การประยุกต์การวิเคราะห์เครือข่ายสังคมเพื่อปรับปรุงกระบวนการทดสอบซอฟต์แวร์

น.ส.พรรณธิภา บุญมาพบ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

Applying Social Network Analysis for Software Test Process Improvement

Miss Pantipa Bunmapob

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2021

หัวข้อวิทยานิพนธ์	การประยุกต์การวิเคราะห์เครือข่ายสังคมเพื่อปรับปรุงกระบวนการทดสอบซอฟต์แวร์
โดย	น.ส.พรรณธิภา บุญมาพบ
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์)

..... กรรมการ
(อาจารย์ ดร.พิตติพล คันธวัฒน์)

..... กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต)

พรรณธิภา บุญมาพบ : การประยุกต์การวิเคราะห์เครือข่ายสังคมเพื่อปรับปรุงกระบวนการทดสอบซอฟต์แวร์. (Applying Social Network Analysis for Software Test Process Improvement) อ.ที่ปรึกษาหลัก : รศ. ดร.ญาใจ ลิ่มปิยะกรณ์

Jira Software เป็นโซลูชันการจัดการโครงการแบบอไจล์ ซึ่งเดิมออกแบบมาให้เป็นเครื่องมือในการติดตามข้อบกพร่องและปัญหาที่เกิดขึ้นภายในโครงการ การค้นหาปัญหาหรือข้อมูลข้อบกพร่องสามารถทำได้โดยใช้ Jira Query Language (JQL) อย่างไรก็ตาม การสืบค้นปัญหาหรือข้อบกพร่องจากแหล่งที่เก็บข้อมูลจะคืนค่าข้อมูลที่เฉพาะเจาะจงมาอย่างง่ายและธรรมดาทั่วไป ในงานวิจัยนี้ ได้นำเสนอแนวทางการสร้างภาพข้อมูลเครือข่ายเพื่อเปิดเผยความสัมพันธ์และการสื่อสารระหว่างตัวบทบาท เช่น คุณสมบัติของซอฟต์แวร์ ข้อบกพร่อง และบุคลากร โดยเทคนิคการวิเคราะห์เครือข่ายโซเชียลใช้สำหรับวิเคราะห์ข้อบกพร่องที่รวบรวมจากโครงการซอฟต์แวร์ภายในธนาคาร และ Gephi ถูกใช้เป็นเครื่องมือในการสร้างเครือข่ายของบทบาทที่ระบุเป็นโหนดและการเชื่อมโยงของโหนดเหล่านั้น ซึ่งแนวทางของการวิเคราะห์เครือข่ายภาพนั้นใช้ได้จริงและให้ข้อมูลเชิงลึกในการวิเคราะห์ข้อบกพร่องที่จำเป็นสำหรับกระบวนการพัฒนาซอฟต์แวร์ในเชิงรุก

สาขาวิชา	วิศวกรรมซอฟต์แวร์	ลายมือชื่อนิสิต .....
ปีการศึกษา	2564	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6370182521 : MAJOR SOFTWARE ENGINEERING

KEYWORD: social network analysis, network centrality, defect analysis, software testing, process improvement

Pantipa Bunmapob : Applying Social Network Analysis for Software Test Process Improvement. Advisor: Assoc. Prof. Yachai Limpiyakorn, Ph.D.

Jira software is agile project management solutions, originally designed as a bug and issue tracker. Searching for issues or defect information can be performed by Jira Query Language (JQL). However, querying a data repository simply returns the specified data and information. In this work, we present an approach of network visualization to uncover relationships and communications among actors such as software features, defects, and staff. The technique of social network analysis is applied for analyzing the defects collected from a banking software project. And Gephi is used as a tool for generating a network of specified actors as nodes and their linkages. The approach of visual network analysis is practical and provides insights in defect analysis essential for the proactive software development process.

Field of Study: Software Engineering Student's Signature .....

Academic Year: 2021 Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จสมบูรณ์ได้ด้วยดีเพราะได้รับความกรุณาอย่างดียิ่งจาก รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ประสิทธิ์ประสาทความรู้ สละเวลาอันมีค่าให้คำแนะนำและคำปรึกษา คอยผลักดัน ติดตามความก้าวหน้า และตรวจสอบแก้ไขข้อบกพร่องของงานวิจัยเป็นอย่างดียิ่งมาโดยตลอด ทำให้งานวิจัยนี้สำเร็จลุล่วงไปได้ด้วยดี ผู้วิจัยขอกราบขอบพระคุณด้วยความเคารพอย่างสูงมา ณ โอกาสนี้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ดร.สุกรี สินธุภิญโญ ประธานกรรมการสอบวิทยานิพนธ์ อาจารย์ดร.พิตติพล คันธวัฒน์ และ อาจารย์ดร.ภาสกร อภิรักษ์วรพินิต กรรมการสอบวิทยานิพนธ์ที่กรุณาเสียสละเวลาอันมีค่า ในการตรวจสอบและให้คำแนะนำที่เป็นประโยชน์ในการทำวิทยานิพนธ์ในครั้งนี้

ขอขอบพระคุณบิดามารดา และญาติพี่น้องที่ได้ให้การสนับสนุน คอยเป็นห่วงและเป็นกำลังใจที่ดีเสมอมา และขอขอบคุณเพื่อน ๆ พี่ ๆ น้อง ๆ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกคน ที่คอยสนับสนุน ติดตาม และให้กำลังใจมาโดยตลอด คอยให้คำแนะนำ ความช่วยเหลือ และคำปรึกษาเป็นอย่างดี รวมถึงขอขอบพระคุณท่านผู้เกี่ยวข้องทุกท่าน ที่กรุณาได้ให้คำปรึกษา ซึ่งมิได้กล่าวลงนามไว้ ณ ที่นี้ด้วย และท้ายที่สุดขอบคุณตัวเองที่ไม่ยอมแพ้และไม่ท้อถอย ซึ่งทำให้วิทยานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี ผู้วิจัยหวังเป็นอย่างยิ่งว่าวิทยานิพนธ์ฉบับนี้จะเป็นประโยชน์บ้างไม่มากก็น้อยต่อผู้ที่สนใจที่จะศึกษาต่อไปในภายภาคหน้า

พรรณธิภา บุญมาพบ

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบัน ปัญหาในกระบวนการทดสอบซอฟต์แวร์ส่วนใหญ่ คือ การพบข้อบกพร่อง (defect) หรือข้อบกพร่องที่เกิดขึ้นภายหลังจากการทดสอบเสร็จสิ้น หรือเกิดขึ้นล่าช้าเกินกว่าระยะเวลาในการพัฒนาซอฟต์แวร์ที่วางแผนไว้ไปมาก เนื่องจากเป็นข้อบกพร่องที่เกิดจากความต้องการ (requirements) ที่ไม่ครบถ้วนสมบูรณ์ตั้งแต่แรก การเปลี่ยนแปลงความต้องการ (requirements change) หรือการแก้งาน (rework) จากการพบเจอข้อบกพร่องที่มีความรุนแรงสูง หรือเกิดขึ้นในระยะเวลาที่ล่าช้า จะทำให้เกิดค่าใช้จ่ายจำนวนมากในช่วงท้ายของการพัฒนาโปรแกรม ไม่สามารถดำเนินการพัฒนาไปตามระยะเวลาที่กำหนดได้ และส่งผลต่อภาพลักษณ์ในทางลบของบริษัทผู้พัฒนาซอฟต์แวร์

ผู้วิจัยจึงได้เล็งเห็นถึงความสำคัญของการวิเคราะห์หาสาเหตุที่ทำให้เกิดข้อบกพร่อง หรือการเปลี่ยนแปลงความต้องการที่เกิดขึ้นภายหลังจากการเจอข้อบกพร่องที่ไม่พึงประสงค์ ซึ่งล้วนแล้วแต่เกิดมาจากข้อบกพร่องที่หลบหนี (escaped defects) ระหว่างการดำเนินการทดสอบซอฟต์แวร์ทั้งสิ้น เป็นปัญหาที่ทำให้เกิดผลกระทบอย่างมากต่อการพัฒนาซอฟต์แวร์ ระยะเวลา และต้นทุนในการผลิตซอฟต์แวร์ในแต่ละโครงการ ผู้วิจัยจึงมีความคิดที่จะสำรวจและวิเคราะห์สาเหตุของข้อบกพร่องจากระดับความรุนแรง (defect severity) ของข้อบกพร่องที่เคยเกิดขึ้นในอดีต ซึ่งเก็บบันทึกไว้ใน Jira Software มาประมวลผลด้วยเทคนิควิธีการวิเคราะห์สังคมเครือข่าย (Social Network Analysis— SNA) เพื่อนำผลลัพธ์ที่ได้ไปปรับปรุงกระบวนการพัฒนาและกระบวนการทดสอบต่อไป

งานวิจัยนี้ได้นำเสนอการปรับปรุงกระบวนการการทดสอบซอฟต์แวร์ จากการวิเคราะห์หาความสัมพันธ์และความเชื่อมโยงระหว่างกรณีทดสอบและระดับความรุนแรงของข้อบกพร่อง โดยใช้โปรแกรม Gephi ซึ่งเป็นเครื่องมือในการวิเคราะห์เครือข่ายสังคม และใช้ทฤษฎีกราฟรวมทั้งค่าตัววัดเครือข่ายสังคม (Social Network metrics) ในการวิเคราะห์ลักษณะโครงสร้างของเครือข่าย แล้วนำผลการวิเคราะห์ที่ได้ไปใช้ในการปรับปรุงกระบวนการทดสอบซอฟต์แวร์ ป้องกันไม่ให้เกิดข้อบกพร่องเดิมซ้ำ ๆ รวมทั้งเป็นการป้องกันไม่ให้ข้อบกพร่องหลุดไปยังลูกค้าหรือการใช้งานจริง อีกทั้งยังสามารถนำผลการวิเคราะห์ที่ได้ เสนอแนะไปยังทีมพัฒนาเพื่อปรับปรุงคุณภาพผลิตภัณฑ์และความพึงพอใจของลูกค้าได้อีกด้วย

1.2 วัตถุประสงค์

1.2.1 สำรวจวิเคราะห์หาสาเหตุของข้อผิดพลาดจากระดับความรุนแรงของข้อบกพร่องในโครงการซอฟต์แวร์

1.2.2 ใช้ทฤษฎีกราฟและเครื่องมือทางการวิเคราะห์เครือข่ายสังคม เพื่อศึกษาวิเคราะห์ความสัมพันธ์ของปัจจัยที่มีผลต่อระดับความรุนแรงของข้อบกพร่อง ผลการวิเคราะห์ที่ได้คาดว่าจะเป็นประโยชน์ต่อการปรับปรุงกระบวนการทดสอบไม่ให้เกิดข้อบกพร่องซ้ำ

1.3 ขอบเขตการดำเนินงาน

1.3.1 ใช้ข้อมูลข้อบกพร่องจากโครงการซอฟต์แวร์ในภาคธุรกิจธนาคาร ซึ่งเก็บรวบรวมจาก Jira Software ของโครงการ

1.3.2 จำแนกแหล่งที่มาของข้อบกพร่องที่เกิดจากงานพัฒนา (Development task) กับงานทดสอบ (Testing task)

1.3.3 ใช้ Gephi เป็นเครื่องมือวิเคราะห์เครือข่ายสังคม

1.3.4 ใช้ค่าตัววัดเครือข่ายสังคม (Social Network Metric) 3 ค่า คือ Betweenness Centrality Closeness Centrality และ Degree Centrality เพื่อวิเคราะห์และรายงานผลที่ค้นพบ

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาค้นคว้าทฤษฎีและงานวิจัยที่เกี่ยวข้อง

1.4.2 วางแผนการดำเนินงานวิจัย

1.4.3 สำรวจและวิเคราะห์ข้อมูลจากบันทึกเหตุการณ์ของข้อบกพร่องในแต่ละโครงการในอดีต

1.4.4 ศึกษาเครื่องมือที่เหมาะสม สามารถนำมาใช้สร้างแบบจำลองความสัมพันธ์ในงานวิจัย

1.4.5 วิเคราะห์ความเชื่อมโยงและความสัมพันธ์ระหว่างปัจจัยต่างๆ กับระดับความรุนแรงของข้อบกพร่อง จากกราฟเครือข่ายและทฤษฎีกราฟ

1.4.6 เผยแพร่ผลงานทางวิชาการ

1.4.7 สรุปผลงานวิจัยและเรียบเรียงเป็นวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ได้แนวทางวิธีการสำหรับการวิเคราะห์ต้นเหตุของปัญหา (Root Cause Analysis— RCA) ที่ทำให้เกิดข้อบกพร่องที่มีนัยสำคัญ ทั้งนี้ ความสามารถขององค์กรในการวิเคราะห์ต้นเหตุของปัญหา เป็นข้อกำหนดหนึ่งที่สำคัญและพึงมีสำหรับกระบวนการทำงานที่มีวุฒิภาวะความสามารถ ซึ่งจะดำเนินงานเชิงป้องกัน หลีกเลี่ยงการเกิดปัญหาหรือข้อบกพร่องซ้ำซาก ลดการแก้งานที่ไม่จำเป็น

ส่งเสริมประสิทธิภาพการบริหารโครงการซอฟต์แวร์และภาพลักษณ์ขององค์กร

1.6 ลำดับการจัดเรียงวิทยานิพนธ์

เนื้อหาในวิทยานิพนธ์แบ่งออกเป็น 5 บท ได้แก่

บทที่ 1 บทนำ อธิบายถึงที่มาและความสำคัญของปัญหา วัตถุประสงค์ของงานวิจัย ขอบเขตงานวิจัย ประโยชน์ที่คาดว่าจะได้รับ และผลงานที่ได้รับการตีพิมพ์จากวิทยานิพนธ์

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

บทที่ 3 แนวคิดและวิธีการวิจัย

บทที่ 4 การทดลองและผลการทดลอง

บทที่ 5 สรุปผลการวิจัย และแนวทางการวิจัยในอนาคต

1.7 ผลงานที่ได้รับการตีพิมพ์

Bunmapob, P. & Limpiyakorn, Y. (2022). Exploring Defect Data with Network Visualization. 2022 2nd IEEE International Conference on Software Engineering and Artificial Intelligence (SEAI 2022), Xiamen, China.

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 Social Network Analysis (SNA)

การวิเคราะห์เครือข่ายสังคม (Social Network Analysis: SNA) คือ การวิเคราะห์ข้อมูลที่เป็นลักษณะเครือข่ายความเชื่อมโยง ประกอบด้วย จุด (Nodes) และ เส้น (Edges) ที่มีความเชื่อมโยงกัน โดยจุดแทนหน่วยของข้อมูล ส่วนเส้น แทนความสัมพันธ์ของหน่วยของข้อมูล โดยสามารถนำ SNA มาวิเคราะห์ต่อได้ ไม่ว่าจะเป็นในเชิงรูปธรรม อาทิ เครือข่ายของถนนในเมืองที่มี จุด คือ ทางแยก และ เส้น คือ ถนน ที่เชื่อมแต่ละแยกเข้าด้วยกัน ไปจนถึงสิ่งที่เป็นนามธรรม เช่น เครือข่ายความเป็นเพื่อนกันในห้องเรียน ที่มี จุด คือ นักเรียน และ เส้น คือ การเป็นเพื่อนสนิทกัน เป็นต้น โดยบางครั้งสามารถใช้คำว่า “โหนด” แทนคำว่า “จุด”

การวิเคราะห์เครือข่ายสังคมถูกนำมาใช้งานในหลากหลายลักษณะ ที่รู้จักกันดีในทางธุรกิจส่วนใหญ่จะใช้ในการวิเคราะห์ข้อมูลบนสังคมออนไลน์เป็นหลัก เช่น เฟซบุ๊ก (Facebook) ทวิตเตอร์ (Twitter) และอินสตาแกรม (Instagram) เป็นต้น ด้วยเหตุนี้ จึงทำให้มีโปรแกรมและเครื่องมือที่พัฒนาสำหรับการวิเคราะห์เครือข่ายสังคมออนไลน์ออกมาเป็นจำนวนมาก

ลักษณะการวิเคราะห์ข้อมูลของ SNA จะมีเป้าหมายหลักอยู่ 2 ประเภท คือ 1) ค้นหาจุดที่เป็นศูนย์กลาง (Centrality) ของเครือข่ายสังคม และ 2) ค้นหาจุดที่มีอิทธิพลมากที่สุด (Influencers) ในเครือข่ายสังคม นอกจากนี้ ยังสามารถค้นหาความใกล้ชิดของเครือข่าย ความเข้มแข็งของเครือข่าย และอื่น ๆ ได้อีกเป็นจำนวนมาก

2.1.2 Network Centrality

วิธีการในวิเคราะห์ข้อมูลกราฟเครือข่ายสังคม (Social Network graph) คือ การหาค่าที่แสดงถึงการเชื่อมโยงระหว่างจุดหนึ่งกับจุดอื่น ๆ ในเครือข่าย ซึ่งการหาค่า Centrality สามารถทำได้หลายวิธี เช่น

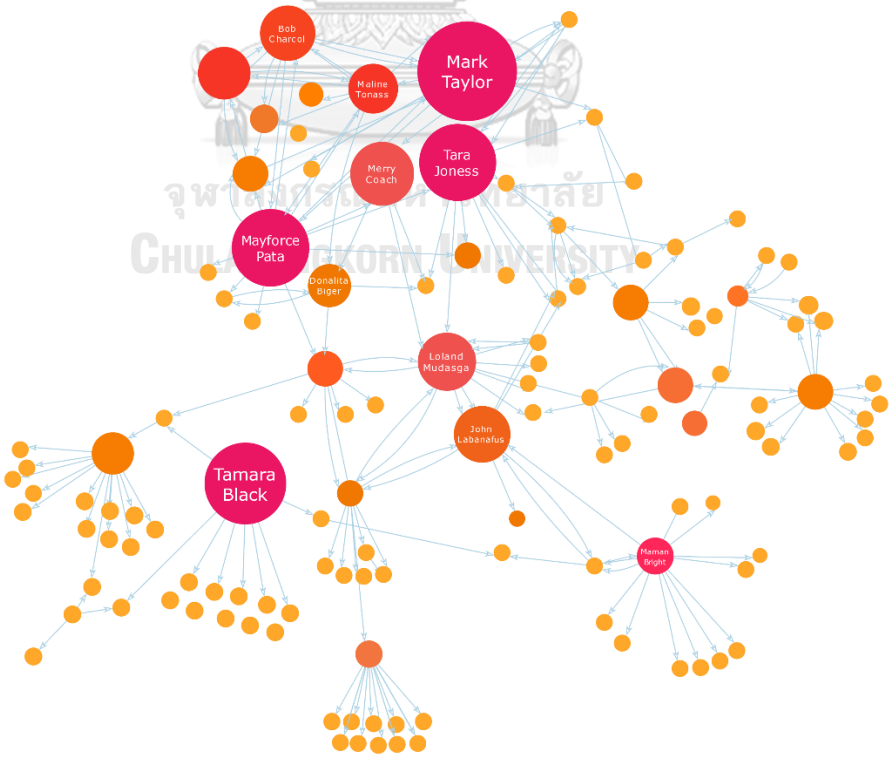
2.1.2.1 *Degree Centrality*

คือ การพิจารณาค่า Degree ของ Node ถ้ามีการเชื่อมโยงของ Degree มาก แสดงว่า Node นั้นจะมีอิทธิพลสูงต่อเครือข่าย Degree Centrality เป็นการวัดค่าความเป็นศูนย์กลางหรือศูนย์รวมในเครือข่าย กล่าวคือ เหล่าศูนย์รวมของกิจกรรมที่มีความเชื่อมโยงเป็นส่วนใหญ่กับศูนย์รวมของกิจกรรมอื่น ๆ กำหนดได้จากการนับจำนวนทิศทางรวมของเส้น

เชื่อมโยงที่เข้ามาสู่ศูนย์รวมของกิจกรรมนั้น ๆ จากศูนย์รวมกิจกรรมอื่นทั้งหมดในเครือข่าย ศูนย์รวมกิจกรรมที่มีค่าความเป็นศูนย์กลางสูง ย่อมถือว่าเป็นศูนย์รวมกิจกรรมที่มีระดับของกิจกรรมในเครือข่ายมากตามไปด้วย [1] ศูนย์รวมกิจกรรมซึ่งมีความเชื่อมโยงกับศูนย์รวมกิจกรรมอื่น ๆ ในเครือข่ายเป็นจำนวนมาก อาจอยู่ในตำแหน่งที่เอื้ออำนวยประโยชน์ให้แก่ศูนย์รวมกิจกรรมต่าง ๆ ได้ ในขณะเดียวกันเป็นไปได้ที่อาจมีการพึ่งพาศูนย์รวมกิจกรรมอื่น ๆ ได้น้อย เนื่องจากเกิดการเข้าถึงโหนดภายในเครือข่ายได้ดีกว่า [2] สมการที่ 1 แสดงสูตรคำนวณ Degree Centrality หรือ ***d(i)*** ของศูนย์รวมกิจกรรม *i* โดยที่ ***mij*** = 1 หากมีการเชื่อมโยงระหว่างศูนย์รวมกิจกรรม ***i*** และ ***j*** หรือ ***mij*** = *0* ถ้าไม่มีการเชื่อมต่อระหว่างกัน

$$d(i) = \sum_{j} m_{ij} \quad (1)$$

การวัดค่าระดับศูนย์กลางจะค้นหาโหนดที่มีจำนวนลิงก์สูงสุดไปยังโหนดอื่น ๆ ในเครือข่าย องศามักเป็นตัววัดอิทธิพลหรือความสำคัญของโหนดที่มีประสิทธิภาพสูง ผู้คนที่มีคอนเนคชันมากกว่ามักจะมีอำนาจและมองเห็นได้ชัดเจนขึ้นในสภาพแวดล้อมทางสังคมหลาย ๆ แห่ง จากภาพที่ 1 โหนดกลางที่มีการเชื่อมต่อมากที่สุดคือ *Mark Taylor* – ซึ่งบอกได้ว่าโหนดนี้มีอิทธิพล



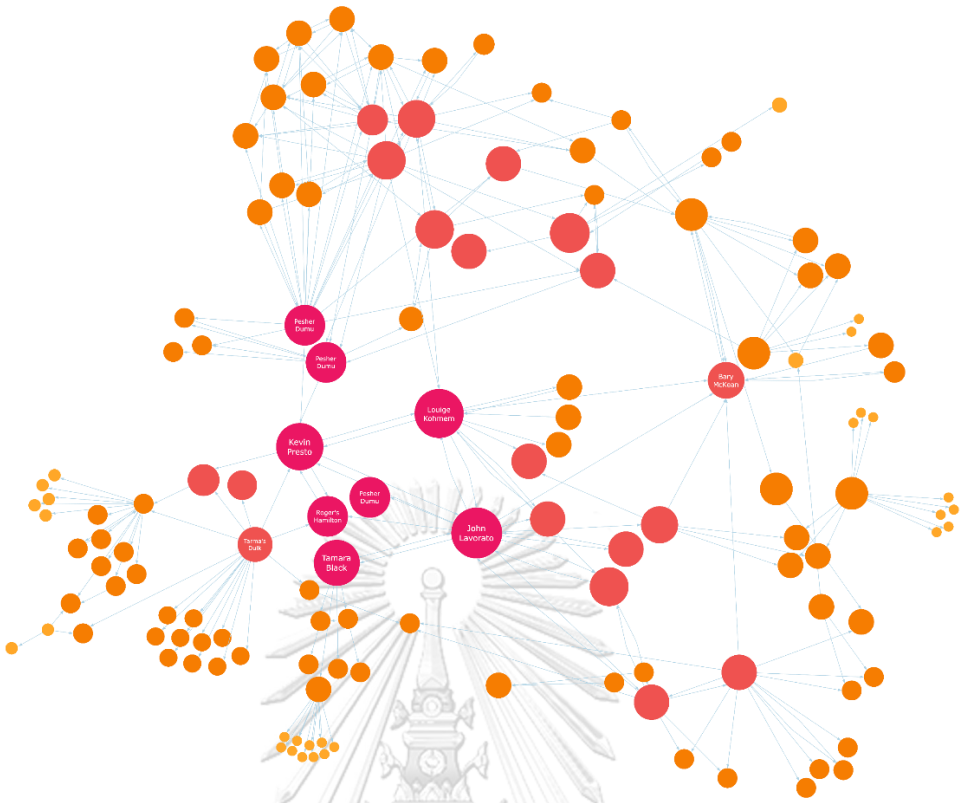
ภาพที่ 1 กราฟเครือข่ายเพื่อเปิดเผยโหนดที่มีค่า Degree Centrality สูง [3]

2.1.2.2 *Closeness Centrality*

คือ การพิจารณาข้อมูลของ Node ที่อยู่จุดศูนย์กลางว่ามีความใกล้ชิดกับ Node ต่าง ๆ ที่อยู่ภายในกลุ่มเดียวกัน ถ้า Node นั้นเข้าเงื่อนไขดังกล่าว แสดงว่า Node นั้นสามารถส่งข้อมูลและสื่อสารกับ Node อื่น ๆ ได้ดี Closeness Centrality เป็นค่าตัววัดความไวของศูนย์รวมกิจกรรมหนึ่ง ๆ ที่จะสามารถเชื่อมต่อไปยังศูนย์รวมกิจกรรมอื่น ๆ ในเครือข่ายได้อย่างไรบ้าง ซึ่งไม่เหมือนกับการวัดระดับความเป็นศูนย์กลางของ Degree Centrality เพราะคำนึงถึงความเชื่อมโยงทั้งทางตรงและทางอ้อมระหว่างศูนย์รวมของกิจกรรมที่เกิดขึ้นผ่านศูนย์รวมของกิจกรรมอื่น ๆ ในเครือข่าย ศูนย์รวมกิจกรรมที่มีค่าคะแนนของ Closeness สูง จะให้ความหมายว่าโหนดนั้นมีความสามารถที่จะติดต่อสื่อสารกับโหนดศูนย์รวมของกิจกรรมอื่น ๆ ได้อย่างรวดเร็ว และถ้าหาก Closeness มีค่าคะแนนสูงยังสื่อถึงประสิทธิภาพในการสื่อสารข้อมูล ข่าวสาร หรือแม้แต่ส่วนของข้อความหรือข้อมูลที่เห็นได้อย่างทั่วถึงทั้งเครือข่าย ซึ่งมีความจำเป็นไม่มากนักที่จะพึ่งพาศูนย์รวมของกิจกรรมอื่น ๆ ในการส่งผ่านข้อมูลข่าวสารอีกด้วย โดยสมการที่ 2 แสดงถึงสูตรการคำนวณค่า Closeness Centrality หรือ ***c(i)*** ของศูนย์รวมกิจกรรม ***i*** โดย ***dij*** คือ จำนวนการเชื่อมต่อในเส้นทางที่มีระยะสั้นที่สุดจากศูนย์รวมกิจกรรม ***i*** ไปสู่ศูนย์รวมกิจกรรม *j*

$$c(i) = \sum_{j} d_{ij} \quad (2)$$

เมื่อพิจารณาจากกราฟเครือข่าย แต่ละโหนดจะได้รับคะแนนที่คำนวณจากผลรวมของเส้นทางที่สั้นที่สุดไปยังโหนดอื่นในเครือข่าย โหนดที่มีค่าความใกล้ชิดสูงจะมีระยะห่างต่ำกว่าโหนดอื่น ๆ ทั้งหมด ตามความสามารถในการเชื่อมต่อกับศูนย์รวมกิจกรรมอื่นได้อย่างรวดเร็ว โหนดที่มีความใกล้ชิดสูง (โหนดขนาดใหญ่กว่าดังแสดงในภาพที่ 2) จะเป็นเครื่องถ่ายทอดข้อมูลที่มีประสิทธิภาพ



ภาพที่ 2 กราฟเครือข่ายเพื่อเปิดเผยโหนดที่มีค่าความใกล้ชิด Closeness Centrality สูง [4]

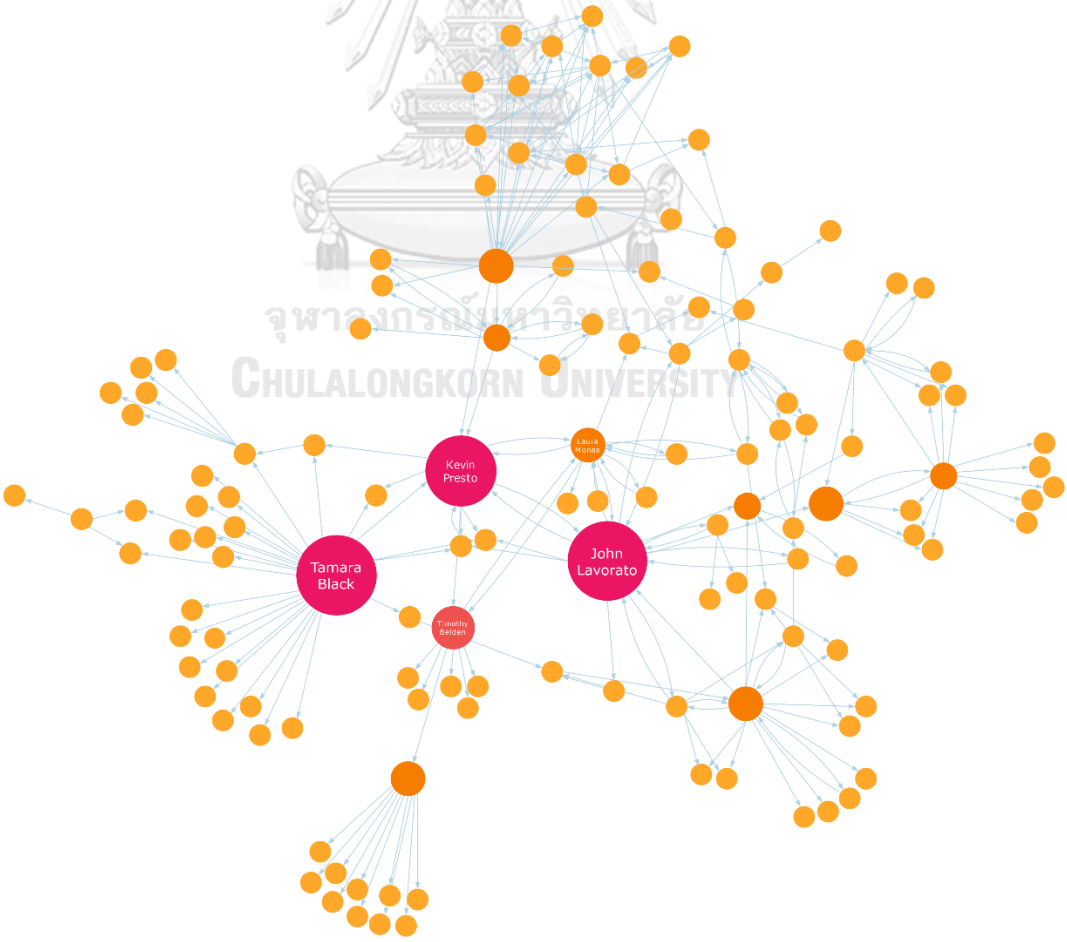
2.1.2.3 *Betweenness Centrality*

คือ การพิจารณาข้อมูลของ Node ที่เชื่อมโยงกับ Node ต่างๆ แบบ Shortest Paths ถ้ามีการเชื่อมโยงมาก แสดงว่า Node นั้นมีความสำคัญต่อระบบเครือข่าย Betweenness Centrality คือ การหา Node ที่อยู่ใจกลางของการสื่อสารบนสมมติฐานที่ว่า ศูนย์รวมกิจกรรมใดก็ตามที่อยู่ระหว่างกลางการเชื่อมโยงของศูนย์รวมกิจกรรมอื่น ๆ ซึ่งเป็นเสมือนศูนย์รวมของกิจกรรมที่เป็นศูนย์กลางของเครือข่าย เพราะศูนย์รวมกิจกรรมในตำแหน่งนี้สามารถควบคุมการมีปฏิสัมพันธ์ของศูนย์รวมกิจกรรมต่าง ๆ ที่เชื่อมโยงผ่านตัวเองได้ โดยสวมบทบาทเป็นเสมือนตัวควบคุมประตูแห่งความสัมพันธ์ หรือทำการทำหน้าที่ขวางกั้นการติดต่อจากสิ่งที่ไม่พึงประสงค์ [1] ซึ่งทำการวัดความเชื่อมโยงทางอ้อมระหว่างศูนย์รวมกิจกรรมต่างๆ ในเครือข่าย การที่มีค่า Betweenness Centrality สูง อาจหมายถึง "ภาวะซ่อนเร้นจากผิวนอกของเครือข่าย" [5] ศูนย์รวมกิจกรรมนี้ทำหน้าที่เสมือนตัวกลางเพื่อการจัดสรรอำนาจ ด้วยเหตุที่อยู่บนแนวทางซึ่งเปิดโอกาสแก่ศูนย์รวมของกิจกรรมอื่น ๆ แม้ว่าจะไม่มีการเชื่อมต่อโดยตรงก็ตาม [5] สมการที่ *3* แสดงสูตรการคำนวณค่า Betweenness Centrality หรือ ***b(i)*** ของศูนย์รวมกิจกรรม ***i*** โดยกำหนดให้ ***gjk*** คือ

จำนวนเส้นทางที่สั้นที่สุดจากศูนย์รวมกิจกรรม ***j*** ไปยังศูนย์รวมกิจกรรม ***k (j, k ≠i)*** และ ***gjik*** คือ จำนวนเส้นทางที่สั้นที่สุดจากศูนย์รวมกิจกรรม ***j*** ไปยังศูนย์รวมกิจกรรม ***k*** ที่ต้องผ่าน ***i***

$$b(i) = \sum_{j,k} \frac{g_{jik}}{g_{jk}} \quad (3)$$

คะแนนของค่า Betweenness Centrality แสดงถึงเส้นทางการสื่อสารที่สั้นที่สุดในเครือข่าย โหนดที่มีคะแนนความเป็นศูนย์กลางระหว่างความเป็นศูนย์กลางสูงถือเป็นศูนย์กลางสำหรับกิจกรรมเครือข่าย โดยสวมบทบาทเป็นผู้รักษาประตูที่สามารถปิดกั้นการติดต่อที่ไม่ต้องการ [1] ในทางกลับกัน อาจหมายถึง "สถานะที่ซ่อนอยู่จากพื้นผิวด้านนอกของเครือข่าย" ที่สามารถให้โอกาสสำหรับศูนย์กิจกรรมอื่นๆ แม้จะไม่มีการเชื่อมต่อโดยตรง [5] ตัวอย่างกราฟเครือข่ายเพื่อค้นหานักแสดงที่มีค่าศูนย์กลางระหว่างความเป็นศูนย์กลางสูงแสดงในภาพที่ 3



ภาพที่ 3 กราฟเครือข่ายเพื่อค้นหาโหนดที่มีค่า Betweenness Centrality สูง [6]

2.1.3 Gephi 0.9.2

Gephi [7] เป็นซอฟต์แวร์โอเพนซอร์ซ (Open Source) สำหรับการวิเคราะห์กราฟและเครือข่าย โดยใช้เครื่องประมวลผลการแสดงภาพ 3 มิติ เพื่อแสดงเครือข่ายขนาดใหญ่แบบเรียลไทม์และเร่งการสำรวจ เป็นสถาปัตยกรรมที่ยืดหยุ่นและทำงานได้หลายอย่างทำให้เกิดความเป็นไปได้ใหม่ๆ เพื่อทำงานกับชุดข้อมูลที่ซับซ้อนและผลิตผลลัพธ์ภาพที่มีคุณค่า และนำเสนอคุณสมบัติที่สำคัญหลายประการของ Gephi ในบริบทของการสำรวจเชิงโต้ตอบและการตีความของเครือข่าย ให้ง่ายและกว้างขึ้น สามารถเข้าถึงข้อมูลเครือข่ายและอนุญาตให้มีการกำหนดพื้นที่ กรองการนำทาง การจัดการ และการจัดกลุ่ม ซึ่งท้ายที่สุดคือการนำเสนอคุณลักษณะแบบไดนามิกของ Gephi โดยเน้นลักษณะสำคัญของการสร้างภาพเครือข่ายแบบไดนามิก การนำ Gephi มาใช้งาน ทำให้สามารถวิเคราะห์ความเชื่อมโยงของข้อมูลในหลายลักษณะตามที่ต้องการได้ โดย Gephi มีความยืดหยุ่นตรงที่สามารถสนับสนุน Plugin และสามารถแสดงผลในรูปแบบกราฟ หรือ Layout ออกมาได้แตกต่างกัน ทำให้เข้าใจข้อมูลนั้น ๆ ได้มากยิ่งขึ้น เช่น จากเดิมข้อมูลการเชื่อมต่อของเครือข่าย LAN หรือ WLAN เป็นแบบข้อความ ทำให้ไม่สามารถมองเห็นภาพการเชื่อมต่อได้ชัดเจน แต่หากสังเกตข้อมูลการเชื่อมต่อเครือข่ายนั้น จะพบว่า เป็นการเชื่อมต่อแบบกราฟ ถ้าทำการแปลงการแสดงผลจากข้อมูลดิบซึ่งเป็นข้อความ (Text) มาแสดงผลแบบกราฟิก หรือ การสร้างภาพนามธรรม (Data Visualization) แล้ว จะทำให้สามารถมองเห็นภาพรวมของระบบได้ชัดเจนมากยิ่งขึ้น และยังสามารถนำข้อมูลการเชื่อมต่อของ Social Network มาทำการวิเคราะห์ได้ เช่น เฟซบุ๊ก ไลน์ (Line) หรือ ทวิตเตอร์ เป็นต้น

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1. Using Social Network Analysis for Mining Collaboration Data in a Defect Tracking System for Risk and Vulnerability

Ashish Sureka และคณะ [8] ได้ทำการศึกษาและเสนอการประยุกต์ใช้การวิเคราะห์เครือข่ายสังคมกับการโต้ตอบของนักพัฒนาข้อมูลโดยนัยในระบบติดตามข้อบกพร่อง และด้วยเหตุนี้เพื่อจำกัดพื้นที่ที่ผู้วิจัยตรวจสอบเฉพาะงานที่เกี่ยวข้องอย่างใกล้ชิดกับบทความนี้ (เช่น งานเกี่ยวกับแอปพลิเคชันโซเชียลเน็ตเวิร์ก การวิเคราะห์ที่เก็บซอฟต์แวร์) มีการจัดประเภทงานที่เกี่ยวข้องในมิติของพื้นที่ในการเก็บข้อมูลซอฟต์แวร์ (ที่เก็บรหัสต้นทาง (source code) ระบบติดตามข้อบกพร่อง และเวอร์ชันของเอกสารสำคัญ ฯลฯ) โดยงานวิจัยนี้ได้นำเอาข้อมูลเกี่ยวกับระดับความรุนแรงของข้อบกพร่อง

และส่วนประกอบ (Component) ของโปรแกรม มาวิเคราะห์หาความเชื่อมโยงและหาส่วนสำคัญที่สุดในเครือข่าย โดยใช้การวิเคราะห์จากกราฟเครือข่ายสังคม

2.2.2. Social Network Analysis in Software Testing to Categorize JUnit Test Cases based on Coverage Information

N. Koochakzadeh และ R. Alhajj [9] ได้นำเสนอเทคนิคในการจัดหมวดหมู่กรณีทดสอบโดยอัตโนมัติตามความครอบคลุมข้อมูล สามารถดำเนินการตามกระบวนการที่เสนอได้แบบไดนามิกตลอดวงจรชีวิตของระบบ เพื่อปรับปรุงคุณภาพของชุดการทดสอบ โดยสร้างเครือข่ายโซเชียลของกรณีทดสอบ และใช้ข้อมูลความครอบคลุมเพื่อกำหนดความเชื่อมโยงระหว่างกัน ซึ่งเครือข่ายนี้จะใช้เพื่อระบุกลุ่มกรณีทดสอบที่สูงขึ้น เช่น ชุดการทดสอบ โดยจากความรู้ที่ดีที่สุดที่กลุ่มผู้วิจัยมี ผู้วิจัยได้กล่าวว่า งานวิจัยนี้เป็นการทดลองครั้งแรกของเรื่องที่ทำการศึกษาในทิศทางนี้ เพื่อประเมินเทคนิคของกลุ่มผู้วิจัย ผู้วิจัยจึงได้ประยุกต์ใช้เทคนิคนี้กับระบบโอเพนซอร์ซจำนวน 3 ระบบ พร้อมกับชุดทดสอบ JUnit ที่พร้อมใช้งานเพื่อระบุแพ็กเกจการทดสอบ โดยผู้วิจัยได้วัดคุณภาพของแพ็กเกจที่ถูกค้นพบในเงื่อนไขของการเชื่อมแน่น (cohesion) และการคู่ควบ (coupling) และเปรียบเทียบสิ่งเหล่านั้นกับชุดการทดสอบเดิมจากผู้ทดสอบของโครงการนี้ ซึ่งผลลัพธ์ที่ได้ คือ เทคนิคของผู้วิจัยที่ได้เสนอมานั้นสามารถจัดหมวดหมู่แบบกรณีทดสอบได้โดยอัตโนมัติ โดยการปรับปรุงคุณภาพของชุดการทดสอบอย่างสม่ำเสมอ โดยจะไปแก้ปัญหาของการทดสอบซอฟต์แวร์ซึ่งเป็นสิ่งที่มองเห็นได้และสิ้นเปลืองที่สุด ซึ่งเป็นกิจกรรมในการรับประกันคุณภาพของระบบซอฟต์แวร์ในทุกวันนี้

2.2.3. GUI Test Case Prioritization using Social Network Analysis

Maitrikul and Limpiyakorn [10] นำเสนอการประยุกต์ใช้ค่าความเป็นศูนย์กลางในระบบเครือข่ายจากการวิเคราะห์เครือข่ายทางสังคมเพื่อจัดอันดับความสำคัญของกรณีทดสอบ และค้นหาพารามิเตอร์ที่เหมาะสมสำหรับการจัดลำดับกรณีทดสอบจียูไอ โดยใช้การวิเคราะห์เครือข่ายสังคมออนไลน์ การวัดความเป็นศูนย์กลางของเครือข่าย ซึ่งรวมถึงการรวมระหว่างความเป็นศูนย์กลาง ความใกล้ชิด ความใกล้ชิด ศูนย์กลางเวกเตอร์ลักษณะเฉพาะ และอันดับเพจ ถูกเลือกสำหรับการจัดอันดับทั้งกรณีทดสอบที่แก้ไขและกรณีทดสอบใหม่ในระหว่างการทดสอบการถดถอยในระบบผู้แนะนำขนาดใหญ่ ผลการวิจัยรายงานว่าความเป็นศูนย์กลางระหว่างกันบรรลุประสิทธิภาพที่ดีที่สุดในการค้นหาข้อผิดพลาดเร็วที่สุดในรอบของการทดสอบ แนวทางนี้จะเป็นประโยชน์ต่อการลดทรัพยากรที่ใช้ในระหว่างกระบวนการทดสอบและเพิ่มอัตราการตรวจจับข้อผิดพลาดในระยะแรกของการทดสอบ แนวทางดังกล่าวจะมีผลทำให้ทรัพยากรที่ใช้ระหว่างการประมวลผลทดสอบลดลง และเพิ่ม

อัตราการตรวจจับข้อบกพร่องได้เร็วขึ้นตั้งแต่ในระยะแรก ๆ ของการทดสอบ และยังมีการใช้โปรแกรมวิเคราะห์ SNA ด้วยโปรแกรม Gephi ซึ่งเล็งเห็นถึงแนวโน้มและวิธีการที่สามารถนำมาประยุกต์ใช้กับงานวิจัยที่จะทำต่อไปได้

2.2.4. Defect Escape Analysis: Test Process Improvement

Mary Ann Vandermark [11] ได้ศึกษาเกี่ยวกับการวิเคราะห์การหลบหนีข้อบกพร่อง และได้อธิบายถึงคำจำกัดความของการหลบหนี (Escape) โดยผู้วิจัยได้กำหนดให้ Escape คือ ข้อบกพร่องที่ไม่ถูกพบโดยทีมทดสอบ แต่ข้อบกพร่องนั้นมีการถูกพบโดยลูกค้าแทน เมื่อปัญหาถูกเปิดเผยโดยลูกค้าอาจทำให้การแก้ไขข้อบกพร่องที่เกิดขึ้นในระยะนั้นมีราคาที่ค่อนข้างสูงเมื่อมองย้อนกลับไปที่กระบวนการพัฒนาซอฟต์แวร์ที่ข้อบกพร่องไม่ได้ถูกพบ แต่หากข้อบกพร่องเหล่านั้นถูกพบในระยะของกระบวนการพัฒนาตั้งแต่เริ่มต้น จะทำให้งบของการพัฒนามีราคาที่ถูกกว่าที่ควรจะเป็น และเงินจำนวนน้อยมากที่จะถูกใช้ในภาพรวมเมื่อปัญหาถูกตรวจจับได้โดยทีมทดสอบแทนที่จะเป็นลูกค้า นอกจากนี้ ยังมีราคาถูกกว่ามากเมื่อเทียบกับการที่ทีมพัฒนาตรวจพบปัญหาตั้งแต่เริ่มต้นแทนที่จะเป็นทีมทดสอบค้นพบ เพราะสิ่งนี้จะเป็นประโยชน์ในการวิเคราะห์การหลบหนีของข้อบกพร่องเพื่อที่จะสามารถบอกได้ว่า ทำไมข้อบกพร่องดังกล่าวถึงหลบหนี เพื่อป้องกันการหลบหนีในอนาคต และขับเคลื่อนการค้นพบข้อบกพร่องให้กลับเข้าสู่กระบวนการพัฒนาซอฟต์แวร์ให้มากที่สุด วิธีนี้จะช่วยลดต้นทุนและปรับปรุงคุณภาพของผลิตภัณฑ์ซอฟต์แวร์ภายในโครงการได้

วัตถุประสงค์การวิเคราะห์การหลบหนีของข้อบกพร่อง คือ เพื่อที่จะแน่ใจได้ว่า มีการปรับปรุงบนผลิตภัณฑ์ซอฟต์แวร์และในกระบวนการทดสอบและพัฒนาอย่างต่อเนื่อง สามารถกระทำได้โดยผ่านการวิเคราะห์ การหลบหนีของข้อบกพร่องในการพัฒนา การทดสอบ และวางแผนป้องกันเพื่อหลีกเลี่ยงการหลบหนีที่คล้ายกันในอนาคตได้ การปรับปรุงกระบวนการทดสอบอย่างต่อเนื่องจะเพิ่มประสิทธิภาพของสภาพแวดล้อมและวิธีการทดสอบ ลดจำนวนข้อบกพร่องที่ลูกค้าพบและค่าใช้จ่าย รวมถึงปรับปรุงคุณภาพของผลิตภัณฑ์ ชื่อเสียง และการขาย ยิ่งหากมีทีมใดไปมีบทบาทในการพัฒนาซอฟต์แวร์โดยรวมของโครงการที่ไปเกี่ยวข้องกับการวิเคราะห์การหลบหนีมากเท่าไหร่ กระบวนการวิเคราะห์การหลบหนีก็จะมีประสิทธิภาพมากขึ้นเท่านั้น และจะมีประสิทธิภาพมากที่สุด หากการพัฒนาเช่นเดียวกับการทดสอบมีส่วนเกี่ยวข้องกันอย่างใกล้ชิด โดยกระบวนการนี้จะมุ่งเน้นไปที่ระดับความรุนแรงสูง ปัญหาที่มีความแพร่หลายและค่าใช้จ่ายสูง ทั้งนี้ วัตถุประสงค์ของการวิเคราะห์การหลบหนี คือ:

1. แยกการหลบหนีออกเป็นหมวดหมู่ที่เป็นประโยชน์ เพื่อการวิเคราะห์เชิงลึกมากขึ้น
2. ประมวลผลสถิติบนข้อมูลที่ถูกจัดตามประเภท
3. ระบุและเปลี่ยนแปลงกระบวนการโดยรวมที่จำเป็นตามข้อมูลทางสถิติ
4. ระบุและทำการเปลี่ยนแปลงในระดับต่ำ (ระดับแผนก) ที่จำเป็นจากการวิเคราะห์เชิงลึกของการหลบหนีที่เฉพาะเจาะจง
5. ใช้การวัดเพื่อแสดงให้เห็นถึงประสิทธิภาพของการเปลี่ยนแปลงกระบวนการ

2.2.5. Predicting Defects using Network Analysis on Dependency Graphs

T. Zimmermann และ N. Nagappan [12] ได้เสนอการใช้การวิเคราะห์เครือข่ายบนกราฟแบบพึ่งพา ซึ่งทำให้ผู้จัดการโครงการสามารถระบุได้ว่า หน่วยโปรแกรมกลางที่มีแนวโน้มที่จะประสบกับข้อบกพร่องในงานวิจัย การประเมินบน Windows Server 2003 ผู้วิจัยพบว่า การเรียกคืนสำหรับโมเดลที่สร้างขึ้นจากการวัดเครือข่ายนั้นสูงกว่า 10% สำหรับโมเดลที่สร้างขึ้นจากตัวชี้วัดความซับซ้อน นอกจากนี้ การวัดเครือข่ายสามารถระบุ 60% ของไบนารีที่ Windows นักพัฒนาถือว่ามีความสำคัญมากเป็นสองเท่าของที่ระบุโดยตัวชี้วัดความซับซ้อน โดยกลุ่มผู้วิจัยได้แสดงให้เห็นว่าเครือข่ายวัดตามกราฟแบบพึ่งพาซึ่งสามารถทำนายข้อบกพร่องสำหรับไบนารีของ Windows Server 2003 สิ่งนี้สนับสนุนผู้จัดการในการจัดสรรทรัพยากร เช่น เวลาและต้นทุน สำหรับการประกันคุณภาพ โดยตามหลักการแล้วชิ้นส่วนที่มีข้อบกพร่องส่วนใหญ่จะเป็นส่วนที่ทดสอบมากที่สุด

ผู้วิจัยได้นำเสนอผลการศึกษาเชิงประจักษ์ของงาน ดังต่อไปนี้

- ตัวชี้วัดที่มีความซับซ้อนล้มเหลวในการทำนายไบนารีที่นักพัฒนาพิจารณาแล้วว่าวิกฤต (คาดการณ์เพียง 30%)
- การวัดเครือข่ายสามารถทำนายได้ถึง 60% ของไบนารีที่สำคัญเหล่านี้
- การวัดเครือข่ายบนกราฟการพึ่งพาสามารถระบุและทำนายจำนวนข้อบกพร่องได้
- เมื่อใช้สำหรับการจัดประเภท การวัดเครือข่ายมีการเรียกคืน ซึ่งสูงกว่าตัววัดความซับซ้อน 0.10 ที่มีความแม่นยำเทียบเท่า

2.2.6. Defect Prediction Using Social Network Analysis on Issue Repositories

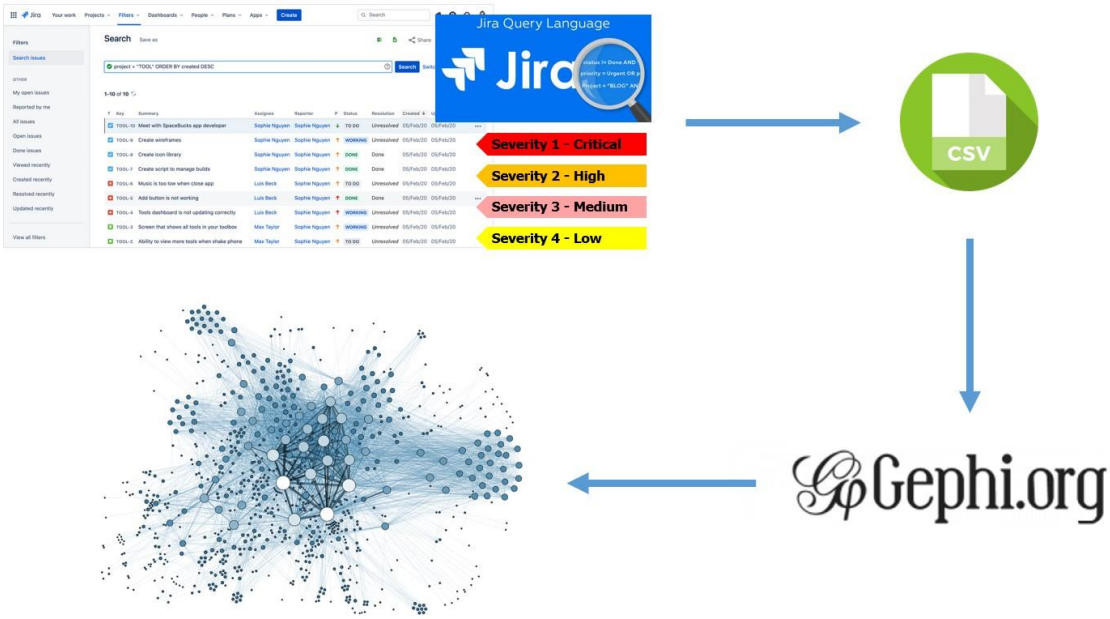
Serdar Biçer และคณะ [13] ได้เสนอผลงานหลักของการศึกษานี้ คือ การใช้แหล่งข้อมูลใหม่และตัววัดในพื้นที่ของการทำนายข้อบกพร่อง จนกระทั่งตอนนี้ วิธีอื่นๆ ก็ยังคงใช้ผลิตภัณฑ์ และ/หรือกระบวนการที่เกี่ยวข้องกับตัววัด ทุกคนทราบดีว่าการสื่อสารและการประสานงานระหว่างนักพัฒนามีความสำคัญ แต่รูปแบบของปฏิสัมพันธ์ระหว่างนักพัฒนาซอฟต์แวร์ยังไม่ได้รับการตรวจสอบการทำนายข้อบกพร่อง ด้วยเหตุนี้ กลุ่มผู้วิจัยจึงได้เสนอตัววัดชุดใหม่เกี่ยวกับคลังเก็บปัญหา (issue

repositories) เพื่อปรับปรุงสถานะปัจจุบันของตัวทำนายข้อบกพร่อง โดยการเพิ่มเนื้อหาข้อมูลของข้อมูลการฝึกอบรม เพื่อจุดประสงค์นี้ กลุ่มผู้วิจัยจึงได้ทำการทดลองเกี่ยวกับการสื่อสาร โดยใช้ข้อมูลของโครงการ RTC และ Drupal และยังเปรียบเทียบการค้นพบของผู้วิจัยด้วยการวัดแบบปั่นป่วนเพื่อระบุประสิทธิภาพเพิ่มเติมของแบบจำลองที่ได้นำเสนอ

ผู้วิจัยสังเกตว่า การใช้ตัวชี้วัดเครือข่ายสังคมในประเด็นที่เก็บลดต้นทุนที่จำเป็นสำหรับการตรวจสอบการคาดการณ์ผลลัพธ์และทำให้ผลลัพธ์ใกล้เคียงกับภาคต้นทุนที่ไม่พึงประสงค์ของเส้นโค้ง ROC จากจุดนี้ผู้วิจัยพบว่าตัวชี้วัดที่นำเสนอสามารถเพิ่มประสิทธิภาพการทำนายอย่างมากตามที่ต้องการ ความพยายามในการทดสอบน้อยลง จึงควรค่าแก่การสะสม ผลลัพธ์ดังกล่าวอาจทำให้ผู้จัดการโครงการซอฟต์แวร์มุ่งเน้นและปรับปรุงนักพัฒนาประสานงานและลดต้นทุนการทดสอบดังกล่าว เป็นการศึกษาครั้งแรก การประยุกต์ใช้การวิเคราะห์เครือข่ายเพื่อการสื่อสารของนักพัฒนาโครงสร้างในการทำนายข้อบกพร่อง ซึ่งงานในอนาคตอาจรวมถึงตัวชี้วัดเครือข่ายทางสังคมอื่น ๆ โดยการวิเคราะห์เพิ่มเติมไปยังเครือข่ายของนักพัฒนา

บทที่ 3
แนวคิดและวิธีการวิจัย

Jira Software เป็นโซลูชันการจัดการโครงการแบบอไจล์ ซึ่งเดิมออกแบบมาให้เป็นเครื่องมือในการติดตามข้อบกพร่องและปัญหาที่เกิดขึ้นภายในโครงการ การค้นหาปัญหาหรือข้อมูลข้อบกพร่องสามารถทำได้โดยใช้ Jira Query Language (JQL) อย่างไรก็ตาม การสืบค้นจากแหล่งที่เก็บข้อมูลจะส่งคืนข้อมูลที่เฉพาะเจาะจงและข้อมูลทั่วไปเป็นแบบง่ายและธรรมดา ในงานวิจัยนี้ ได้นำเสนอแนวทางการสร้างภาพข้อมูลเครือข่ายเพื่อเปิดเผยความสัมพันธ์และการสื่อสารระหว่างตัวแสดง เช่น ฟีเจอร์ ข้อบกพร่อง และบุคลากร โดยเทคนิคการวิเคราะห์เครือข่ายโซเชียลใช้สำหรับวิเคราะห์ข้อบกพร่องที่รวบรวมจากโครงการซอฟต์แวร์ภายในธนาคาร และ Gephi ถูกใช้เป็นเครื่องมือในการสร้างเครือข่ายของตัวแสดงที่ระบุเป็นโหนดและการเชื่อมโยงของพวกเขา ซึ่งแนวทางของการวิเคราะห์เครือข่ายภาพนั้นใช้ได้จริงและให้ข้อมูลเชิงลึกในการวิเคราะห์ข้อบกพร่องที่จำเป็นสำหรับกระบวนการพัฒนาซอฟต์แวร์ในเชิงรุก ลำดับขั้นตอนการดำเนินงานวิจัย แบ่งออกเป็น 4 ขั้นตอนหลัก ดังแสดงในภาพที่ 4 ประกอบด้วย: 1) การเก็บรวบรวมข้อมูลข้อบกพร่องจาก JIRA Software 2) การเตรียมชุดข้อมูล 3) การสำรวจและวิเคราะห์ข้อมูลเกี่ยวกับสาเหตุและระดับความรุนแรงของข้อบกพร่อง และ 4) การสร้างแบบจำลอง SNA เพื่อช่วยในการวิเคราะห์ข้อมูล

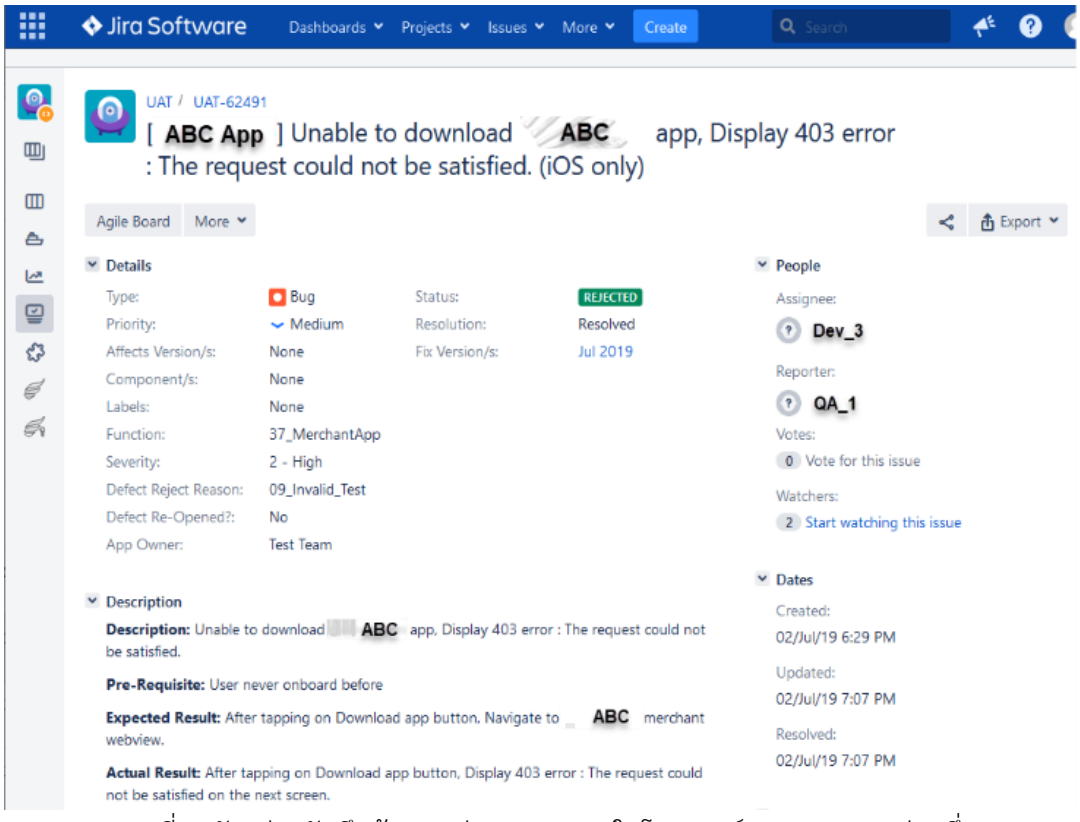


ภาพที่ 4 ขั้นตอนวิธีวิจัยการสร้างแบบจำลอง SNA เพื่อการวิเคราะห์ข้อมูลข้อบกพร่อง

3.1 การเก็บรวบรวมข้อมูลข้อบกพร่อง

เริ่มต้นจากการรวบรวมข้อมูลข้อบกพร่องจาก JIRA Software ซึ่งเป็น Tracking Tool ที่ใช้ใน

การบันทึกข้อมูลข้อบกพร่องในโครงการของธนาคารแห่งหนึ่ง ซึ่งเป็นเครื่องมือติดตามปัญหาที่เป็นกรรมสิทธิ์ของโครงการ ในงานวิจัยนี้ มีการเลือกข้อมูลข้อบกพร่องของโครงการซอฟต์แวร์ในองค์กรธนาคาร โพรเจกต์ประกอบด้วยสี่โมดูล ซึ่งแต่ละโมดูลมีคุณสมบัติหลายอย่าง จำนวนฟีเจอร์ทั้งหมดที่นำมาใช้เริ่มตั้งแต่ ม.ค. - เม.ย. 2565 คือ 170 ฟีเจอร์ ภาพที่ 5 แสดงตัวอย่างภาพหน้าจอสำหรับการป้อนข้อมูลข้อบกพร่องลงใน JIRA โดยข้อมูลดังกล่าวสามารถ Export ออกมาเป็นไฟล์ CSV เพื่อนำมาใช้ในการประมวลผลต่อไปได้ ข้อมูลข้อบกพร่องที่ใช้เป็นข้อมูลนำเข้าจะถูกจับคู่และจัดเก็บไว้ในไฟล์สเปรดชีตดังแสดงในภาพที่ 6



ภาพที่ 5 ตัวอย่างบันทึกข้อบกพร่องจาก JIRA ในโพรเจกต์ของธนาคารแห่งหนึ่ง

[UAT-74898] [My Car Insurance] Display Error: Unable to Proceed, After Tap "proceed via BEASY" (Android Only) Created: 22/Oct/19 Updated: 22/Oct/19

Status:	Investigating
Project:	UAT
Component/s:	FE_005
Affects Version/s:	None
Fix Version/s:	Nov 2019

Id	Label	Release	Defect	Severity	Developer	Tester	Reporter
3	FE-003	21-Jan	NO		DEV_2	QA_2	
4	FE-004	21-Jan	NO		DEV_3	QA_2	
5	FE-005	21-Jan	YES	2	DEV_3	QA_1	QA
6	FE-006	21-Jan	YES	2	DEV_2	QA_2	QA
7	FE-007	21-Jan	YES	1	DEV_5	QA_3	BA

Type:	Bug	Priority:	High
Reporter:	QA_1	Assignee:	Dev_3
Resolution:	Unresolved	Votes:	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

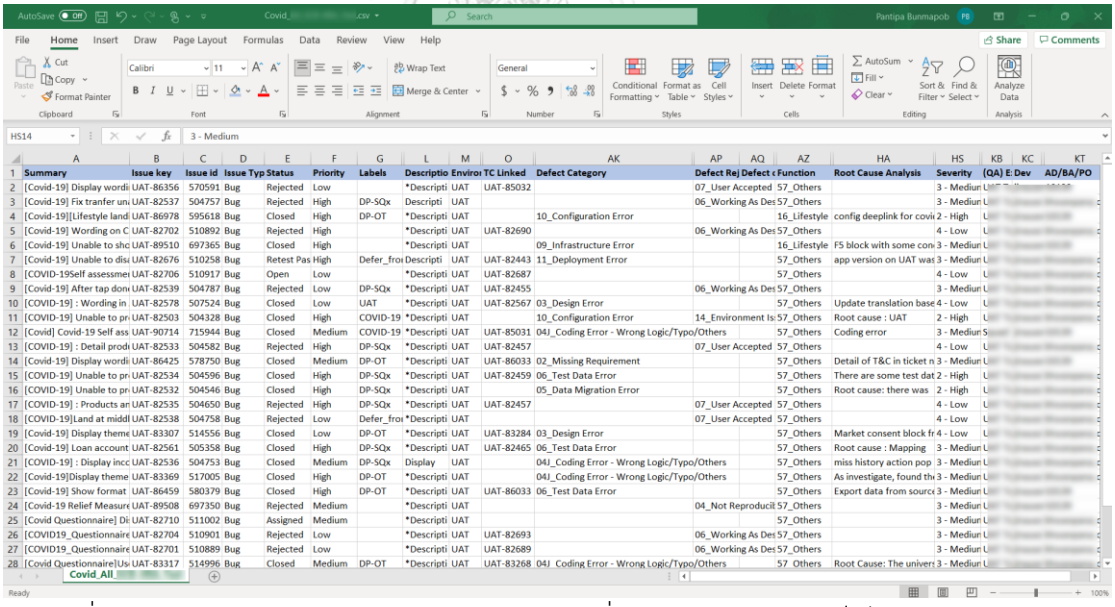
Attachments:	87043.jpg 87044.jpg SVID_20191022_171312_1.mp4
Issue Links:	Relates relates to UAT-74534 TS_BL-4065_MyCar_Insurance_ _009 To Do
Function:	04D_BillPay
Severity:	2 - High
Defect Re-Opened?:	No
App Owner:	Squad 5

Description

Description - [My Car Insurance] Display Error: Unable to Proceed, After Tap "proceed via BEASY" (Android Only)

Expected Result - Able to navigate to B Easy succcessfully.
Actual Result - Display Error: Unable to Proceed, After Tap "proceed via BEASY" (Android Only)

ภาพที่ 6 ตัวอย่างการจับคู่และการจัดเก็บข้อมูลข้อบกพร่องเพื่อป้อนข้อมูลลงในสเปรดชีต



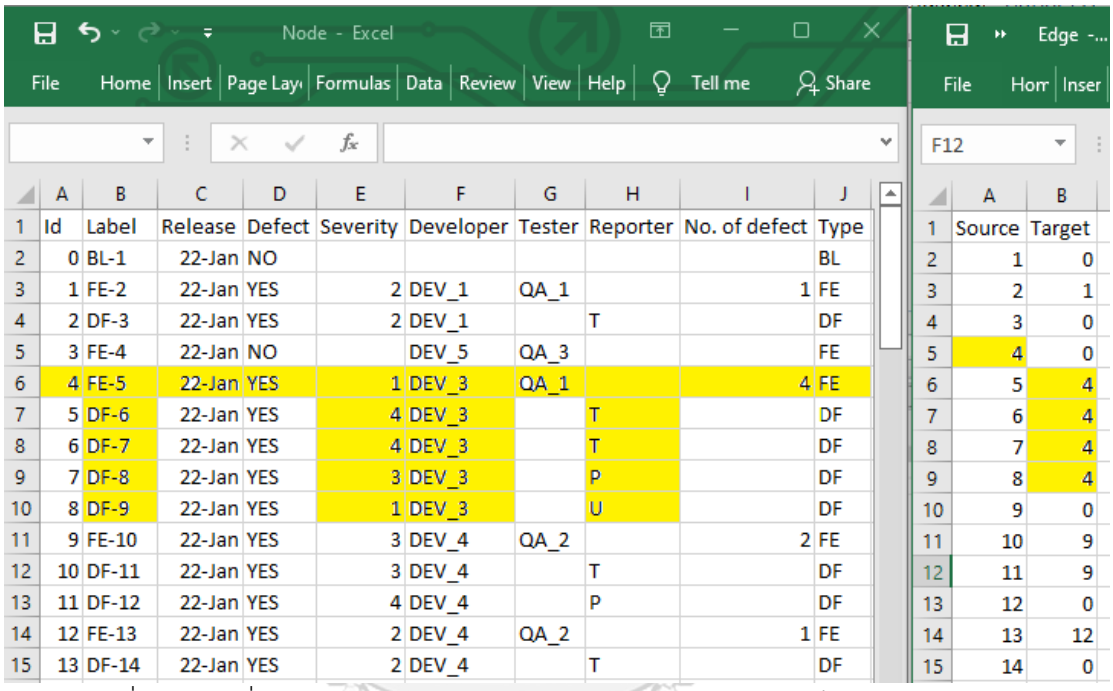
Summary	Issue key	Issue id	Issue Typ	Status	Priority	Labels	Descriptio	Enviror	TC Linked	Defect Category	Defect Rej	Defect (	Function	Root Cause Analysis	Severity	(QA) E	Dev	AD/BA/PO
[Covid-19] Display wordi	UAT-86356	570591	Bug	Rejected	Low		*Descripti	UAT	UAT-85032		07_User Accepted		57_Others		3 - Medium			
[Covid-19] Fix tranfer uni	UAT-82537	504757	Bug	Rejected	High	DP-SQx	Descripti	UAT			06_Working As Des		57_Others		3 - Medium			
[Covid-19][Lifestyle landi	UAT-86978	595618	Bug	Closed	High	DP-OT	*Descripti	UAT		10_Configuration Error			16_Lifestyle	config deeplink for covi	2 - High			
[Covid-19] Wording on C	UAT-82702	510892	Bug	Rejected	High		*Descripti	UAT	UAT-82690		06_Working As Des		57_Others		4 - Low			
[Covid-19] Unable to shc	UAT-89510	697365	Bug	Closed	High		*Descripti	UAT		09_Infrastructure Error			16_Lifestyle	F5 block with some con	3 - Medium			
[Covid-19] Unable to dis	UAT-82676	510258	Bug	Retest Pas	High	Defer_fro	Descripti	UAT	UAT-82443	11_Deployment Error			57_Others	app version on UAT was	3 - Medium			
[COVID-19Self assessmei	UAT-82706	510917	Bug	Open	Low		*Descripti	UAT	UAT-82687				57_Others		4 - Low			
[Covid-19] After tap don	UAT-82539	504787	Bug	Rejected	Low	DP-SQx	*Descripti	UAT	UAT-82455		06_Working As Des		57_Others		3 - Medium			
[COVID-19] : Wording in	UAT-82578	507524	Bug	Closed	Low	UAT	*Descripti	UAT	UAT-82567	03_Design Error			57_Others	Update translation base	4 - Low			
[COVID-19] Unable to pr	UAT-82503	504328	Bug	Closed	High	COVID-19	*Descripti	UAT		10_Configuration Error	14_Environment Is		57_Others	Root cause : UAT	2 - High			
[Covid] Covid-19 Self ass	UAT-90714	715944	Bug	Closed	Medium	COVID-19	*Descripti	UAT	UAT-85031	04J_Coding Error - Wrong Logic/Typo/Others			57_Others	Coding error	3 - Medium			
[COVID-19] : Detail prod	UAT-82533	504582	Bug	Rejected	High	DP-SQx	*Descripti	UAT	UAT-82457		07_User Accepted		57_Others		4 - Low			
[Covid-19] Display wordi	UAT-86425	578750	Bug	Closed	Medium	DP-OT	*Descripti	UAT	UAT-86033	02_Missing Requirement			57_Others	Detail of T&C in ticket n	3 - Medium			
[COVID-19] Unable to pr	UAT-82534	504596	Bug	Closed	High	DP-SQx	*Descripti	UAT	UAT-82459	06_Test Data Error			57_Others	There are some test dat	2 - High			
[COVID-19] Unable to pr	UAT-82532	504546	Bug	Closed	High	DP-SQx	*Descripti	UAT		05_Data Migration Error			57_Others	Root cause: there was	2 - High			
[COVID-19] : Products an	UAT-82535	504650	Bug	Rejected	High	DP-SQx	*Descripti	UAT	UAT-82457		07_User Accepted		57_Others		4 - Low			
[COVID-19]Land at middl	UAT-82538	504758	Bug	Rejected	Low	Defer_fro	*Descripti	UAT			07_User Accepted		57_Others		4 - Low			
[Covid-19] Display theme	UAT-83307	514556	Bug	Closed	Low	DP-OT	*Descripti	UAT	UAT-83284	03_Design Error			57_Others	Market consent block fr	4 - Low			
[Covid-19] Loan account	UAT-82561	505358	Bug	Closed	High	DP-SQx	*Descripti	UAT	UAT-82465	06_Test Data Error			57_Others	Root cause : Mapping	3 - Medium			
[COVID-19] : Display incc	UAT-82536	504753	Bug	Closed	Medium	DP-SQx	Display	UAT		04J_Coding Error - Wrong Logic/Typo/Others			57_Others	miss history action pop	3 - Medium			
[Covid-19]Display theme	UAT-83369	517005	Bug	Closed	High	DP-OT	*Descripti	UAT		04J_Coding Error - Wrong Logic/Typo/Others			57_Others	As investigate, found th	3 - Medium			
[Covid-19] Show format	UAT-86459	580379	Bug	Closed	High	DP-OT	*Descripti	UAT	UAT-86033	06_Test Data Error			57_Others	Export data from source	3 - Medium			
[Covid-19 Relief Measure	UAT-89508	697350	Bug	Rejected	Medium		*Descripti	UAT			04_Not Reproducil		57_Others		3 - Medium			
[Covid Questionnaire] Di	UAT-82710	511002	Bug	Assigned	Medium		*Descripti	UAT					57_Others		3 - Medium			
[COVID19_Questionnaire	UAT-82704	510901	Bug	Rejected	Low		*Descripti	UAT	UAT-82693		06_Working As Des		57_Others		3 - Medium			
[COVID19_Questionnaire	UAT-82701	510889	Bug	Rejected	Low		*Descripti	UAT	UAT-82689		06_Working As Des		57_Others		3 - Medium			
[Covid Questionnaire]Us	UAT-83317	514996	Bug	Closed	Medium	DP-OT	*Descripti	UAT	UAT-83268	04J_Coding Error - Wrong Logic/Typo/Others			57_Others	Root Cause: The univers	3 - Medium			

ภาพที่ 7 ตัวอย่างข้อมูลข้อบกพร่องจากระบบ JIRA ที่ถูก Export ออกมาเป็นไฟล์ CSV ด้วย JQL

เมื่อใช้คำสั่ง JQL ในการ query ข้อมูลข้อบกพร่องจากระบบ JIRA Software ทำให้สามารถ Export ข้อมูลข้อบกพร่องทั้งหมดที่มีภายในโพรเจกต์ที่ถูกเลือกได้ รวมถึงข้อมูลอื่น ๆ ที่จำเป็นต่อการวิเคราะห์ข้อมูลเครือข่ายออกมาในรูปแบบของไฟล์ CSV ดังตัวอย่างที่แสดงในภาพที่ 7

3.2 การเตรียมชุดข้อมูล

ในการเตรียมไฟล์ข้อมูลสำหรับเป็นข้อมูลนำเข้าสำหรับใช้ในการวิเคราะห์สังคมเครือข่าย ถูกแบ่งออกเป็น 2 ไฟล์ในรูปแบบ csv คือ Node และ Edge ซึ่งถูกสร้างขึ้นเองด้วยมือ ไฟล์เหล่านี้จำเป็นต้องใช้เป็นอินพุตของโปรแกรม Gephi เพื่อสร้างกราฟเครือข่าย โดยส่วนของไฟล์ Node.csv และ Edge.csv ที่เป็นอินพุตของโปรแกรม Gephi ซึ่งถูกแสดงในภาพที่ 8



Node - Excel

Id	Label	Release	Defect	Severity	Developer	Tester	Reporter	No. of defect	Type
0	BL-1	22-Jan	NO						BL
1	FE-2	22-Jan	YES	2	DEV_1	QA_1		1	FE
2	DF-3	22-Jan	YES	2	DEV_1		T		DF
3	FE-4	22-Jan	NO		DEV_5	QA_3			FE
4	FE-5	22-Jan	YES	1	DEV_3	QA_1		4	FE
5	DF-6	22-Jan	YES	4	DEV_3		T		DF
6	DF-7	22-Jan	YES	4	DEV_3		T		DF
7	DF-8	22-Jan	YES	3	DEV_3		P		DF
8	DF-9	22-Jan	YES	1	DEV_3		U		DF
9	FE-10	22-Jan	YES	3	DEV_4	QA_2		2	FE
10	DF-11	22-Jan	YES	3	DEV_4		T		DF
11	DF-12	22-Jan	YES	4	DEV_4		P		DF
12	FE-13	22-Jan	YES	2	DEV_4	QA_2		1	FE
13	DF-14	22-Jan	YES	2	DEV_4		T		DF

Edge -...

Source	Target
1	0
2	1
3	0
4	0
5	4
6	4
7	4
8	4
9	0
10	9
11	9
12	0
13	12
14	0

ภาพที่ 8 ข้อมูลที่ถูกตัดตอนมาของไฟล์ Node.csv และ Edge.csv เป็นอินพุตของ Gephi

- **Node file—** เอนทิตีที่เป็นองค์ประกอบและปรากฏอยู่ใน Node.csv ถูกระบุโดย id ที่เริ่มต้นด้วยค่าศูนย์ แต่ละโหนดหรือตัวแสดงอาจเชื่อมโยงกับชุดของคุณลักษณะที่น่าสนใจ เช่น ป้ายกำกับ ชื่อบุคคล รายละเอียดของข้อบกพร่องอื่น ๆ ที่จำเป็นต่อการวิเคราะห์เครือข่าย เช่น ระดับความรุนแรงของข้อบกพร่อง (Defect Severity) นักพัฒนา (Developer) นักทดสอบ (Tester) หรือ ผู้รายงาน (Reporter) เป็นต้น
- **Edge file—** เป็นการให้รายละเอียดของความสัมพันธ์ที่ทราบทั้งหมดของโหนดต้นทางและโหนดเป้าหมายซึ่งถูกแมปกับฟิลด์ข้อมูลของปัญหาที่เชื่อมโยงกันจาก Jira โดยคอลัมน์ Source แทนโหนดที่เป็นต้นทาง ส่วนคอลัมน์ Target แทนโหนดเป้าหมายที่มีความเชื่อมโยงระหว่างโหนดต้นทางและโหนดอื่น ๆ ที่เกี่ยวข้องกัน

ในงานวิจัยนี้ มีรายการข้อมูลนำเข้าจำนวนทั้งหมด 347 รายการ ที่ประกอบไปด้วยฟีเจอร์การทำงานและข้อบกพร่องที่มีระดับความรุนแรงอยู่ในระดับ 1-4 ถูกบันทึกเป็นข้อมูลที่ถูกเก็บไว้ใน JIRA

และนำมาใช้สำหรับการวิเคราะห์ข้อบกพร่องในสังคมเครือข่าย โดยระดับความรุนแรงของข้อบกพร่องประกอบไปด้วย ระดับความรุนแรง 1 (sev1) - Critical; ระดับความรุนแรง 2 (sev2) - High; ระดับความรุนแรง 3 (sev3) - Medium; และระดับความรุนแรง 4 (sev4) – Low

3.3การสำรวจและวิเคราะห์ข้อมูลเกี่ยวกับสาเหตุและระดับความรุนแรงของข้อบกพร่อง

ระดับความรุนแรงของข้อบกพร่องสามารถจำแนกออกเป็น 4 ระดับ ดังนี้

1. **Severity 1: Critical** คือ ระดับความรุนแรงสูงสุดของข้อบกพร่องที่จำเป็นต้องแก้ไขในทันที หากไม่รีบแก้ไข จะทำให้บล็อกการทำงานของระบบ หรือส่วนอื่น ๆ ภายในระบบ ซึ่งผู้ใช้จะไม่สามารถใช้งานต่อได้ เช่น ไม่สามารถล็อกอินเข้าสู่ระบบได้ หรือ กดปุ่ม Next แล้ว Page error

2. **Severity 2: High** คือ ระดับความรุนแรงระดับสูง อาจเป็นผลการทำงานของโปรแกรมที่ผิดพลาด หรือมีส่วนสำคัญของฟังก์ชันการทำงานที่ไม่ถูกต้อง แล้วมีผลกระทบต่อฟังก์ชันการทำงานหลักและผลกระทบในทางธุรกิจ เช่น การคำนวณค่าเงินผิดพลาด เป็นต้น

3. **Severity 3: Medium** คือ ระดับความรุนแรงระดับปานกลาง เป็นข้อบกพร่องที่ไม่ได้กระทบกับการทำงานหลักของระบบ แต่ทำให้ระบบสร้างผลลัพธ์ที่ไม่ถูกต้อง ไม่สมบูรณ์ เช่น คำผิด ส่วนที่แสดงผลบางอย่างไม่ถูกต้อง เป็นต้น

4. **Severity 4: Low หรือ Minor** คือ ระดับความรุนแรงของข้อบกพร่องที่เป็นระดับต่ำสุด ซึ่งไม่มีผลกระทบต่อการใช้งานระบบ เป็นข้อบกพร่องในเรื่องความสวยงาม เช่น สี ขนาดอักษร การจัดวาง layout เป็นต้น

จากการวิเคราะห์หาสาเหตุที่ทำให้เกิดข้อบกพร่องระดับต่างๆ มีสาเหตุดังต่อไปนี้

1. สาเหตุที่ทำให้เกิดข้อบกพร่องที่มีระดับความรุนแรงสูงสุด (Critical)

- เกิดจากการไม่ได้ตรวจสอบความพร้อมของระบบก่อนการเริ่มทดสอบ หรือความพร้อมของสภาพแวดล้อมที่ใช้ในการทดสอบ (Test environment) ก่อนการเริ่มทดสอบจริง เช่น ไม่มีการทำ Pre-test หรือ Smoke test ก่อนการเริ่มทดสอบ เป็นต้น
- ระบบที่พัฒนาเกิดปัญหาขัดข้องทางเครือข่าย (Network) หรือแม่ข่าย (Sever) ณ ขณะเวลาที่ทำการทดสอบระบบ เช่น เกิดระบบล่ม

2. สาเหตุที่ทำให้เกิดข้อบกพร่องที่มีระดับความรุนแรงสูง (High)

- นักพัฒนาระบบ (Developers) ไม่ได้ทำการทดสอบในระดับหน่วยย่อย (unit test) ภายในโปรแกรมให้ครอบคลุมตามข้อกำหนด (Requirements) ทั้งหมดที่มี
- นักพัฒนาระบบ ไม่ได้ทำการทวนสอบโค้ด (Code review) จึงไม่สามารถพบข้อบกพร่องที่อาจหลงเหลืออยู่ในโปรแกรม ซึ่งทำให้ไปตรวจพบในระยะของการทดสอบโดยทีมทดสอบแทน
- ความซับซ้อนของโปรแกรมและฟังก์ชันการทำงาน
- กรณีทดสอบที่ถูกออกแบบไว้ไม่ครอบคลุม

3. สาเหตุที่ทำให้เกิดข้อบกพร่องที่มีระดับความรุนแรงปานกลาง/ ต่ำ (Medium/ Low)

- ข้อจำกัดของทรัพยากร เช่น นักพัฒนามีประสบการณ์น้อย หรือนักทดสอบออกแบบกรณีทดสอบไม่ครอบคลุมเนื่องจากประสบการณ์น้อย รวมถึงจำนวนทรัพยากรบุคคลที่จำกัด
- ข้อจำกัดของเวลาในการพัฒนา เช่น นักพัฒนามีเวลาจำกัด ทำให้อาจมีข้อบกพร่องเล็กน้อยที่มองข้ามไป หรือระยะเวลาทดสอบที่มีจำกัด อาจทำให้นักทดสอบเลือกทดสอบเฉพาะฟังก์ชันงานที่สำคัญแล้วไม่สามารถทดสอบให้ครอบคลุมได้โดยละเอียด ส่งผลให้อาจมีข้อบกพร่องที่หลบหนีไปและตรวจพบโดยลูกค้าได้

3.4 การสร้างแบบจำลอง SNA เพื่อช่วยในการวิเคราะห์ข้อมูล

หลังจากที่ได้วิเคราะห์ถึงสาเหตุที่ทำให้เกิดข้อบกพร่องในแต่ละระดับความรุนแรงต่าง ๆ ซึ่งมีส่วนสำคัญที่ทำให้การพัฒนาซอฟต์แวร์มีความล่าช้า ผู้วิจัยจึงได้เล็งเห็นความสำคัญที่จะประยุกต์ใช้โดยนำหลักการและเครื่องมือ Gephi มาวิเคราะห์ สำรวจ และทำความเข้าใจกราฟโดยหาความสัมพันธ์ระหว่างคุณลักษณะหรือส่วนประกอบของซอฟต์แวร์ภายในโครงการที่ทำให้เกิดข้อบกพร่องเหล่านั้น เพื่อวิเคราะห์หาความสัมพันธ์และความเชื่อมโยงไปยังฟีเจอร์การทำงานที่ทำให้เกิดข้อบกพร่องเหล่านั้นมากที่สุด และจุดที่สนใจอื่น ๆ ทั้งทางด้านการพัฒนาและการทดสอบ โดยเชื่อมโยงความสัมพันธ์ไปยังข้อบกพร่องในแต่ละระดับ เช่น การกำหนดโหนดเป็น Development Task เพื่อบอกว่าใครเป็นผู้รับผิดชอบทางฝั่งการพัฒนาที่ทำให้เกิดข้อบกพร่องที่เป็นจุดสนใจมากที่สุด หรือทางฝั่งการทดสอบ มีการพบข้อบกพร่องที่เป็นประเภท Change Request–CR ที่ถูกรายงานหรือพบโดยใครมากที่สุด ตัวอย่างเช่น Product Owner Business Analyst Tester หรือลูกค้า เป็นต้น นอกจากนี้ ยังวิเคราะห์ผ่านกราฟ SNA สำหรับข้อบกพร่องที่หลบหนี

บทที่ 4
การทดลองและผลการทดลอง

งานวิจัยนี้นำเสนอแนวทางการสร้างภาพข้อมูลเครือข่ายเพื่อเปิดเผยความสัมพันธ์และการสื่อสารระหว่างโหนด เช่น ฟีเจอร์ ข้อบกพร่อง และบุคลากร โดยใช้เทคนิคการวิเคราะห์เครือข่ายสังคม และทฤษฎีกราฟมาใช้สำหรับวิเคราะห์ข้อบกพร่องที่รวบรวมจากโครงการซอฟต์แวร์ภายในธนาคาร และใช้ Gephi เป็นเครื่องมือในการสร้างเครือข่ายสังคมของตัวแสดงที่ถูกระบุเป็นโหนดและการเชื่อมโยงของโหนดเหล่านั้น ซึ่งแนวทางของการวิเคราะห์เครือข่ายภาพนั้นใช้ได้จริงและให้ข้อมูลเชิงลึกในการวิเคราะห์ข้อบกพร่องที่จำเป็นสำหรับกระบวนการพัฒนาซอฟต์แวร์ ขั้นตอนการทดลองอธิบายในหัวข้อต่อไปนี้

4.1 เครื่องมือที่ใช้ในการทดลอง

ในการทดลองนี้ ผู้วิจัยได้เลือกใช้โปรแกรม Gephi [7] ซึ่งเป็นซอฟต์แวร์โอเพนซอร์ซ สำหรับการวิเคราะห์กราฟและเครือข่าย ใช้เครื่องมือนี้สร้างภาพ 3 มิติเพื่อแสดงเครือข่ายขนาดใหญ่ในแบบเรียลไทม์และเร่งการสำรวจ โดยสามารถเข้าถึงข้อมูลเครือข่ายและอนุญาตให้มีการระบุตำแหน่งทางภูมิศาสตร์ การกรองการนำทาง การจัดการ และการจัดกลุ่ม โดยสามารถทำการติดตั้งและดาวน์โหลดได้ฟรี ซึ่งเวอร์ชันที่ผู้วิจัยเริ่มใช้ในการพัฒนาและคิดค้นแบบจำลองคือ Gephi version 9.0.2 และสามารถทำการติดตั้งได้ทั้งในระบบปฏิบัติการ Windows และ Mac OS ซึ่งผู้จัยได้เลือกระบบปฏิบัติ Windows เป็นระบบปฏิบัติการหลักในการทำงานวิจัยในครั้งนี้

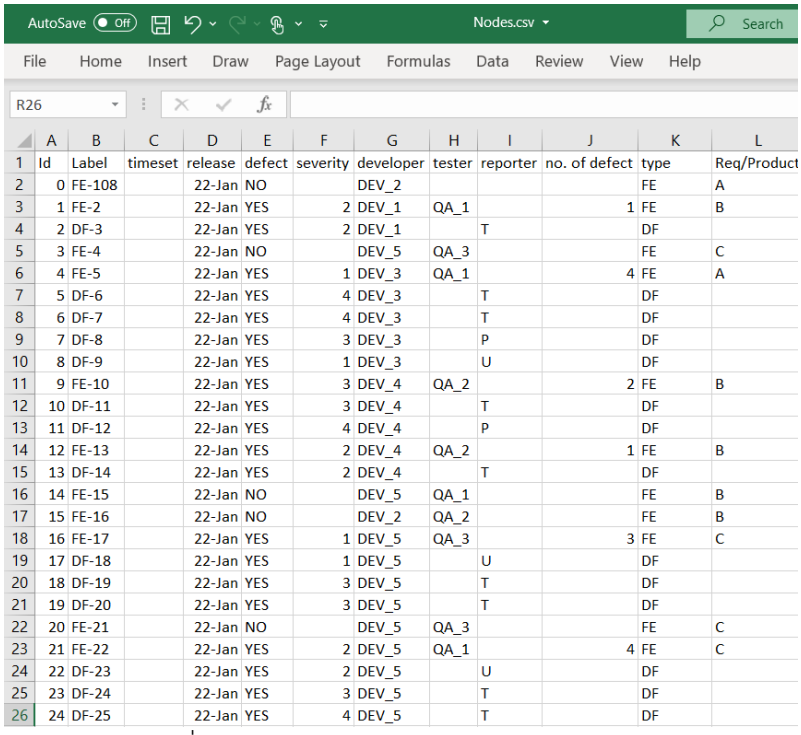
นอกจากนี้ ผู้วิจัยได้ใช้คำสั่ง JQL ในการสืบค้นข้อมูลข้อบกพร่องจาก JIRA Software ซึ่งเป็น Tracking Tools ที่ใช้ในการเก็บคลังข้อมูล Issue ตัวอย่างคำสั่งในการ query ข้อมูลข้อบกพร่องจากระบบ JIRA ดังตารางที่ 1 โดยสุ่มเลือกกลุ่มตัวอย่างมา 1 โครงการจากโครงการภายในธนาคาร เพื่อเป็นแนวทางในการจำลองข้อมูลที่สร้างขึ้นใหม่เพื่อใช้สำหรับวิเคราะห์เครือข่ายของข้อบกพร่องของโพรเจกต์ดังกล่าว ในงานวิจัยครั้งนี้

ตารางที่ 1 คำสั่ง JQL ในการสืบค้นข้อมูลข้อบกพร่องใน Jira Software

ลำดับ	ชุดคำสั่ง
1.	“project = "UAT" AND fixVersion = "Jan 2022" AND type = “Bug” AND cycleName in ("Release 1") and Labels in (Covid_19_App)”

4.2 การเตรียมชุดข้อมูลที่ใช้ในการทดลอง

การเตรียมชุดข้อมูลเพื่อให้ Gephi สามารถอ่านข้อมูลนำเข้าได้ จำเป็นต้องแปลงข้อมูลดิบจากไฟล์ CSV ที่ถูก export ออกมาจากโปรแกรม JIRA ซึ่งเป็นไฟล์ข้อมูลสองเสปรดชีตแยกกัน ประกอบไปด้วย: ไฟล์ที่มีเสปรดชีต "Nodes" และ ไฟล์ที่มีสเปรดชีต"Edges" โดยสเปรดชีท "โหนด" จะมีลักษณะดังนี้ 1. ไฟล์ต้องประกอบไปด้วยคอลัมน์แรกที่ต้องตั้งชื่อคอลัมน์แรกว่า "Id" 2. ทางผู้ใช้งานสามารถตั้งชื่อคอลัมน์อื่น ๆ ได้ และสามารถเพิ่มจำนวนคอลัมน์ได้ตามต้องการ โดยข้อมูลเหล่านี้เป็นคอลัมน์ที่บอกถึงแอตทริบิวต์ ซึ่งอธิบายคุณสมบัติของแต่ละคอลัมน์ โดยที่คีย์ของคอลัมน์ทั้งหมดต้องไม่ซ้ำกัน และเป็นข้อมูลที่ผู้ใช้งานต้องการนำไปแสดงบนกราฟเครือข่าย ดังแสดงในภาพที่ 9



AutoSave Off | Nodes.csv | Search

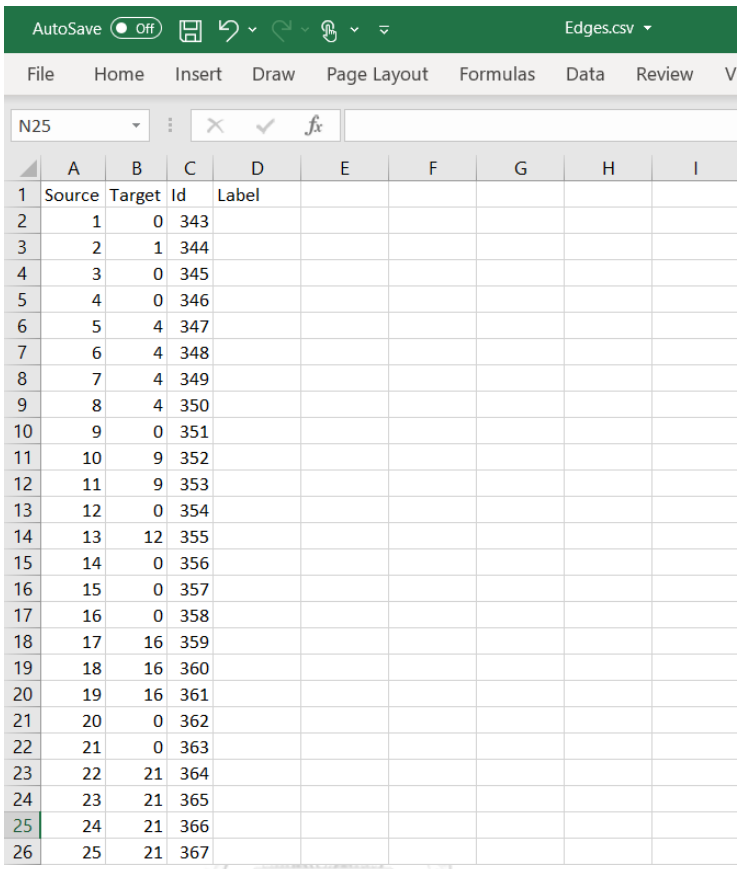
File Home Insert Draw Page Layout Formulas Data Review View Help

R26

	A	B	C	D	E	F	G	H	I	J	K	L
1	Id	Label	timeset	release	defect	severity	developer	tester	reporter	no. of defect	type	Req/Product
2	0	FE-108		22-Jan	NO		DEV_2				FE	A
3	1	FE-2		22-Jan	YES	2	DEV_1	QA_1		1	FE	B
4	2	DF-3		22-Jan	YES	2	DEV_1		T		DF	
5	3	FE-4		22-Jan	NO		DEV_5	QA_3			FE	C
6	4	FE-5		22-Jan	YES	1	DEV_3	QA_1		4	FE	A
7	5	DF-6		22-Jan	YES	4	DEV_3		T		DF	
8	6	DF-7		22-Jan	YES	4	DEV_3		T		DF	
9	7	DF-8		22-Jan	YES	3	DEV_3		P		DF	
10	8	DF-9		22-Jan	YES	1	DEV_3		U		DF	
11	9	FE-10		22-Jan	YES	3	DEV_4	QA_2		2	FE	B
12	10	DF-11		22-Jan	YES	3	DEV_4		T		DF	
13	11	DF-12		22-Jan	YES	4	DEV_4		P		DF	
14	12	FE-13		22-Jan	YES	2	DEV_4	QA_2		1	FE	B
15	13	DF-14		22-Jan	YES	2	DEV_4		T		DF	
16	14	FE-15		22-Jan	NO		DEV_5	QA_1			FE	B
17	15	FE-16		22-Jan	NO		DEV_2	QA_2			FE	B
18	16	FE-17		22-Jan	YES	1	DEV_5	QA_3		3	FE	C
19	17	DF-18		22-Jan	YES	1	DEV_5		U		DF	
20	18	DF-19		22-Jan	YES	3	DEV_5		T		DF	
21	19	DF-20		22-Jan	YES	3	DEV_5		T		DF	
22	20	FE-21		22-Jan	NO		DEV_5	QA_3			FE	C
23	21	FE-22		22-Jan	YES	2	DEV_5	QA_1		4	FE	C
24	22	DF-23		22-Jan	YES	2	DEV_5		U		DF	
25	23	DF-24		22-Jan	YES	3	DEV_5		T		DF	
26	24	DF-25		22-Jan	YES	4	DEV_5		T		DF	

ภาพที่ 9 ข้อมูลจำลองบางส่วนในไฟล์ Nodes.csv

นอกจากไฟล์ Nodes.csv แล้ว ยังมีอีก 1 ไฟล์สำคัญที่ขาดไม่ได้เลย คือ ไฟล์ Edges.csv เป็นการระบุความสัมพันธ์และความเชื่อมโยงของแต่ละโหนดเพื่อที่จะสามารถนำมาวิเคราะห์และสร้างออกมาเป็นกราฟเครือข่ายได้ ซึ่งไฟล์ Edges.csv ต้องประกอบไปด้วยคอลัมน์ Source และ Target เป็นคอลัมน์จำเป็นและสำคัญที่ต้องมีเป็นอย่างน้อยดังแสดงในภาพที่ 10



AutoSave Off Edges.csv

File Home Insert Draw Page Layout Formulas Data Review V

N25

	A	B	C	D	E	F	G	H	I
1	Source	Target	Id	Label					
2	1	0	343						
3	2	1	344						
4	3	0	345						
5	4	0	346						
6	5	4	347						
7	6	4	348						
8	7	4	349						
9	8	4	350						
10	9	0	351						
11	10	9	352						
12	11	9	353						
13	12	0	354						
14	13	12	355						
15	14	0	356						
16	15	0	357						
17	16	0	358						
18	17	16	359						
19	18	16	360						
20	19	16	361						
21	20	0	362						
22	21	0	363						
23	22	21	364						
24	23	21	365						
25	24	21	366						
26	25	21	367						

ภาพที่ 10 ข้อมูลจำลองบางส่วนในไฟล์ Edges.csv

การแปลงข้อมูลประกอบด้วยสองขั้นตอน ในระยะเริ่มแรก จำเป็นต้องสร้างแผ่นงานของโหนด โดยที่แต่ละโหนดจะได้รับ Id ที่ไม่ซ้ำกัน ในขั้นตอนที่สอง จะสร้างสเปรดชีต Edges ซึ่งจะบอกถึงความสัมพันธ์ทั้งหมดระหว่างโหนดซึ่งจะแสดงเป็นความสัมพันธ์ระหว่างโหนดแต่ละ Id

ก่อนที่จะเริ่มสร้างข้อมูล ต้องตรวจสอบให้แน่ใจว่าอักขระแต่ละตัวในแผ่นงาน "มีปฏิสัมพันธ์" ซึ่งมีชื่อที่ไม่ซ้ำกัน และชื่อที่ไม่ซ้ำกันนี้ถูกใช้อย่างสม่ำเสมอในการอ้างอิงถึงอักขระในแผ่นงานทั้งไฟล์ Nodes และ Edges จากนั้นทำการสร้างไฟล์ทั้งสองตามลักษณะการแมปข้อมูลดังที่แสดงในภาพที่ 8 หน้า 17 ให้เป็นไฟล์ CSV

4.3 การดำเนินการสร้างแบบจำลอง

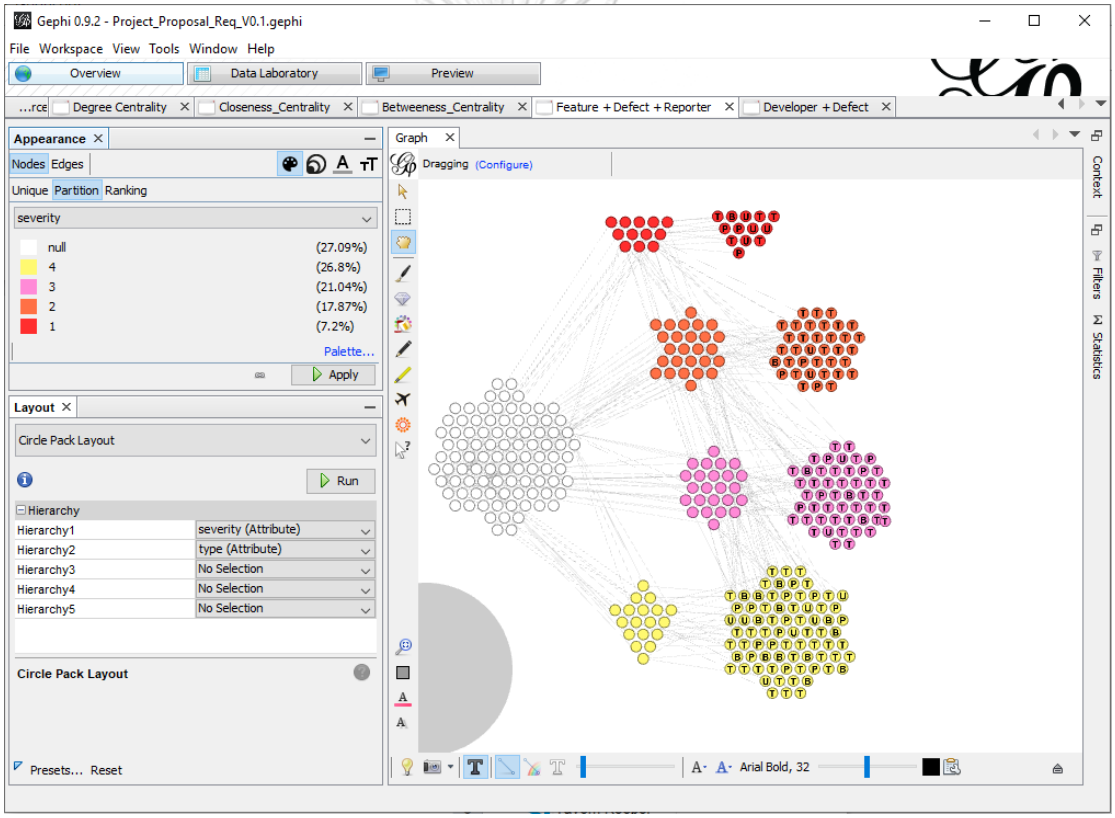
ทำการนำเข้าข้อมูลอินพุตที่เป็นไฟล์ CSV ทั้ง 2 ไฟล์ ได้แก่ Nodes.csv และ Edges.csv ที่ได้เตรียมไว้ในข้อ 4.2 เข้าสู่โปรแกรม Gephi

ขั้นตอนในการสร้างกราฟ SNA ประกอบด้วย

- นำเข้า Node.csv และ Edge.csv ไปในโปรแกรม Gephi
- คลิกที่แถบ “Overview”จากนั้นปรับพารามิเตอร์ในแถบเมนู Appearance โดยเลือกสีของโหนดเพื่อแสดงคลาส Modularity และขนาดของโหนดตามด้วย Degree และเลือก

แอตทริบิวต์ที่ต้องการนำเสนอเพื่อมาแสดงบนกราฟ SNA เพื่อเสนอมุมมองที่อยากให้กราฟแสดง

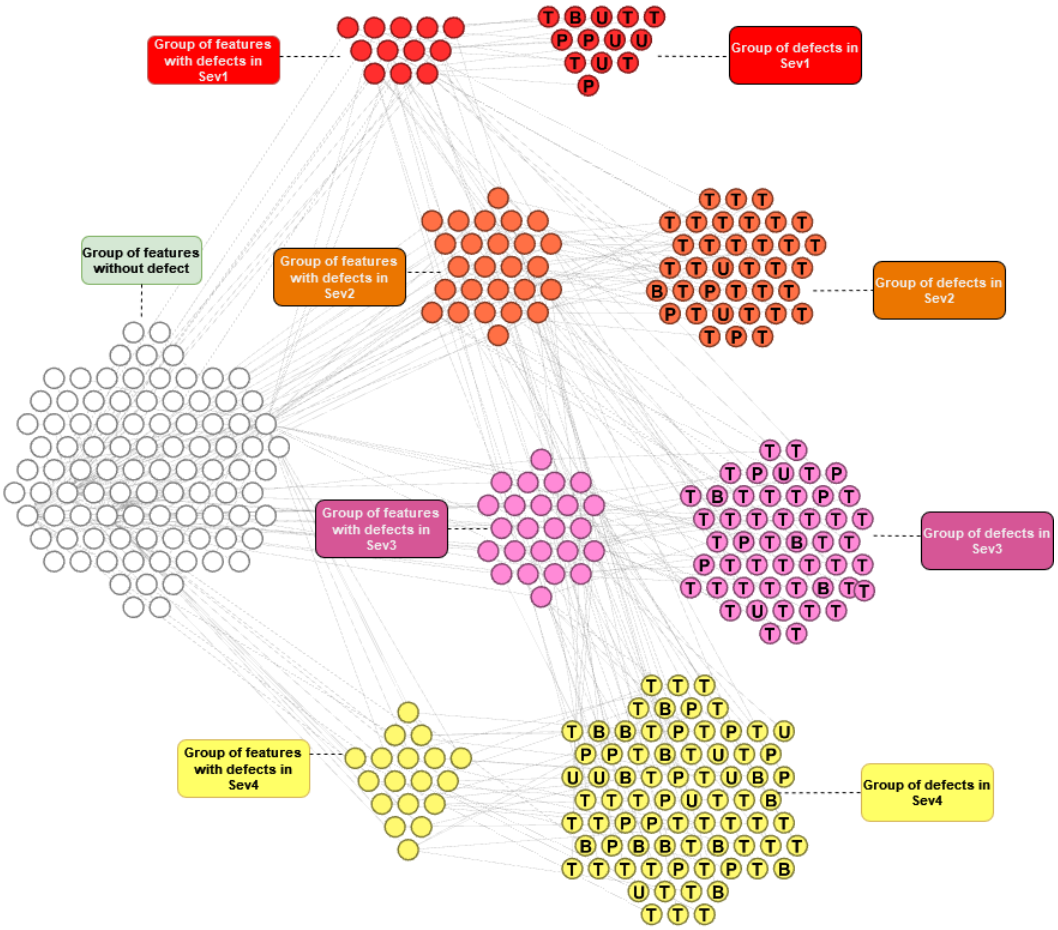
- เลือก Layout สำหรับการแสดงภาพกราฟ SNA ที่แถบเมนู “Layout” ด้านซ้ายล่างได้ตามต้องการ โดยผู้วิจัยเลือกใช้ “Circle Pack Layout” เพื่อจำแนกการแสดงผลออกเป็นลำดับชั้นของข้อมูลที่ต้องการนำเสนอผ่านกราฟ นอกจากนี้ ผู้ใช้ยังสามารถเลือกใช้ Layout อื่น ๆ ตามลักษณะของกราฟที่อยากให้แสดงได้ตามลักษณะของข้อมูลที่ต้องการ ซึ่งมีให้เลือกในการแสดงผลได้อีกมากมาย
- เมื่อตั้งค่าพารามิเตอร์ทั้งหมดเรียบร้อยแล้ว จากนั้นคลิก “Run” ที่แถบเมนู “Layout” เพื่อสร้างกราฟเครือข่าย ดังที่แสดงในภาพที่ 11



ภาพที่ 11 ตัวอย่างโปรแกรม Gephi และแถบเมนู

เมื่อทำการวิเคราห์และปรับพารามิเตอร์ให้ตรงตามสิ่งที่ต้องการนำเสนอตามวิธีการข้างต้น ทำให้ผู้วิจัยได้พยายามทำการทดลองเพื่อสร้างกราฟเครือข่ายให้สามารถนำเสนอข้อมูลข้อบกพร่องเหล่านี้ในเชิงการวิเคราะห์ผ่านแบบจำลองกราฟ SNA ซึ่งได้ผลการทดลองโดยแบ่งออกมาเป็น 4 แบบจำลองตามมุมมองที่ต้องการนำเสนอดังต่อไปนี้

4.3.1 แบบจำลองที่ 1 มุมมองทางด้านการทดสอบ (Testing tasks view)



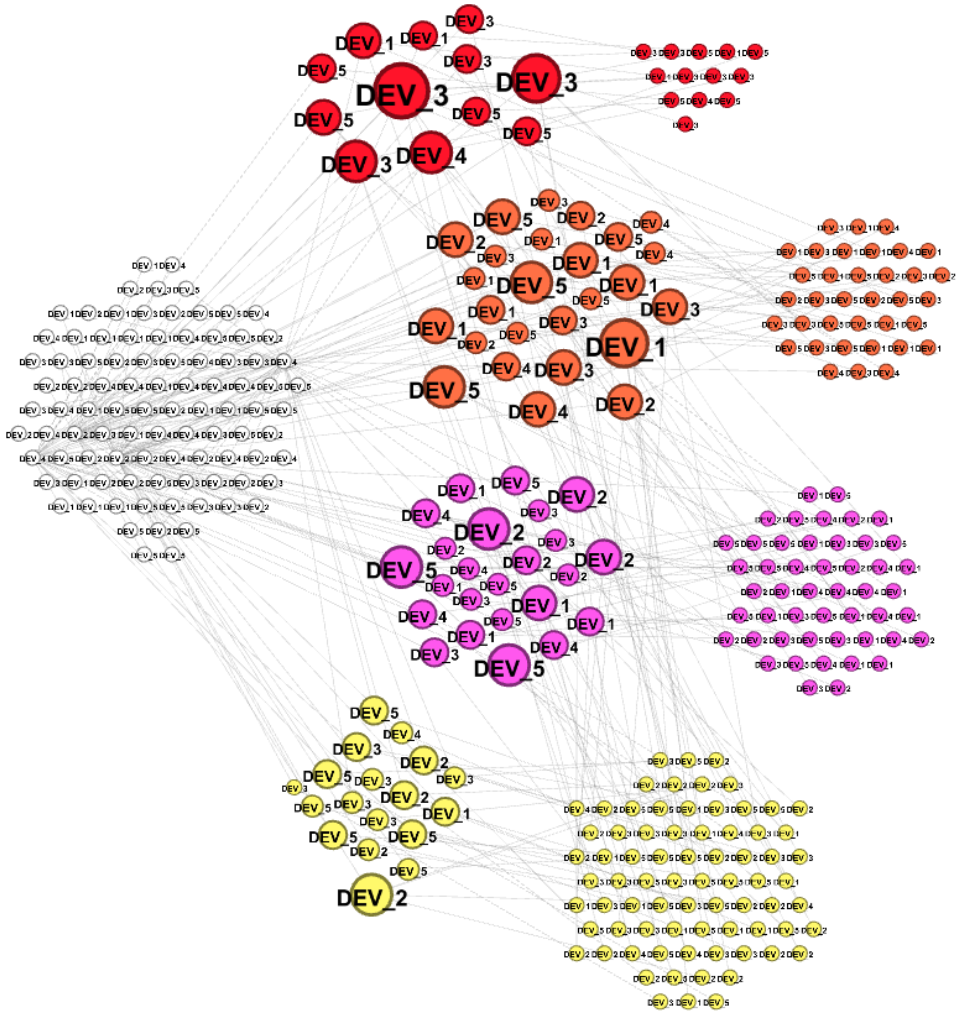
ภาพที่ 12 แบบจำลองกราฟ SNA สำหรับด้านการทดสอบ

ภาพที่ 12 แสดงโหนดในเลเยอร์กลางซึ่งบอกถึงฟีเจอร์ที่มีอยู่ในโมดูลบางโมดูลที่มีการเชื่อมโยงกัน ฟีเจอร์แต่ละอย่างอาจมีข้อบกพร่องหลายอย่างที่มีระดับความรุนแรงแตกต่างกัน สีของโหนดของฟีเจอร์ถูกกำหนดโดยระดับความรุนแรงของข้อบกพร่องสูงสุด โหนดของฟีเจอร์ที่ไม่พบข้อบกพร่องจะเป็นสีขาว เลเยอร์สุดท้ายแสดงคำอธิบายข้อบกพร่องที่เชื่อมโยงกับฟีเจอร์บางอย่าง การสังเกตว่าโหนดของฟีเจอร์ที่มีสีเหลืองในเลเยอร์กลางเชื่อมต่อกับคลัสเตอร์ของโหนดข้อบกพร่องสีเหลืองในเลเยอร์สุดท้ายเท่านั้น ในขณะที่โหนดของฟีเจอร์สีแดงสามารถเชื่อมโยงไปยังคลัสเตอร์ใดๆ ในเลเยอร์สุดท้ายได้ สำหรับแต่ละรุ่น ข้อบกพร่องที่พบในแต่ละฟีเจอร์สามารถจัดประเภทและติดป้ายกำกับบนโหนดในเลเยอร์สุดท้ายได้ดังนี้:

- **P** หรือ **U** หมายถึงข้อบกพร่องที่พบใน Production และรายงานโดย **P**roduct Owner (PO) หรือ Business **U**nit (BU)

- **B** หมายถึงข้อบกพร่องที่หลบหนีซึ่งรายงานในแบบฟอร์มคำขอเปลี่ยนแปลงโดยนักวิเคราะห์ธุรกิจ
- **T** หมายถึงข้อบกพร่องที่หลบหนีซึ่งพบระหว่าง SIT หรือ UAT และรายงานโดยผู้ทดสอบ
- สีของโหนดข้อบกพร่องถูกกำหนดตามระดับความรุนแรงตามที่ระบุไว้ก่อนหน้านี้

4.3.2 แบบจำลองที่ 2 มุมมองทางด้านการพัฒนาระบบ (Development tasks view)

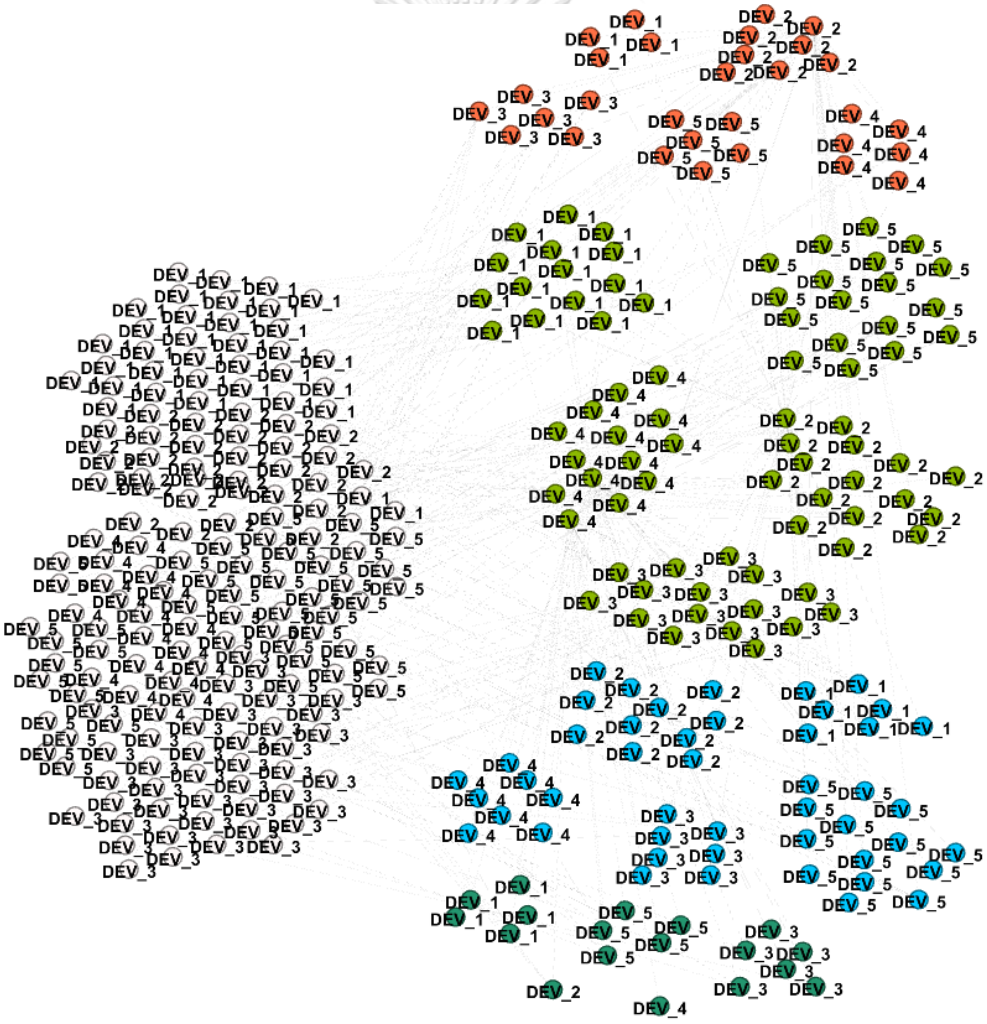


ภาพที่ 13 แบบจำลองกราฟ SNA สำหรับภาพความสัมพันธ์ระหว่างนักพัฒนาและข้อบกพร่อง

แบบจำลองที่สอง นำเสนอมุมมองในอีกอื่นของการพัฒนาระบบ ซึ่งมีฟีเจอร์ที่มีอยู่ในสี่โมดูลที่เลือกดังแสดงในภาพที่ 13 เพื่อมุ่งเน้นไปที่นักพัฒนาที่รับผิดชอบในการพัฒนาฟีเจอร์บางฟีเจอร์ของโหนดที่แสดงถึงฟีเจอร์ในเลเยอร์กลางจะมีป้ายกำกับชื่อของนักพัฒนาผู้รับผิดชอบ หากพบข้อบกพร่อง โหนดของฟีเจอร์ในเลเยอร์กลางจะเป็นสีที่มีข้อบกพร่องที่มีความรุนแรงสูงสุดตามที่

อธิบายไว้ก่อนหน้า มิฉะนั้นโหนดของฟีเจอร์จะแสดงเป็นสีขาว จำนวนข้อบกพร่องที่พบในแต่ละฟีเจอร์สามารถบอกได้ตามขนาดโหนด เลเยอร์สุดท้ายแสดงคลัสเตอร์สี่ระดับของระดับความรุนแรงต่างๆ ด้วยสีที่แตกต่างกันตามที่อธิบายไว้ก่อนหน้านี้ แต่ละโหนดแสดงถึงข้อบกพร่องเดียว การกระจายของจำนวนข้อบกพร่องที่จัดหมวดหมู่ในแต่ละระดับความรุนแรงนั้นเป็นเรื่องปกติ กล่าวคือ จำนวนของข้อบกพร่องที่สำคัญจะน้อยที่สุด ในขณะที่จำนวนข้อบกพร่องที่มีความรุนแรงต่ำจะมากที่สุด และ DEV_3 ก็เป็นตัวเลือกที่โดดเด่นและเหมาะสำหรับการปรับปรุง

4.3.3 แบบจำลองที่ 3 มุมมองทางด้านความต้องการและการจัดสรรทรัพยากร (Requirements and Resources view)

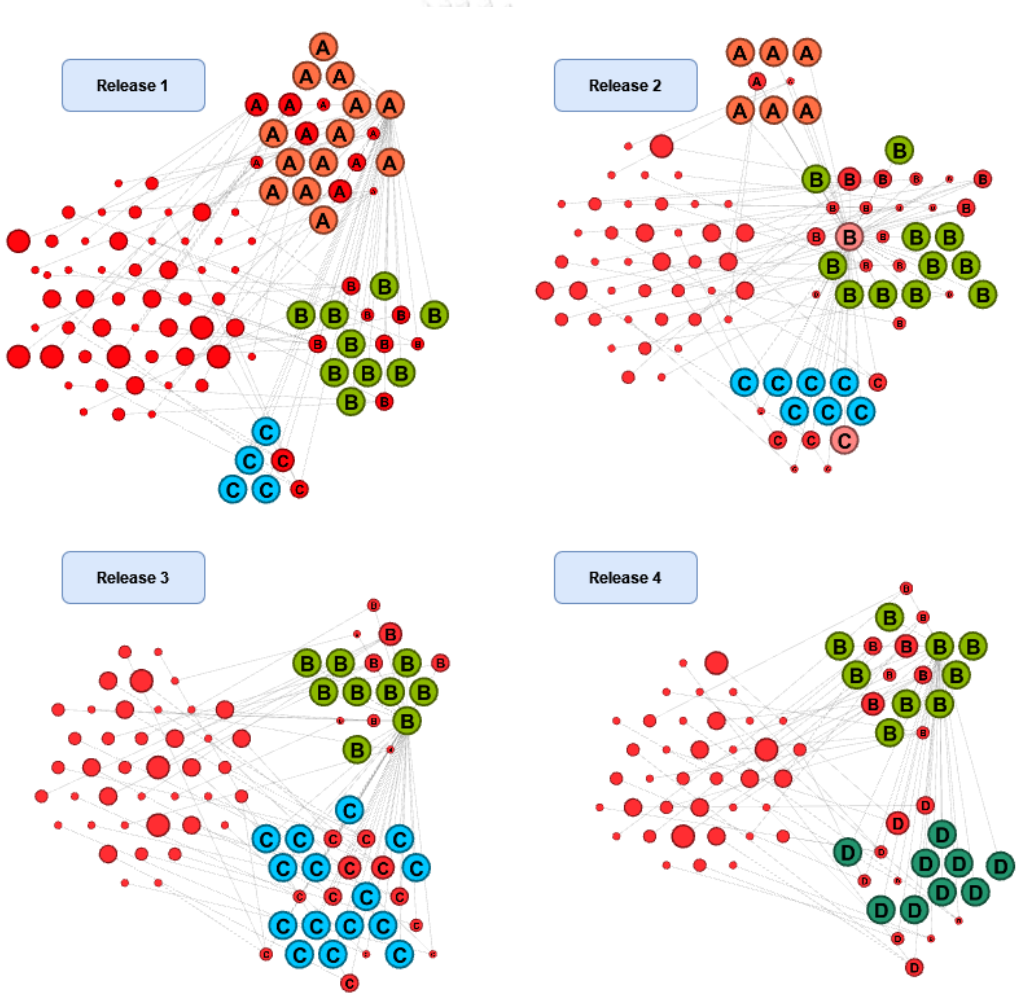


ภาพที่ 14 แบบจำลองกราฟ SNA ด้านความต้องการและการจัดสรรทรัพยากร

ภาพที่ 14 แสดงมุมมองในอีกมุมที่ทำให้เห็นว่า มีการกระจายตัวของ Requirements และนักพัฒนาผู้รับผิดชอบได้เหมาะสมหรือไม่ ซึ่งถือเป็นการจัดสรรทรัพยากรและสามารถมองเห็น

ภาพรวมของระบบว่าการทำงานในตอนนี้ มีการกระจุกหรือกระจายตัวของ resources มากน้อยเพียงใด เพื่อนำผลกราฟเครือข่ายที่ได้ไปจัดสรรให้กับ resources ที่มีอยู่อย่างจำกัด และนำผลที่ได้ไปปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ในการจัดลำดับความสำคัญของซอฟต์แวร์ในแต่ละ release ได้ โดยไม่ให้งานไปกระจุกอยู่ที่ทีมใดทีมหนึ่ง หรือนักพัฒนาคนใดคนหนึ่ง ข้อเสนอแนะดังกล่าวจึงมีส่วนช่วยให้ผู้บริหารโครงการสามารถนำไปใช้เพื่อจัดการกับ resources ในการจ่ายงานแต่ละคนให้มุ่งเน้นการทำงานไปที่แต่ละ product ได้อย่างมีประสิทธิภาพ

4.3.4 แบบจำลองที่ 4 มุมมองภาพรวมของผลิตภัณฑ์และข้อบกพร่องที่เกิดขึ้นในแต่ละ Release



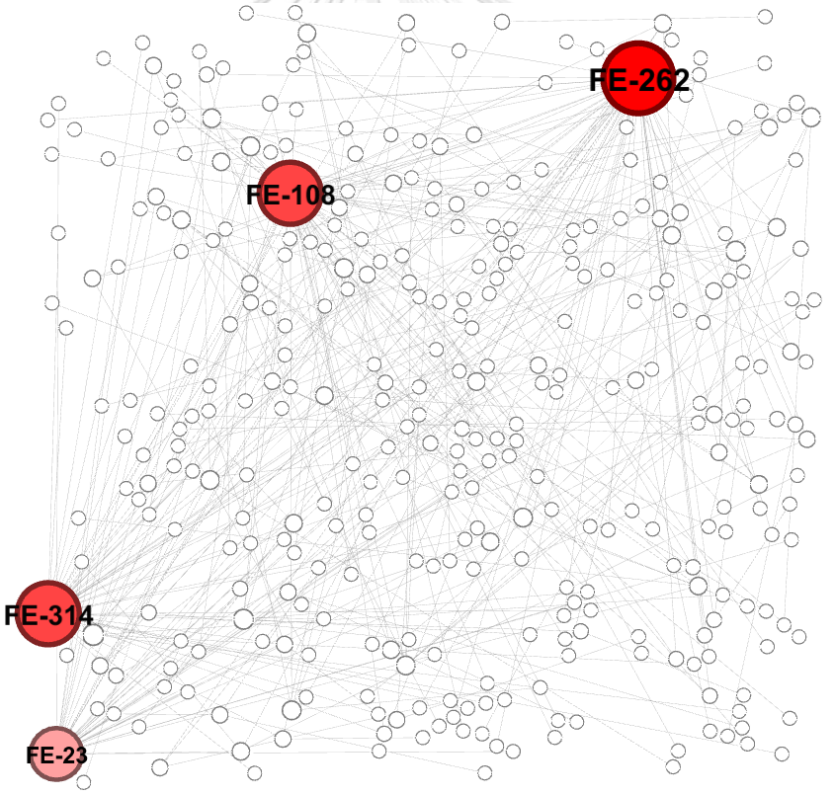
ภาพที่ 15 แบบจำลองกราฟ SNA ด้านผลิตภัณฑ์และข้อบกพร่องที่เกิดขึ้นในแต่ละ Release

ภาพที่ 15 เป็นแบบจำลองที่นำเสนอมุมมองของทั้งผลิตภัณฑ์ที่เกิดข้อบกพร่องในแต่ละ Release ตั้งแต่ Release1 - 4 โดยจากกราฟจะเห็นได้ว่า มีผลิตภัณฑ์ A, B, C และ D ซึ่งถูกแบ่งแยกโดยสี และโหนดทางซ้ายมือแสดงถึงจำนวนข้อบกพร่องที่เกิดขึ้นทั้งหมดในผลิตภัณฑ์นั้นซึ่งถูกแสดง

ด้วยโหนดที่มีสีแดงและขนาดของโหนดบ่งบอกถึงระดับความรุนแรงของข้อบกพร่องที่ต่างกัน ส่วนกลุ่มผลิตภัณฑ์ที่ไม่ใช่สีแดงคือผลิตภัณฑ์ที่ไม่พบข้อบกพร่องเกิดขึ้นใน Release นั้น ๆ ซึ่งแบบจำลองนี้ทำให้เห็นถึงภาพรวมของข้อบกพร่องที่เกิดขึ้นในแต่ละผลิตภัณฑ์และในแต่ละ Release ได้ชัดเจนยิ่งขึ้น

4.4 ผลการทดลอง

จากการสร้างแบบจำลองกราฟ SNA ทั้ง 4 โครงสร้างข้างต้น ผู้วิจัยได้นำเอาข้อมูลชนิดเดียวกันจากการสร้างแบบจำลองที่ได้นั้นมาทำการทดสอบด้วยการวัดค่าความเป็นศูนย์กลางที่สามารถทดสอบผ่านฟังก์ชันการทำงานของโปรแกรม Gephi ทำให้ได้กราฟเครือข่าย 3 กราฟที่ถูกสร้างขึ้นเพื่อเปิดเผยโหนดผู้ทรงอิทธิพลที่วัดจากค่าความเป็นศูนย์กลางของเครือข่าย 3 ค่า ได้แก่: 1) Degree Centrality 2) Closeness Centrality และ 3) Betweenness Centrality

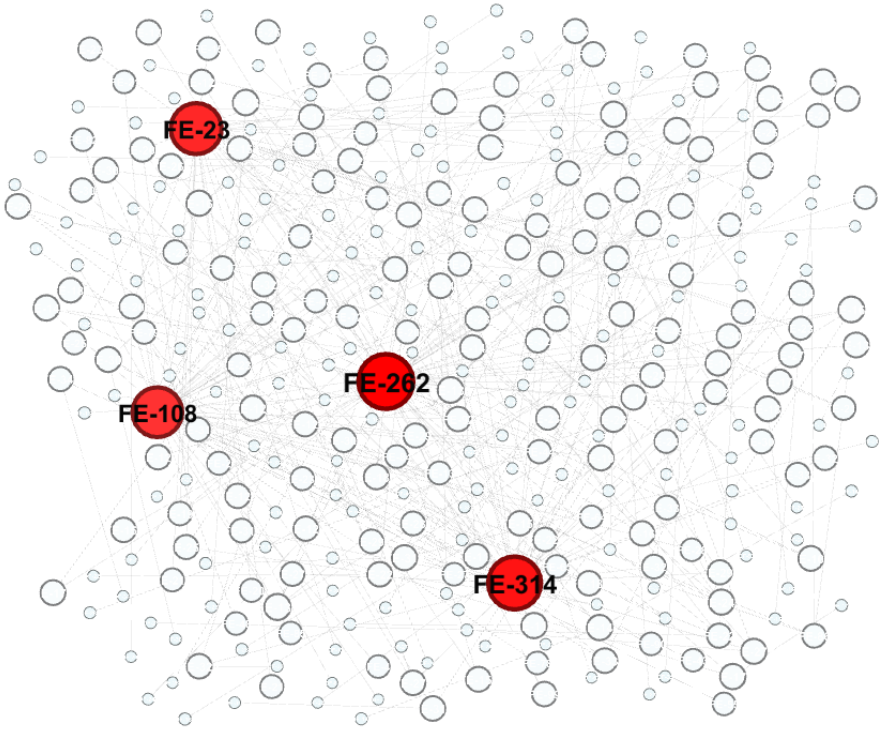


ภาพที่ 16 กราฟ SNA เพื่อบ่งชี้โหนดสำคัญจากค่า Degree Centrality

ในเครือข่ายสังคม Software ของโครงการตัวอย่างนี้มีข้อมูลทั้งหมด 346 โหนด มีเส้นที่เชื่อมโยงระหว่างโหนด หรือ ฟีเจอร์ ทั้งสิ้น 345 เส้น

กราฟเครือข่ายที่แสดงในรูปที่ 16 รายงานโหนดที่ใหญ่ที่สุดของ FE-262 โดยมีค่าสูงสุดของ Degree Centrality=50 ค่าสูงสุดของ Closeness Centrality=0.689394 (โหนดที่ใหญ่ที่สุดของ FE-262 ในภาพที่ 17) และ ค่า Betweenness Centrality = 0.067555 ในขณะที่ FE-262 บรรลุค่าของ Degree Centrality=43, Closeness Centrality=0.644295 และค่าสูงสุดของ Betweenness Centrality=0.074307 นำเสนอเป็นโหนดที่ใหญ่ที่สุดในภาพที่ 18

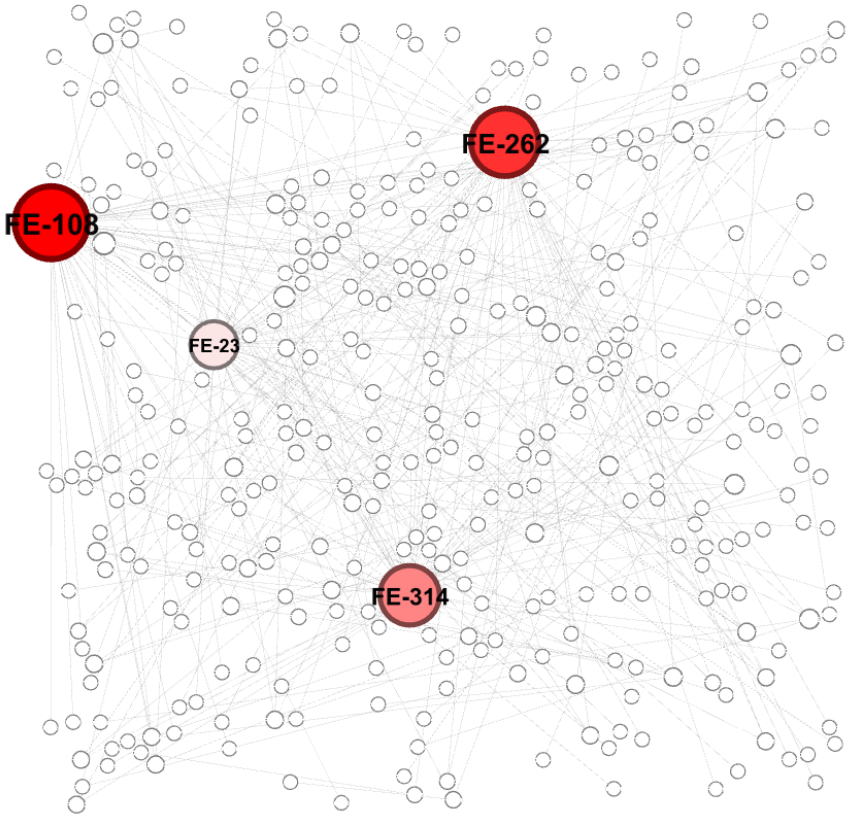
จากผลการทดลองและการวัดค่าความเป็นศูย์กลางสามารถบ่งชี้ได้จากกราฟเครือข่าย เมื่อต้องการค้นหาฟีเจอร์ที่ทรงอิทธิพลที่สุด ที่เป็นตัวทำให้เกิดข้อบกพร่องในฟีเจอร์นั้น ๆ มากที่สุด ต้องค้นหาโดยใช้วิธี Degree Centrality ซึ่งแสดงได้จากกราฟในภาพที่ 16 เห็นได้ชัดว่าโหนดที่เป็น FE-262 เป็นฟีเจอร์ที่ทรงอิทธิพลที่สุด มีความเชื่อมโยงกับฟีเจอร์อื่น ๆ มากที่สุด ซึ่งจะเห็นว่า มีลิงก์ไปยังโหนดอื่น ๆ ในเครือข่ายสูงสุด เป็นการวัดอิทธิพลหรือความสำคัญของโหนดที่มีประสิทธิภาพสูง โดยโหนดที่มีคอนเนคชันมากกว่ามักจะมีอำนาจและมองเห็นได้ชัดเจนขึ้นในสภาพแวดล้อมของโครงการนั้น



ภาพที่ 17 กราฟ SNA เพื่อบ่งชี้โหนดสำคัญจากค่า Closeness Centrality

ลำดับถัดมา คือ การค้นหาโหนดที่ทรงอิทธิพลด้วยวิธี Closeness Centrality ซึ่งจากกราฟที่แสดงออกมา คือ โหนด FE-262 เป็นโหนดที่ทรงอิทธิพลที่สุด และสามารถอธิบายได้ว่า FE-262 เป็น

โหนด หรือ ฟีเจอร์ที่มีการเชื่อมต่อกับฟีเจอร์อื่น ๆ และเข้าถึงฟีเจอร์อื่น ๆ ได้เร็วที่สุดโดยพิจารณาจากระยะทางระหว่างโหนดที่สั้นที่สุด (shortest path) ซึ่งเมื่อโหนดนั้นเข้าถึงฟีเจอร์ต่างๆได้เร็วที่สุด นั่นหมายความว่าโหนดมีความสำคัญต่อสังคมซอฟต์แวร์ในโครงการนี้ที่สุด



ภาพที่ 18 กราฟ SNA เพื่อบ่งชี้โหนดสำคัญจากค่า Betweenness Centrality

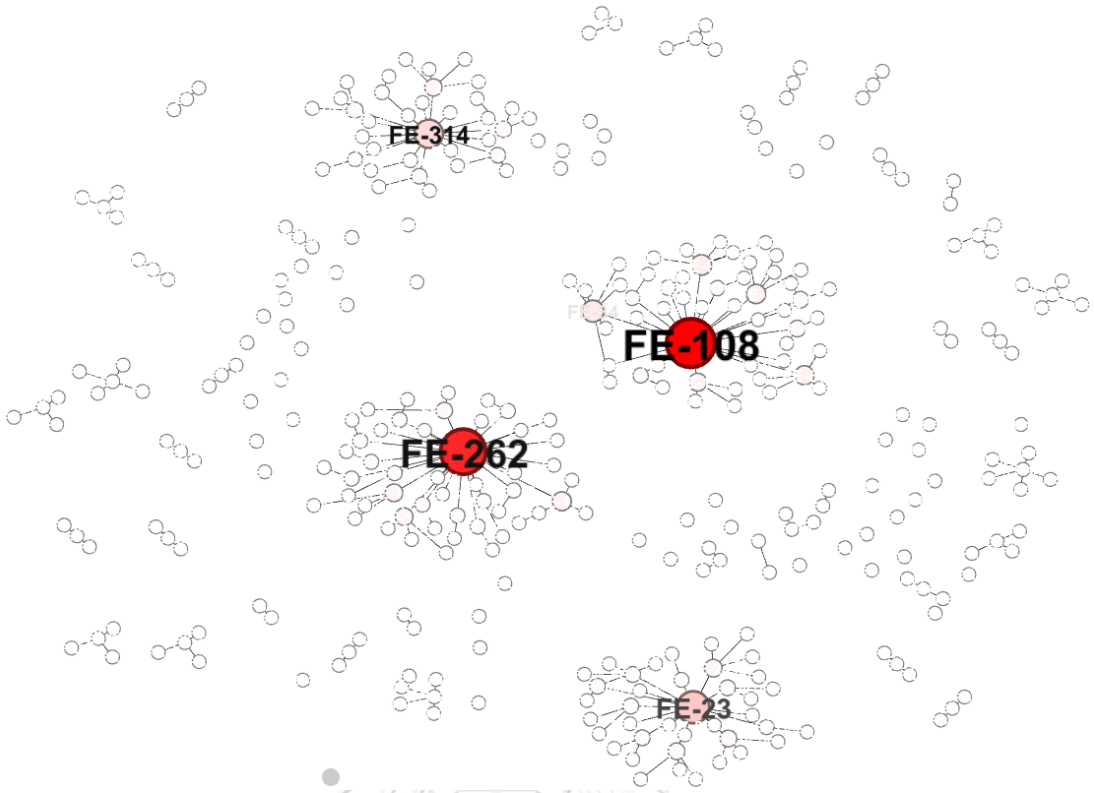
และวิธีสุดท้ายคือ การค้นหาโหนดทรงอิทธิพลด้วยวิธี Betweenness Centrality ในกรณี Betweenness Centrality ดังที่แสดงในภาพที่ 18 ค่าที่ได้คือโหนด FE-108 ถือเป็นโหนดทรงอิทธิพลที่สุด เพราะมีค่า Betweenness สูงสุด ซึ่งอธิบายในบริบทของซอฟต์แวร์ได้ว่า โหนดนี้เป็นฟีเจอร์ที่เชื่อมโยงหรือเป็นทางผ่านของฟีเจอร์อื่น ๆ ที่เกี่ยวข้องกับข้อบกพร่อง ซึ่งข้อบกพร่องเหล่านั้นที่เกิดขึ้นจะต้องทำการ Reproduce โดยผ่านขั้นตอนการทำงานในแต่ละขั้นตอนที่เป็นส่วนของฟีเจอร์นั้น ๆ ด้วยเช่นกัน

การแสดงภาพเครือข่ายที่แสดงในภาพที่ 16, 17 และ 18 สามารถบอกถึงความสำคัญของโหนดบางโหนดในเครือข่ายได้ ซึ่งการวิเคราะห์ข้อบกพร่องในแต่ละคุณสมบัติสามารถสรุปได้โดยใช้อัลกอริธึมการตรวจจับชุมชนในการตรวจจับเครือข่ายสังคม การตรวจหาชุมชนเป็นเทคนิคของการ

วิเคราะห์เครือข่ายสังคมออนไลน์ที่สามารถแยกข้อมูลออกเป็นกลุ่มที่เกี่ยวข้องกันด้วยโมดูลาร์ ซึ่งจากข้อมูลข้างต้น โมดูลาร์สามารถแบ่งออกเป็น 22 คลาส โดยกลุ่มที่เกี่ยวข้องมากที่สุด คือ Node FE-262 ถูกกำหนดให้อยู่ในคลาส 9 และมีโมดูลาร์ขั้นต่ำ 0.79% โดยมี Degree Centrality สูงสุดที่ 50 ซึ่งเป็นค่าสูงสุด ซึ่งบ่งชี้ว่า FE-262 เป็นผู้มีอิทธิพล

ส่วนโหนดที่สำคัญและมีอิทธิพลรองลงมา ดังจะเห็นเป็นโหนดที่ถูกแสดงด้วยขนาดและสีที่ชัดเจนรองลงมา ได้แก่ FE-314 FE-108 และ FE-23 ซึ่งอธิบายได้ว่า กลุ่มของฟีเจอร์ดังกล่าวยังมีค่า Degree Centrality Betweenness Centrality และ Closeness Centrality สูงอยู่ ซึ่งยังทำให้ปรากฏเป็นโหนดที่เด่นชัดจากกราฟที่แสดงผลของค่าความเป็นศูนย์กลางสูงสุดดังที่ได้กล่าวมาข้างต้น และแสดงถึงฟีเจอร์ที่ถูกพบข้อบกพร่องมากที่สุดรองลงมาเช่นกัน ซึ่งมีความเกี่ยวโยงกับข้อบกพร่องที่หลบหนีมาจากรอบการทดสอบก่อนหน้า และหลุดรอดมาถึงรอบปัจจุบัน โดยมีความเชื่อมโยงจากขั้นตอนในการทดสอบจากฟีเจอร์นั้นไปอีกฟีเจอร์หนึ่ง ซึ่งระหว่างทางมีขั้นตอนที่ทำให้เกิดข้อบกพร่องอยู่ อาจจะด้วยความบกพร่องหรือความเลินเล่อที่นักทดสอบเองทดสอบระบบไม่เจอหรือนักพัฒนายังคงทำให้เกิดข้อบกพร่องในฟีเจอร์นั้นอยู่ โดยสามารถแสดงออกมาในรูปแบบของกราฟ SNA และผลลัพธ์ที่ได้ ส่วนโหนดอื่น ๆ เป็นแค่โหนดหรือฟีเจอร์ทั่ว ๆ ไปที่ยังไม่พบข้อบกพร่องในฟีเจอร์เหล่านั้นมากนัก

เมื่อผู้วิจัยได้ทำการวิเคราะห์เพิ่มเติมจากผลการทดลองที่ได้จากกราฟเพื่อวัดค่าความเป็นศูนย์กลางทั้งสามกราฟข้างต้น ผู้วิจัยได้เล็งเห็นว่า กราฟทั้งสามมีลักษณะความสัมพันธ์หรือมีการพึ่งพาต่อกันในแต่ละโหนดที่บ่งชี้ถึงฟีเจอร์เป็นจำนวนมาก ซึ่งจะส่งผลต่อการทดสอบระบบหากฟีเจอร์เหล่านั้นเกิดข้อบกพร่องขึ้น นั่นหมายความว่า นักทดสอบระบบจำเป็นจะต้องทำการทดสอบระบบในทุก ๆ ฟีเจอร์ที่เกี่ยวข้องโยงกันหรือมี dependency ต่อกันเป็นจำนวนมากมายมหาศาล ซึ่งจะทำให้เปลืองทรัพยากรและเวลาในการทดสอบระบบเป็นอย่างยิ่ง เนื่องจากจำเป็นต้องทำการทดสอบระบบทั้งหมดที่มีความเกี่ยวข้องกันอีกครั้ง (Regression test) ทำให้ผู้วิจัยนำข้อมูลเหล่านี้กลับมาวิเคราะห์โดยพิจารณาจากค่า Betweenness Centrality ของโหนดที่มีค่าสูงสุด โดยเลือกนำโหนดเหล่านั้นมาพิจารณาความสัมพันธ์ที่เกิดและทำการลด dependency ระหว่างกันลงจากการลดเส้นเชื่อมความสัมพันธ์ในไฟล์ Edge เพื่อแบ่งกลุ่มกราฟย่อยออกมาภายในกราฟ SNA เดียวกัน ให้เห็นถึงกลุ่มของของฟีเจอร์ที่มีการพึ่งพากันและกลุ่มของฟีเจอร์ที่เป็นอิสระต่อกันและแยกออกจากกันอย่างชัดเจน และเมื่อทำการสร้างกราฟจากการปรับลดความความสัมพันธ์ดังกล่าว จึงได้ผลการทดลองดังที่แสดงในภาพที่ 19



ภาพที่ 19 กราฟ SNA เพื่อแบ่งกลุ่มย่อยของโหนดสำคัญที่ได้จากค่า Betweenness Centrality

จากภาพที่ 19 เมื่อพิจารณาจากค่า Betweenness Centrality สูงสุด สามารถบอกได้ว่าหากเกิดข้อบกพร่องขึ้นในโหนดใดก็ตามในกราฟดังกล่าว สามารถพิจารณาได้ว่า ควรแก้ข้อบกพร่องที่อยู่ภายในฟีเจอร์ที่มีค่า Betweenness Centrality สูงที่สุดก่อน เพื่อให้กราฟมีกลุ่มของโหนดหรือฟีเจอร์ที่แยกออกจากกันและเป็นอิสระต่อกันอย่างชัดเจน เมื่อได้กราฟที่มีลักษณะที่เห็นถึงการพึ่งพาระหว่างโหนดและกลุ่มของโหนดที่เป็นอิสระต่อกันอย่างสิ้นเชิง จะทำให้นักทดสอบระบบสามารถเลือกพิจารณาทดสอบระบบเฉพาะกลุ่มที่มี dependency ต่อกันเท่านั้น และตัดกลุ่มของฟีเจอร์ที่ไม่เกี่ยวข้องออกจากการทดสอบซ้ำในรอบ Regression test ได้หากเกิดข้อบกพร่องขึ้น และจำนวนการทดสอบจะลดลงอย่างมาก โดยแสดงให้เห็นดังตัวอย่างในกราฟจากภาพที่ 19 เช่น เมื่อเกิดข้อบกพร่องขึ้นในกลุ่มของฟีเจอร์ FE-23 นักทดสอบระบบก็จะสามารถเลือกทดสอบเฉพาะกลุ่มของฟีเจอร์ที่เชื่อมโยง หรือมี dependency เฉพาะภายในโหนดที่เชื่อมโยงกับ FE-23 เท่านั้น โดยไม่จำเป็นต้องไปทดสอบใหม่ทั้งระบบ เป็นต้น

บทที่ 5
สรุปผลการวิจัย

วิทยานิพนธ์นี้ได้นำเสนอวิธีการวิเคราะห์ข้อมูลข้อบกพร่องของโครงการซอฟต์แวร์จากโดเมนธนาคาร โดยกราฟเครือข่ายสังคมจำนวนหนึ่งถูกสร้างขึ้นเพื่อแสดงภาพความสัมพันธ์และความเชื่อมโยงระหว่างโหนดหรือตัวแสดงที่กำหนดไว้ เช่น คุณลักษณะ ผู้รับผิดชอบ และข้อบกพร่อง การวัดความเป็นศูนย์กลางของเครือข่าย ซึ่งรวมถึงระดับความเป็นศูนย์กลาง ความใกล้ชิด และความสัมพันธุ์ระหว่างกัน ถูกนำไปใช้เพื่อเปิดเผยส่วนประกอบซอฟต์แวร์ที่โดดเด่น ผลการวิจัยจากแบบจำลองเครือข่ายจะสามารถแนะนำช่องว่างสำหรับการปรับปรุงกระบวนการพัฒนาซอฟต์แวร์แทนที่จะค้นหาข้อมูลจำนวนมากจากข้อบกพร่องที่จัดเก็บไว้ในที่เก็บแบบคงที่ เช่น Jira แต่สามารถนำแบบจำลองกราฟการวิเคราะห์ที่ดึงข้อมูลจำนวนมากมาแสดงให้เห็นเป็นข้อสรุปของโครงการได้ในกราฟเพียงไม่กี่กราฟ และสามารถเห็นช่องโหว่ในการพัฒนาได้หลายมุมมอง การวิเคราะห์ข้อบกพร่องโดยใช้เทคนิค SNA จะเป็นประโยชน์มากกว่าในการป้องกันไม่ให้เกิดปัญหาซ้ำ ๆ จากข้อบกพร่องที่หลบหนี ลดวิธีแก้ปัญหาชั่วคราวที่ไม่จำเป็น ส่งเสริมประสิทธิภาพการจัดการโครงการซอฟต์แวร์ และภาพลักษณ์ขององค์กร

ผลลัพธ์ที่ผู้วิจัยได้จากสร้างแบบจำลองและการทดลองในงานวิจัยครั้งนี้ แบ่งออกเป็นข้อสรุปได้ 4 มุมมอง คือ 1) มุมมองด้านการพัฒนาซอฟต์แวร์ของนักพัฒนาระบบ 2) มุมมองด้านการทดสอบ 3) มุมมองด้านความความต้องการและการจัดสรรทรัพยากร 4) มุมมองด้านผลิตภัณฑ์และข้อบกพร่องที่เกิดขึ้นในแต่ละ Release ซึ่งมุมมองทั้ง 4 ด้าน สามารถบอกได้ว่า โหนดผู้ทรงอิทธิพลในแต่ละมุมมองมีความสำคัญต่อเครือข่ายในแต่ละมุมมองอย่างไร เช่น มุมมองของนักพัฒนา การพบข้อบกพร่องที่เกิดขึ้นทำให้รู้ว่ามีนักพัฒนาคนใดเป็นผู้รับผิดชอบที่ทำให้เกิดข้อบกพร่องมากที่สุด และสามารถให้ข้อเสนอแนะแก่นักพัฒนาในแต่ละคนเพื่อแก้ปัญหาได้อย่างตรงจุดและพึงระวังในส่วนของฟังก์ชันการทำงานที่ทำให้เกิดข้อบกพร่องอย่างระมัดระวังมากยิ่งขึ้น มุมมองที่สอง คือ มุมมองด้านการทดสอบ มุมมองของกราฟนี้ทำให้เห็นว่ามีจำนวนข้อบกพร่องที่หลบหนีไปในแต่ละรอบและถูกพบโดยใครบ้าง และสามารถบอกถึงระดับความรุนแรงของข้อบกพร่องที่เกิดขึ้นภายใต้ฟีเจอร์นั้น ๆ ได้อีกด้วย มุมมองที่สาม คือ มุมมองด้านการความต้องการและการจัดสรรทรัพยากร สามารถบอกได้ว่าการจัดสรรทรัพยากรในโครงการนี้มีการกระจายงานตามความต้องการหรือฟังก์ชันการทำงานเหมาะสมหรือไม่ และมุมมองสุดท้าย คือ มุมมองด้านผลิตภัณฑ์และข้อบกพร่องที่เกิดขึ้นในแต่ละ Release บ่งบอกถึงภาพรวมของผลิตภัณฑ์ที่ไม่มีและมีจำนวนข้อบกพร่องเกิดขึ้นในผลิตภัณฑ์นั้น

โดยแสดงออกมาผ่านกราฟในแต่ละ Release พร้อมทั้งอธิบายระดับความรุนแรงของข้อบกพร่องได้ด้วยขนาดของโหนด ซึ่งผลวิเคราะห์เหล่านี้จะช่วยให้เป็นประโยชน์ต่อการปรับปรุงกระบวนการซอฟต์แวร์เป็นอย่างมาก นอกจากนี้ ผลการทดลองที่ได้จากการวัดค่าความเป็นศูนย์กลางหรือ Network Centrality ทั้งสี่กราฟที่แสดงอยู่ในส่วนของผลการทดลองที่ได้จากงานวิจัยในครั้งนี้ ทำให้นักทดสอบระบบสามารถเลือกกลุ่มของฟีเจอร์ที่จำเป็นต้องทำการทดสอบซ้ำเมื่อเกิดข้อบกพร่องขึ้น โดยพิจารณาจากกราฟที่บ่งชี้ถึงค่า Betweenness Centrality ซึ่งบอกถึงการพึ่งพากันของฟีเจอร์และความเป็นอิสระของฟีเจอร์นั้น ๆ ได้เป็นอย่างดี ทำให้สามารถลดทรัพยากร เวลา และค่าใช้จ่ายที่เกิดขึ้นในการพัฒนาโครงการซอฟแวร์ภายในองค์กรได้

สำหรับแนวทางการวิจัยในอนาคต เนื่องจากขั้นตอนในการเตรียมข้อมูลในการวิเคราะห์กราฟเครือข่าย ซึ่งเป็นวิธีที่ง่ายและใช้เวลารวบรวมข้อมูลไม่นานนัก จึงทำให้ การสร้างแบบจำลอง SNA ร่วมกับเทคนิคในการใช้ทฤษฎีกราฟสามารถเป็นแนวทางในการศึกษางานวิจัยในโดเมนอื่น ๆ ที่น่าสนใจต่อไปได้ รวมถึงเป็นต้นแบบในการวิเคราะห์ข้อมูลของข้อบกพร่องหรือข้อมูลที่น่าสนใจในโครงการต่อ ๆ ไปภายในองค์กรได้ เพื่อที่จะสกัดข้อมูลจำนวนมหาศาลที่ไม่สามารถดูและสรุปผลได้ด้วยตาเปล่าผ่านข้อมูลที่มีอยู่เดิมได้ โดยสามารถใช้เทคนิคนี้ในการนำข้อมูลมาแสดงผลในรูปแบบกราฟ SNA เพื่อเป็นแนวทางในการปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ทั้งภายในและภายนอกองค์กรให้ดียิ่งขึ้นในภายภาคหน้าได้

บรรณานุกรม

1. Wasserman, S. and K. Faust, *Social network analysis : methods and applications*. Structural analysis in the social sciences. 1994, Cambridge: Cambridge University Press.
2. Hanneman, R.A. and M. Riddle, *Introduction to social network methods*. 2005, University of California Riverside.
3. *Degree Centrality*. [cited 2022 05/01/2022]; Available from: https://cambridge-intelligence.com/social-network-analysis/.
4. *Closeness Centrality*. [cited 2022 05/01/2022]; Available from: https://cambridge-intelligence.com/social-network-analysis/.
5. Durland, M. and K. Fredericks, *New directions for evaluation: Social network analysis in program evaluation. 107*. 2005, San Francisco: Jossey-Bass.
6. *Betweenness Centrality*. [cited 2022 05/01/2022]; Available from: https://cambridge-intelligence.com/social-network-analysis/.
7. Bastian, M., S. Heymann, and M. Jacomy. *Gephi: an open source software for exploring and manipulating networks*. in *Proceedings of the international AAAI conference on web and social media*. 2009.
8. Sureka, A., A. Goyal, and A. Rastogi. *Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis*. in *Proceedings of the 4th india software engineering conference*. 2011.
9. Koochakzadeh, N. and R. Alhajj. *Social network analysis in software testing to categorize unit test cases based on coverage information*. in *2011 IEEE International Conference on High Performance Computing and Communications*. 2011. IEEE.
10. Maitrikul, C. and Y. Limpiyakorn. *GUI Test Case Prioritization using Social Network Analysis*. in *Journal of Physics: Conference Series*. 2020. IOP Publishing.
11. Vandermark, M.A., *Defect Escape Analysis: Test Process Improvement.* 2003.
12. Zimmermann, T. and N. Nagappan. *Predicting defects using network analysis on dependency graphs*. in *Proceedings of the 30th international conference on*

Software engineering. 2008.

13. Biçer, S., A.B. Bener, and B. Çağlayan. *Defect prediction using social network analysis on issue repositories*. in *Proceedings of the 2011 International Conference on Software and Systems Process*. 2011.

ประวัติผู้เขียน

ชื่อ-สกุล	พรรณธิภา บุญมาพบ
วัน เดือน ปี เกิด	23 สิงหาคม 2531
สถานที่เกิด	นครราชสีมา
วุฒิการศึกษา	มหาวิทยาลัยธรรมศาสตร์
ที่อยู่ปัจจุบัน	889/235 ไลฟ์บางกอก บูเลอวาร์ด รามอินทรา ถ. รามอินทรา แขวง คันนายาว เขต คันนายาว กรุงเทพฯ 10230
ผลงานตีพิมพ์	P Bunmapob, Y Limpiyakorn. (2022). Exploring Defect Data with Network Visualization. 2022 2nd IEEE International Conference on Software Engineering and Artificial Intelligence (SEAI), Virtual Conference.