85
5	3	89
6	4	93
7	1	93
8	4	99
9	1	104
10	3	104

4.2.1 Version 2

We used results from each centrality to prioritize our test case Figure 27 shows the step of finding faults detected versus the test suite in version 2. While Figure 28 shows APFD results of each centrality. In this version, non-prioritized giving the best output comparing to another centrality.

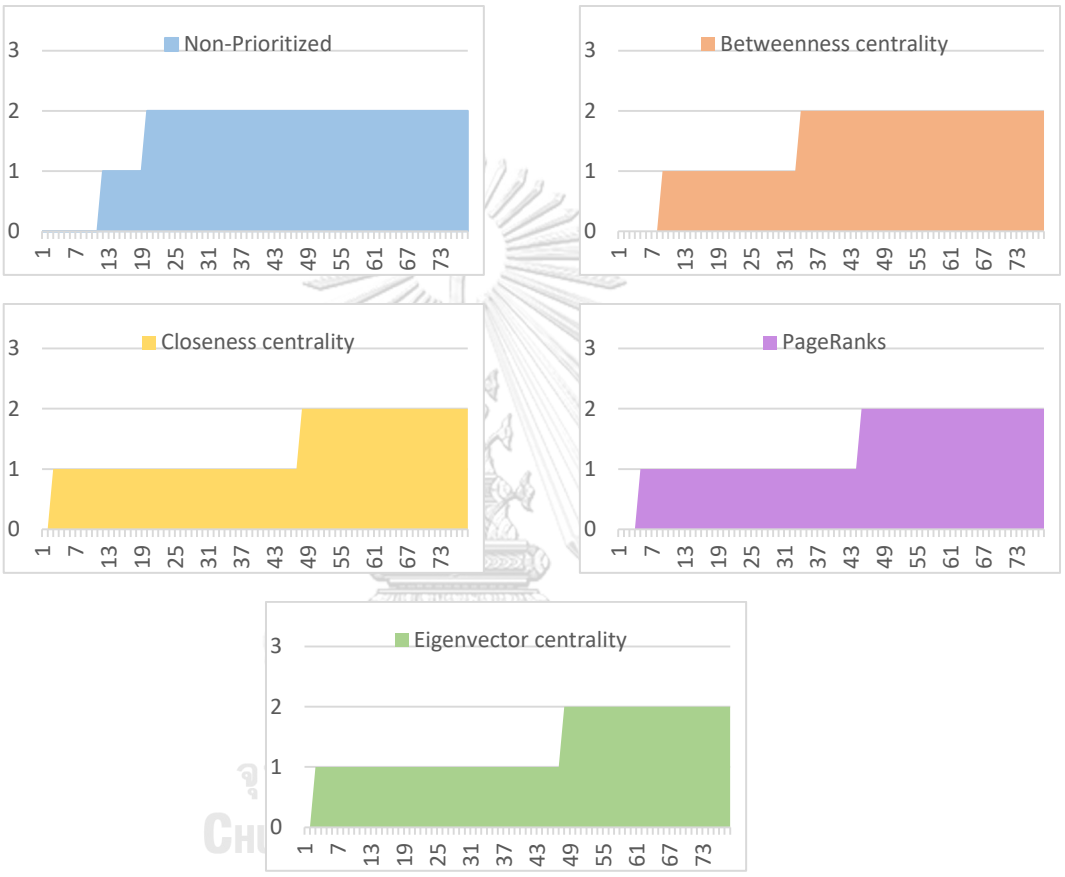

Figure 27 Test suite version 2

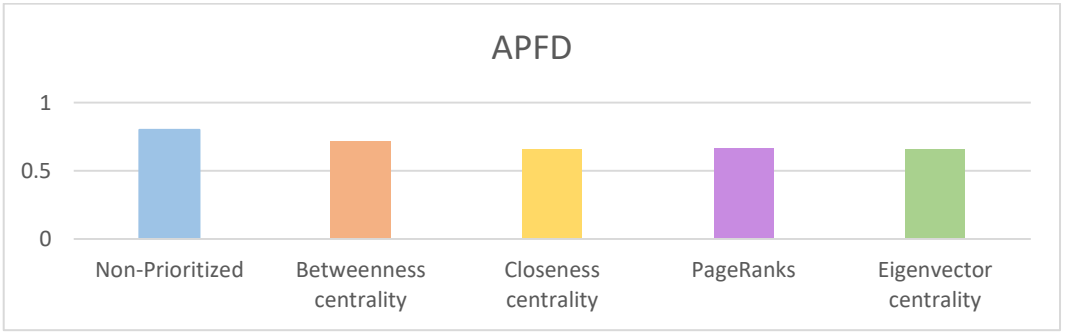

Figure 28 APFD version 2

4.2.2 Version 3

Figure 29 shows the step of finding faults detected versus the test suite in version 3. We used results from each centrality to prioritize our test case and found the best APFD results of each centrality as shows in Figure 30. In this version, Closeness centrality giving the best output comparing to another centrality.

Figure 29 Test suite version 3

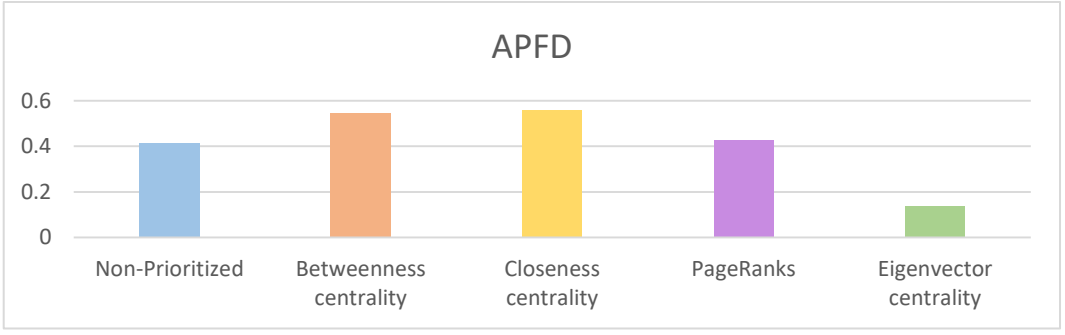

Figure 30 APFD version 3

4.2.3 Version 4

Figure 31 shows the step of finding faults detected versus test suite in version 4. We used results from each centrality to prioritize our test case and found the best APFD results of each centrality as shows in Figure 32. In this version, Eigenvector centrality giving the best output comparing to another centrality.

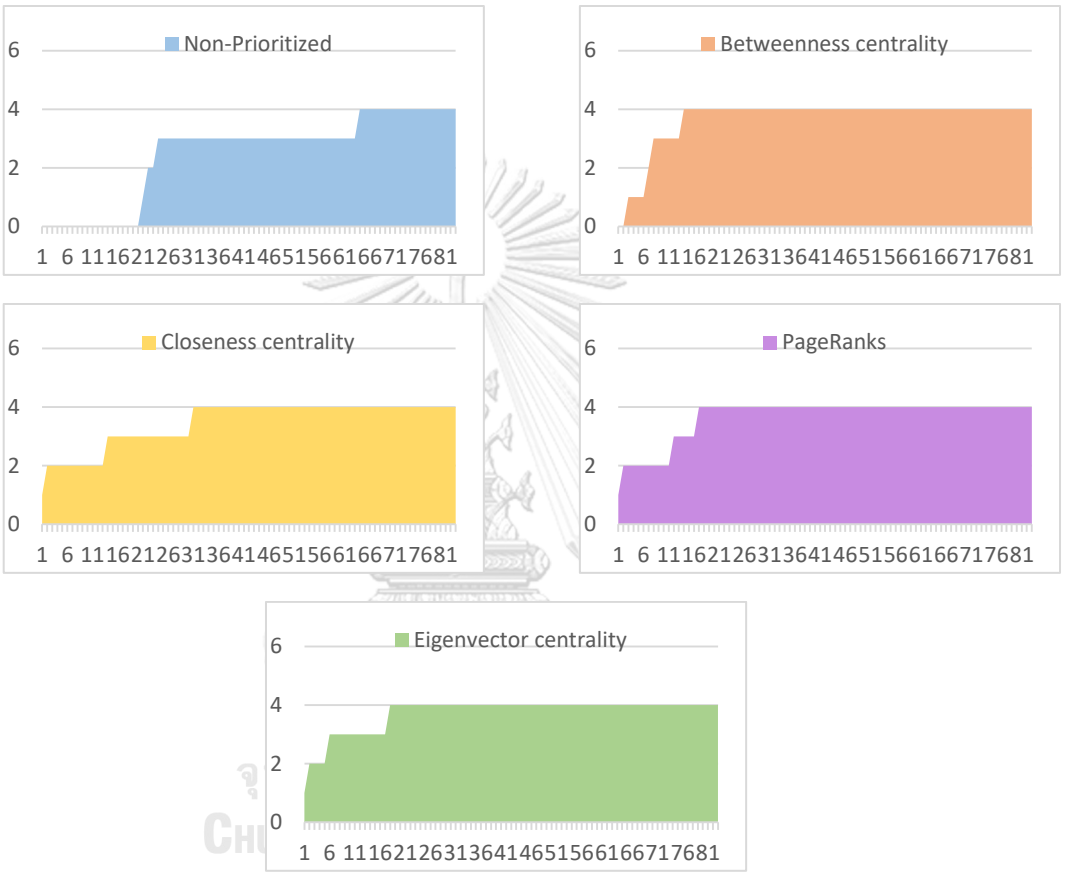

Figure 31 Test suite version 4

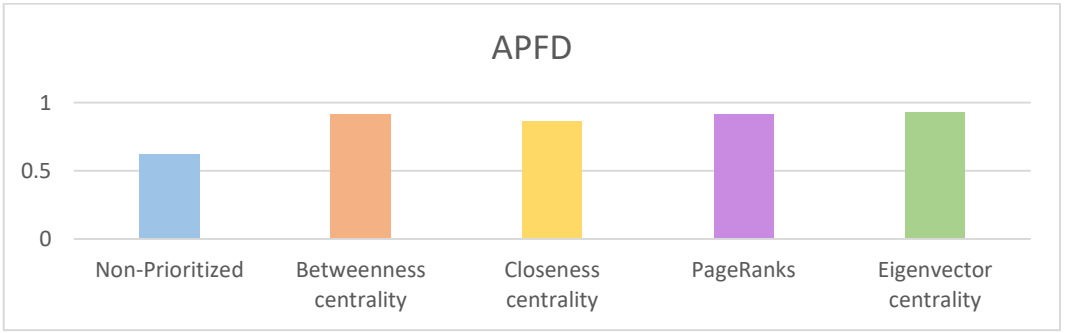

Figure 32 APFD version 4

4.2.4 Version 5

Figure 33 shows the step of finding faults detected versus the test suite in version 5. We used results from each centrality to prioritize our test case and found the best APFD results of each centrality as shows in Figure 34. In this version, Betweenness centrality giving the best output comparing to another centrality.

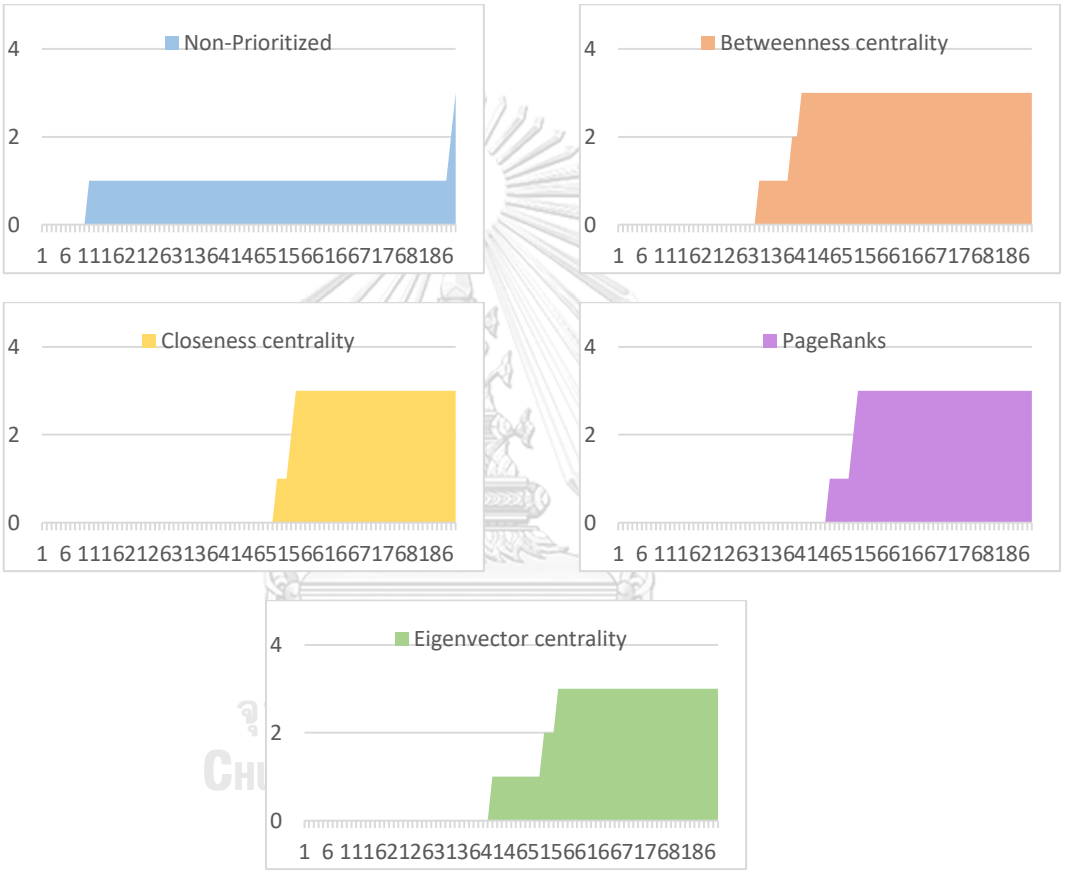

Figure 33 Test suite version 5

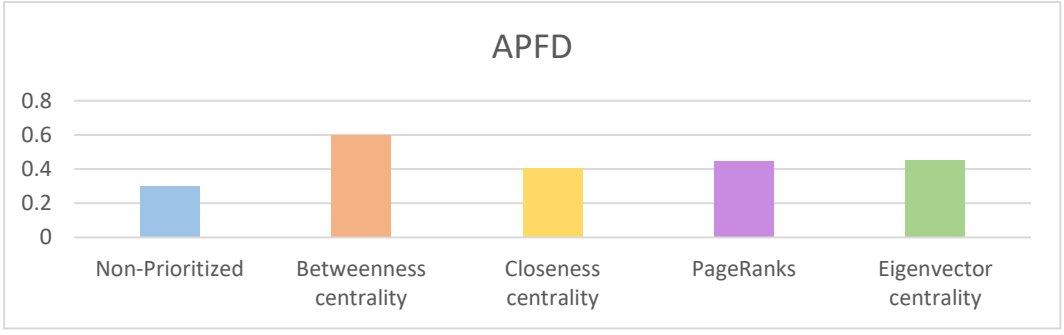

Figure 34 APFD version 5

4.2.5 Version 6

Figure 35 shows the step of finding faults detected versus the fraction of the test suite in version 6. We used results from each centrality to prioritize our test case and found the best APFD results of each centrality as shows in Figure 36. In this version, Betweenness centrality giving the best output comparing to another centrality.

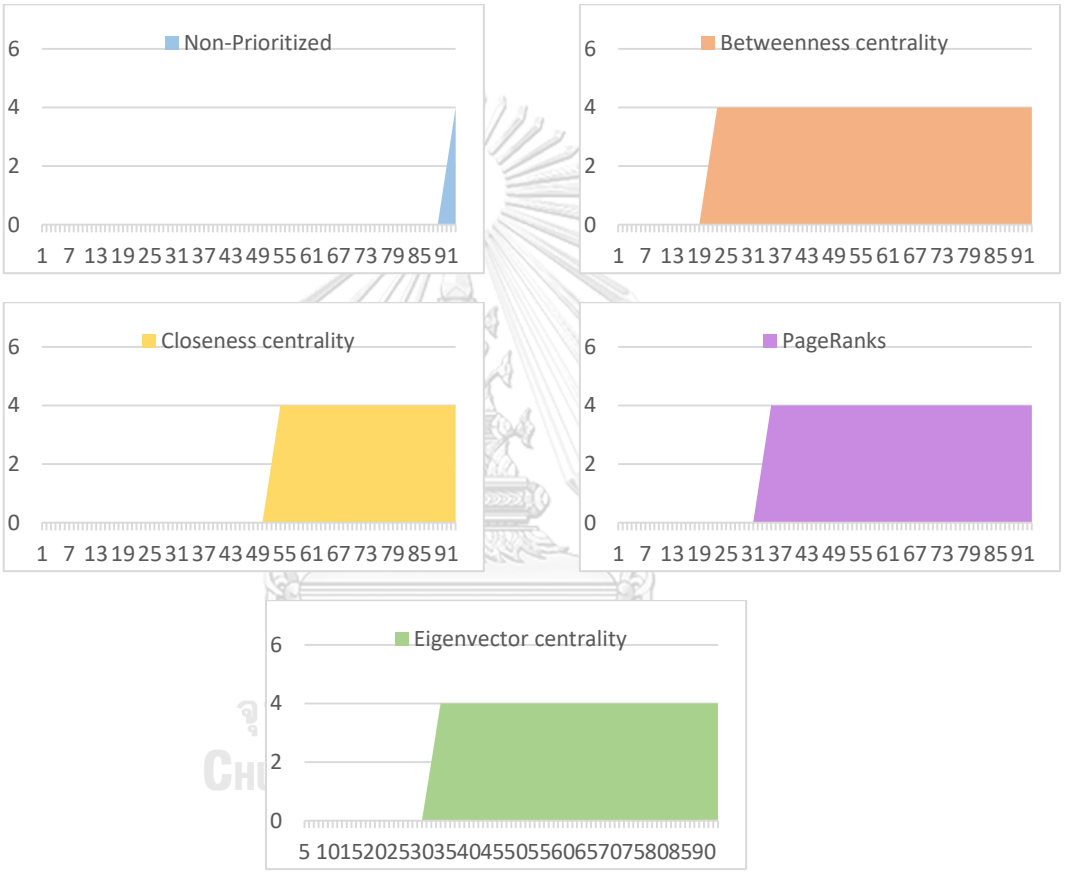

Figure 35 Test suite version 6

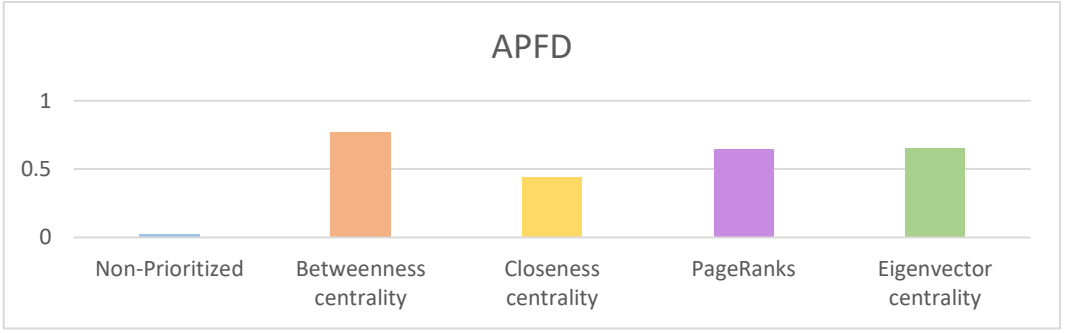

Figure 36 APFD version 6

4.2.6 Version 7

Figure 37 shows the step of finding faults detected versus the fraction of the test suite in version 7. We used results from each centrality to prioritize our test case and found the best APFD results of each centrality as shows in Figure 38. In this version, Betweenness centrality giving the best output comparing to another centrality.

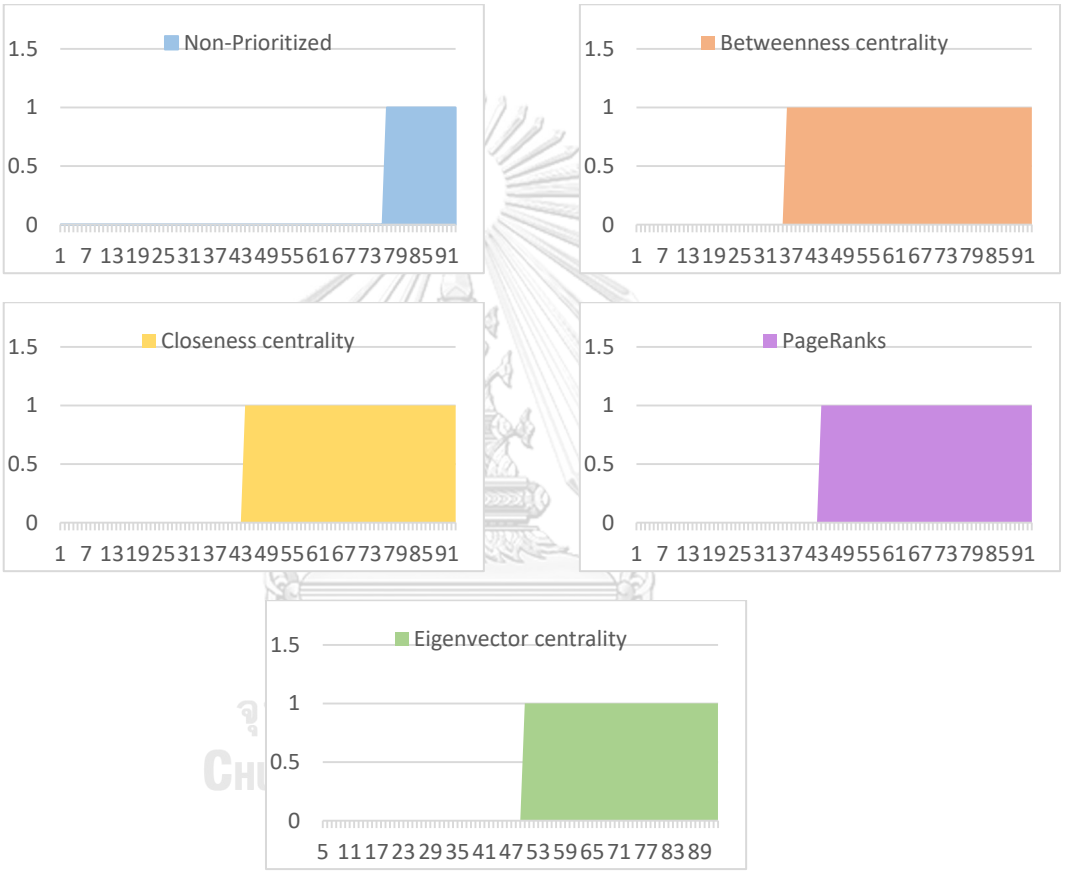

Figure 37 Test suite version 7

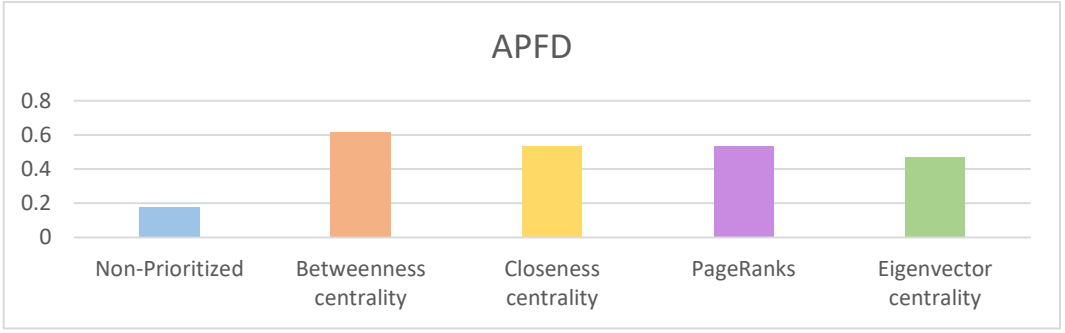

Figure 38 APFD version 7

4.2.7 Version 8

Figure 39 shows the step of finding faults detected versus the fraction of the test suite in version 8. We used results from each centrality to prioritize our test case and found the best APFD results of each centrality as shows in Figure 40. In this version, Betweenness centrality giving the best output comparing to another centrality.

Figure 39 Test suite version 8

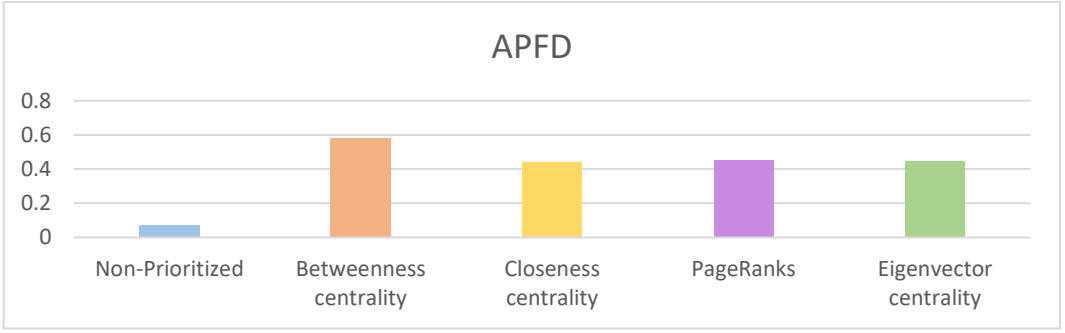

Figure 40 APFD version 8

4.2.8 Version 9

Figure 41 shows the step of finding faults detected versus the fraction of the test suite in version 9. We used results from each centrality to prioritize our test case and found the best APFD results of each centrality as shows in Figure 42. In this version, Betweenness centrality giving the best output comparing to another centrality.

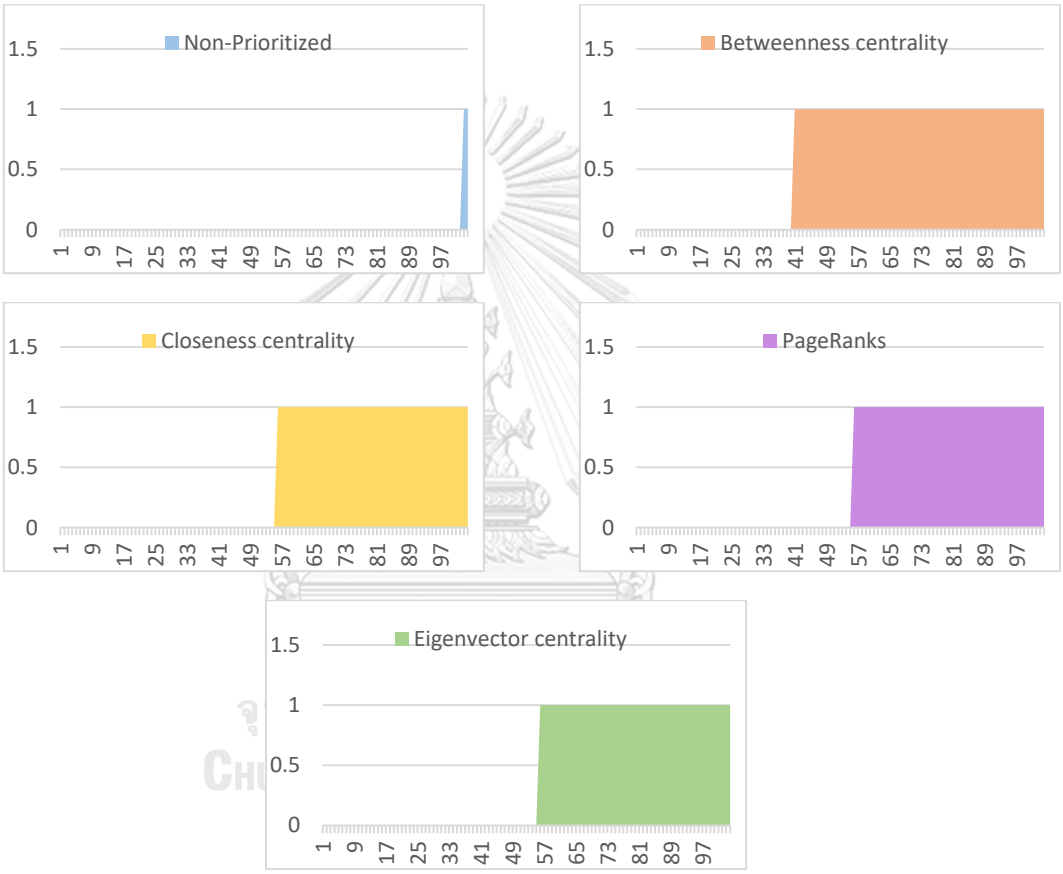

Figure 41 Test suite version 9

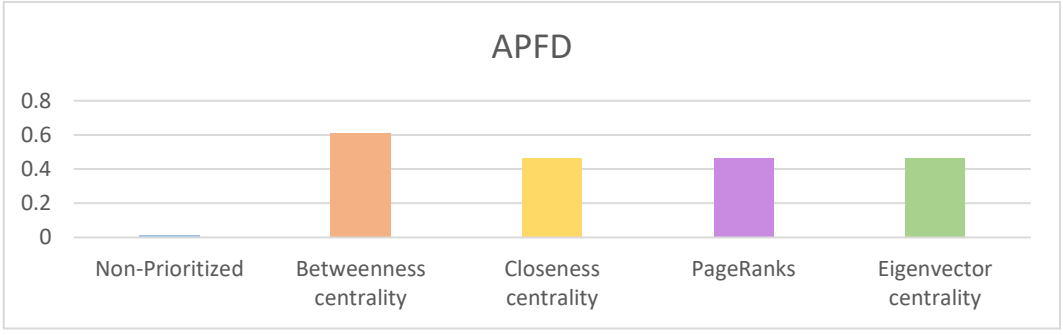

Figure 42 APFD version 9

4.2.9 Version 10

Figure 43 shows the step of finding faults detected versus the fraction of the test suite in version10. We used results from each centrality to prioritize our test case and found the best APFD results of each centrality as shows in Figure 44. In this version, Betweenness centrality giving the best output comparing to another centrality.

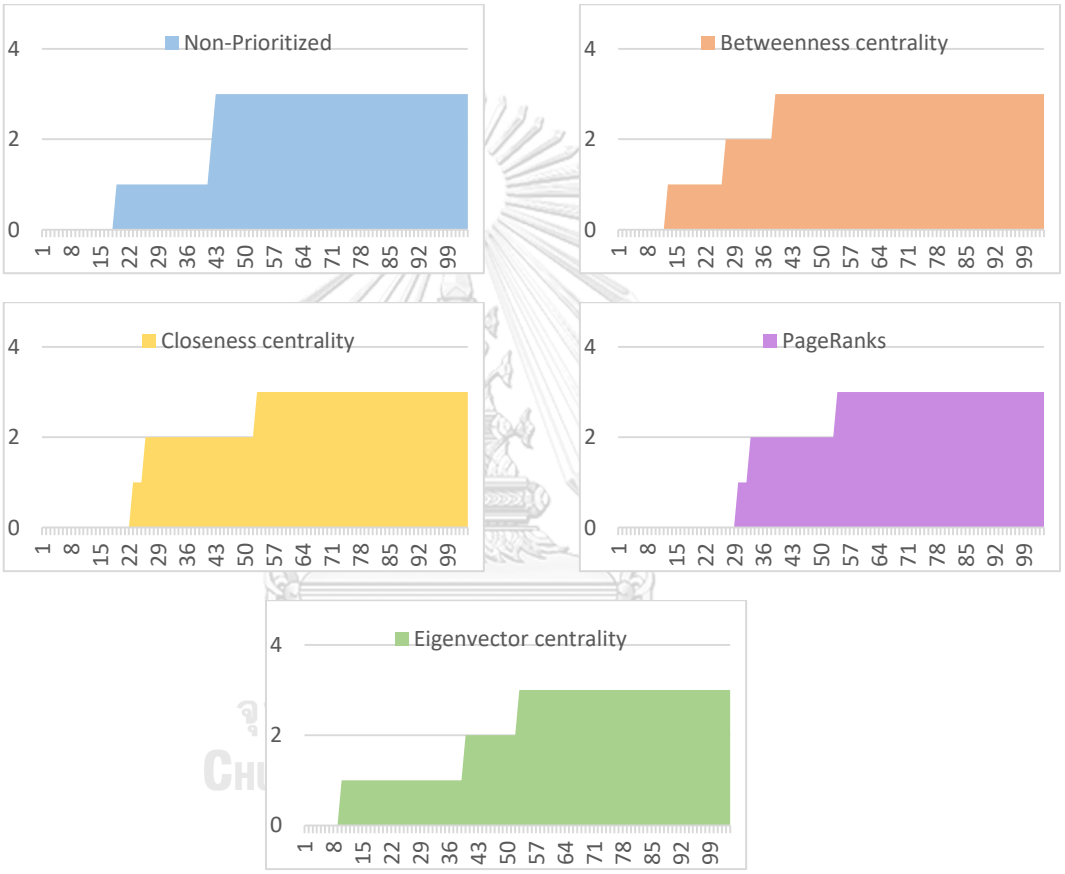

Figure 43 Test suite version 10

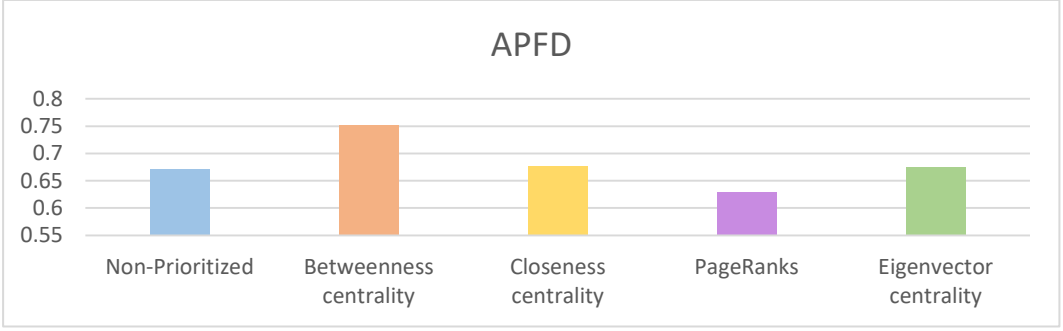

Figure 44 APFD version 10

4.3 Summary

After we given a deep analyze of each centrality over Wongnai RMS application version in the last section, we found that in each version there are difference measurement centrality that give us the best results as shows in Table64. However, the centrality that got a potential and can help us find the earlier the defect in our application is Betweenness centrality because it has been given as number 1 or number 2 in every version of our test.

Table 64 The best centrality for fault detection

version	Number 1 Centrality	Number 2 Centrality
1	-	-
2	Non-Prioritized	Betweenness centrality
3	Closeness centrality	Betweenness centrality
4	Eigenvector centrality	Betweenness centrality
5	Betweenness centrality	Eigenvector centrality
6	Betweenness centrality	Eigenvector centrality
7	Betweenness centrality	PageRank
8	Betweenness centrality	PageRank
9	Betweenness centrality	Eigenvector centrality
10	Betweenness centrality	Closeness centrality

Chapter 5

Conclusion

This research aimed to apply network centrality methods of social network analysis to find the most valuable functions in the application and the best centrality that can prioritize test cases and increase fault detection in earlier stage.

We evaluate the important function in the application by combining the importance of each function in the network by using social network analysis measurement and using it to prioritize the function. The reason we combine all the measurement together because we want to find the most importance node that the best node in every dimension. A node that got high value means that it is an important function in the application which might cause the issue to the whole system if it has an issue but couldn't catch by the tester earlier before roll out to end-use or even worse, it gets an error in end-user hand which will cause both application reliability and customer satisfaction. After knowing the most valuable function, by considering a test case that consisted of the top function can lead the tester to be aware of the defect that can destroy the overall program. After considering application release over 10 versions, the experimental result shows that by using social network measurement, the core function of the application is not changed easily over time because it got a complex structure and relation to other functions in the network. But the priority order affected at the minor level.

Another output we got from our experiment is the best measurement help tester catch defect earlier in each test cycle is Betweenness centrality. Betweenness centrality measurement shows that it always be number1 or number2 in all of our experiments and it will help Wongnai Tester to be able to find the defect in each test cycle earlier and be able to provide feedback to programmer to solve the issue as soon as possible, which will help launch the program faster than non-prioritized testing.

REFERENCES

[1] Khatibsyarbini M, Adham Isa M, Jawawi D N.A, and Tumeng R,“Test Case Prioritization Approaches in regression testing: A systematic literature review,” in Information and software Technology, January 2018, pp 74-93

[2] Shin Y, and Harman M, “Regression Testing Minimization, Selection, and Prioritization: A Survey," Software Testing, Verification and Reliability 22.2 (2012): 67-120.

[3] Mini tools, Retrieved April 11, 2020, from minitool.com website: https://www.minitool.com/lib/graphical-user-interface.html

[4] Sangwan S, "Software Testing Techniques and Strategies," 2014 Journal of Engineering Research and Applications, April 2014, pp.99-102

[5] Rena I and Rena E, “Study Paper on Test Case Generation for GUI Based Testing,” in International Journal of Software Engineering & Application (IJSEA), January 2012

[6] Sharma A, Patani R, Aggarwal A, “Software Testing Using Genetic Algorithm,” 2016

[7] Sun W, Gao Z, Yang W, Fang C, Chen Z, “Multi-Objective Test Case Prioritization for GUI Application”

[8] Ren Y, Yin B, and Wang B, "Test Case Prioritization for GUI Regression Testing Based on Centrality Measures," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, 2018, pp. 454-459.

[9] Harrold et al., Wong et al., “Test case Reduction,” 1993

[10] Wang B, Yin B, and Cai K “Event Handler Tree Model for GUI Test Case Generation,” 2016 IEEE 40nd Annual Computer Software and Application Conference

[11] Bloch F, Jackson M, and Tebaldi P, “Centrality Measure in Network,” (2019)

[12] Borgatti S, “Centrality and network flow,” 2005 Social Networks, 27, pp 55-77

[13] Riquelme F, Gonzalez-Cantergiani P, Molinero X and Serna M, “Centrality measure in social networks based on linear threshold model,” 2017

[14] Anda B, Dreiem H, Sjøberg D. I.K and Jørgensen M, “Estimation Software Development Effort based on Use- Case-Experiences from Industry,” 2001 Norway

[15] Nageswaran S “Test Effort Estimation Using Use Case Points,” 2001 Quality Week, San Francisco California, June 2001

VITA

NAME	Chawannut Maitrikul
DATE OF BIRTH	18 December 1994
PLACE OF BIRTH	Yala
INSTITUTIONS ATTENDED	Computer Engineering KMUTT
HOME ADDRESS	285/473 The Presient Condo2 Bangkok 10160
PUBLICATION	C Maitrikul and Y Limpiyakorn, "GUI Test case Proritization using Social Network Analysis", International Conference on Computer and Electronical Enginerring (ICCEE 2020)