แบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซเพื่อการวัดคุณภาพอย่างอัตโนมัติ

นายอัรกอม มะแดเฮาะ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

An Open-Source Software Quality Model for Automated Quality Measurement

Mr. Arkom Madaehoh

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 2022

หัวข้อวิทยานิพนธ์	แบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซเพื่อการวัดคุณภาพอย่างอัตโนมัติ
โดย	นายอัรกอม มะแดเฮาะ
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

..... กรรมการ
(รองศาสตราจารย์ ดร.ดวงดาว วิชาดากุล)

..... กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.เบญจพร ลิ้มธรรมาภรณ์)

อัรกอม มะแดเฮาะ : แบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซเพื่อการวัดคุณภาพอย่างอัตโนมัติ. (An Open-Source Software Quality Model for Automated Quality Measurement) อ.ที่ปรึกษาหลัก : รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

ปัจจุบันได้มีการนำเสนอแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซหลายแบบจำลองเพื่อใช้ในการประเมินคุณภาพของซอฟต์แวร์โอเพนซอร์ซ แต่แบบจำลองเหล่านั้นมีข้อจำกัดเนื่องจากเป็นการประเมินตามความคิดส่วนตัวซึ่งต้องอาศัยผู้ใช้ในการประเมิน และแบบจำลองดังกล่าวต้องการข้อมูลจากหลากหลายแหล่ง เพื่อเสริมการประเมินคุณภาพซอฟต์แวร์โอเพนซอร์ซที่เป็นอยู่ให้ทำได้สมบูรณ์มากยิ่งขึ้น วิทยานิพนธ์นี้จึงได้นำเสนอแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซใหม่ที่ชื่อว่าโอเอสเอส-เอคิวเอ็ม โดยมีเป้าหมายเพื่อการวัดคุณภาพซอฟต์แวร์โอเพนซอร์ซอย่างอัตโนมัติ แบบจำลองโอเอสเอส-เอคิวเอ็มได้นำเสนอตัววัดคุณภาพและเครื่องมืออัตโนมัติที่สามารถดึงข้อมูลเกี่ยวกับซอฟต์แวร์โอเพนซอร์ซจากกิตฮับ ซอร์ซโค้ด โซนาร์คิวบ์ และสแต็กเอกซ์เชนจ์ ทำให้สามารถกำหนดคะแนนคุณภาพของซอฟต์แวร์โอเพนซอร์ซได้ โอเอสเอส-เอคิวเอ็มได้รับการตรวจสอบจากวิศวกรซอฟต์แวร์ที่มีประสบการณ์ในการเลือกใช้ซอฟต์แวร์โอเพนซอร์ซ นอกจากนี้การจัดลำดับซอฟต์แวร์โอเพนซอร์ซโดยเครื่องมือโอเอสเอส-เอคิวเอ็มยังถูกนำไปเปรียบเทียบกับการจัดลำดับด้วยวิธีอื่น และพบว่าการจัดลำดับของโอเอสเอส-เอคิวเอ็มมีสหสัมพันธ์ระดับต่ำมากถึงปานกลางในทิศทางตรงกันข้ามกับวิธีจัดลำดับอื่น ๆ ตามความคิดเห็นและความนิยมของผู้ใช้ และมีสหสัมพันธ์ระดับปานกลางในทิศทางเดียวกันกับวิธีการจัดลำดับอื่นที่เน้นการตรวจสอบความมั่นคงที่ซอฟต์แวร์โดยตรง ทั้งนี้เนื่องจากโอเอสเอส-เอคิวเอ็มครอบคลุมปัจจัยคุณภาพหลายอย่างที่ไม่ได้ถูกพิจารณาโดยวิธีการจัดลำดับอื่น ๆ ดังกล่าว โอเอสเอส-เอคิวเอ็มจึงให้ข้อมูลด้านคุณภาพของซอฟต์แวร์โอเพนซอร์ซในเชิงลึกที่ดีกว่า

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2565 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6272103221 : MAJOR SOFTWARE ENGINEERING

KEYWORD: open-source quality model, open-source software, software quality measurement, software selection

Arkom Madaehoh : An Open-Source Software Quality Model for Automated Quality Measurement. Advisor: Assoc. Prof. TWITTIE SENIVONGSE, Ph.D.

At present, several open-source software quality models have been proposed to assess open-source software quality, but the assessment is rather limited because it is often subjective, relies on heavy user intervention, and requires information from different sources. To complement and enhance existing approaches to open-source quality assessment, this thesis proposes a new open-source software quality model called OSS-AQM that aims at automating the measurement of open-source software quality. The OSS-AQM provides a set of quality metrics and an automation tool that can retrieve information about the open-source software from GitHub, source code, SonarQube, and Stack Exchange, and quantitatively determine the overall quality of the open-source software. The OSS-AQM has been reviewed by software engineers who have experiences in open-source software selection. In addition, the ranking of open-source software measured by the OSS-AQM tool is compared with other ranking methods. The OSS-AQM ranking has very low to medium negative correlation with other methods of subjective popularity-based ranking and medium positive correlation with another method of objective security-based ranking. This is because OSS-AQM covers several quality factors that are not considered by those ranking methods and hence provides better insights into quality of open-source software.

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

Academic Year: 2022 Advisor's Signature .....

กิตติกรรมประกาศ

ข้าพเจ้าขอขอบพระคุณ รองศาสตราจารย์ ดร. ทวิตีย์ เสนีย์วงศ์ ณ อยุธยา ที่เป็นทั้งอาจารย์ผู้ให้ความรู้ และอาจารย์ที่ปรึกษาวิทยานิพนธ์ คอยสละเวลาให้ความรู้ คำแนะนำ และให้คำปรึกษาที่มีคุณค่าตลอดในการศึกษาวิจัยและการจัดทำวิทยานิพนธ์นี้ ทำให้ผลงานวิทยานิพนธ์นี้เป็นจริง รวมทั้งขอขอบคุณอาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ประสิทธิ์ประสาทวิชาความรู้ให้แก่ข้าพเจ้าตลอดระยะเวลาการศึกษา

ขอขอบคุณ รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ ประธานกรรมการสอบวิทยานิพนธ์ รองศาสตราจารย์ ดร.ดวงดาว วิชาดากุล และรองศาสตราจารย์ ดร. เบญจพร ลิ้มธรรมาภรณ์ กรรมการสอบวิทยานิพนธ์ ที่ได้กรุณาสละเวลาให้คำแนะนำต่าง ๆ ที่ทำให้วิทยานิพนธ์เล่มนี้สมบูรณ์มากยิ่งขึ้น

ขอขอบคุณเพื่อน ๆ พี่ ๆ น้อง ๆ ทุกคนรวมไปถึงครอบครัวของข้าพเจ้าโดยเฉพาะบิดาและมารดาของข้าพเจ้าที่คอยให้กำลังใจและให้การสนับสนุนในทุกด้านตลอดการศึกษาจนสามารถสำเร็จลุล่วงด้วยดี

ท้ายที่สุด ข้าพเจ้าหวังเป็นอย่างยิ่งว่าผลงานวิทยานิพนธ์เล่มนี้จะเป็นประโยชน์ต่อผู้ที่สนใจ หากมีข้อผิดพลาดประการใด ข้าพเจ้าขอน้อมรับไว้เพื่อนำไปปรับปรุงต่อไป ความดีและคุณประโยชน์ของวิทยานิพนธ์เล่มนี้ ข้าพเจ้าขอมอบให้แก่ผู้มีพระคุณทุกท่าน

อรกอม มะแดเฮาะ

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ซอฟต์แวร์โอเพนซอร์ซได้รับความนิยมอย่างมากนับตั้งแต่ปี ค.ศ. 1991 จนถึงปัจจุบัน และมีแนวโน้มความนิยมเพิ่มมากขึ้นอีกในอนาคต [1],[2],[3] โดยที่ซอฟต์แวร์โอเพนซอร์ซเกิดจากความร่วมมือของเหล่านักพัฒนาจากทั่วทุกมุมโลก ปล่อยสิทธิให้เข้าถึงและใช้งานได้ฟรี จากข้อมูลเว็บไซต์กิตฮับดอทคอม (Github.com) ซึ่งเป็นแพลตฟอร์มจัดเก็บซอร์ซโค้ดอันดับหนึ่งของโลกเปิดเผยว่ามีจำนวนนักพัฒนามากถึง 60 ล้านคนที่มีส่วนร่วมในการพัฒนาซอฟต์แวร์โอเพนซอร์ซถึง 100 ล้านโครงการในปัจจุบัน [4]

ความนิยมของซอฟต์แวร์โอเพนซอร์ซทำให้บริษัทผลิตซอฟต์แวร์เริ่มให้ความสนใจในการนำซอฟต์แวร์โอเพนซอร์ซมาใช้งานมากขึ้น อันด้วยเหตุผลหลักคือสามารถลดต้นทุน และสามารถย่นระยะเวลาพัฒนาได้อย่างมีนัยสำคัญ โดยที่ซอฟต์แวร์โอเพนซอร์ซสามารถนำไปใช้งานในหลากหลายรูปแบบ เช่น นำไปใช้เป็นส่วนประกอบหนึ่งของระบบใหญ่ เครื่องมือช่วยเหลือภายในองค์กร หรือเป็นองค์ประกอบหนึ่งของซอฟต์แวร์เชิงพาณิชย์ อย่างไรก็ตามการนำซอฟต์แวร์โอเพนซอร์ซไปใช้งานจริงยังเป็นที่ถกเถียงในเรื่องคุณภาพของซอฟต์แวร์ เนื่องจากซอฟต์แวร์โอเพนซอร์ซนั้นพัฒนาบนสิ่งแวดล้อมที่ไร้การควบคุม และเป็นอิสระ แตกต่างจากซอฟต์แวร์เชิงพาณิชย์ที่พัฒนาจากความต้องการของผู้ใช้งานเป็นหลักและอยู่ภายใต้สิ่งแวดล้อมที่ถูกควบคุมคุณภาพ [5] ด้วยเหตุนี้บริษัทต่าง ๆ ที่ต้องการนำซอฟต์แวร์โอเพนซอร์ซไปใช้งานจริง จึงจำเป็นต้องกำหนดปัจจัยและเกณฑ์เพื่อตรวจสอบคุณภาพของซอฟต์แวร์ก่อนนำไปใช้งาน ทั้งนี้ เพื่อให้ได้ซอฟต์แวร์ที่ตรงตามความต้องการและคุณภาพที่ยอมรับได้

องค์กรพัฒนาซอฟต์แวร์มักประสบกับความท้าท้ายแบบเดียวกันในการเลือกซอฟต์แวร์โอเพนซอร์ซที่ต้องการตอบโจทย์การใช้งานและธุรกิจขององค์กร โดยผู้จัดการโครงการ (Project Manager) มักกล่าวเสมอว่าการเลือกโอเพนซอร์ซมาใช้งานเป็นเรื่องที่ยากมาก เพราะมักต้องคำนึงถึงหลายปัจจัย โดยมักทิ้งท้ายคำถามว่า “**ทำอย่างไรจึงจะสามารถเปรียบเทียบโอเพนซอร์ซว่าโครงการใดดีกว่ากัน?**” ซึ่งผู้วิจัยนำมาเป็นโจทย์ปัญหาของวิทยานิพนธ์ฉบับนี้

หากแต่ปัจจัยเพื่อตรวจสอบคุณภาพ และวิธีการวัดคุณภาพโอเพนซอร์ซของแต่ละองค์กรนั้นแตกต่างกัน ในอดีตจึงมีผู้คิดค้นแบบจำลองคุณภาพอย่าง [3],[6],[7] ที่ใช้เพื่อวัดคุณภาพของซอฟต์แวร์โอเพนซอร์ซ โดยแบบจำลองเหล่านี้ได้รับการยอมรับและนำไปใช้ในวงกว้าง

จากงานวิจัย [5] ทำให้ทราบว่าแบบจำลองคุณภาพต่าง ๆ ในอดีต เช่น คิวเอสโอเอส (QSOS) [7], โอเพนบีอาร์อาร์ (OpenBRR) [3], โอเอสเอ็มเอ็ม (OSMM) [6] ใช้ระยะเวลานานในกระบวนการตรวจสอบคุณภาพซอฟต์แวร์โอเพนซอร์ซอันเนื่องจากกระบวนการรวบรวมข้อมูลที่จำเป็นนั้นใช้ระยะเวลานานเกินไป และการประเมินผลจำเป็นต้องมีผู้ประเมินที่มีความรู้และประสบการณ์เกี่ยวกับซอฟต์แวร์โอเพนซอร์ซ ด้วยเหตุนี้ในบางองค์กรที่มีบุคลากรและเวลาที่จำกัด เช่น บริษัทสตาร์ตอัพ (Startup) จึงหลีกเลี่ยงการตรวจสอบคุณภาพของซอฟต์แวร์โอเพนซอร์ซแบบละเอียด และใช้วิธีเลือกจากคำแนะนำของผู้เชี่ยวชาญหรือเพื่อนร่วมงานที่มีประสบการณ์ใช้งานแทน [8]

จากรายงานของ [1] ได้รวบรวมแบบจำลองที่ใช้คัดเลือกและวัดคุณภาพโอเพนซอร์ซ โดยรวบรวมงานวิจัยตั้งแต่ปี ค.ศ. 2004 ถึงปี ค.ศ. 2019 โดยมีแบบจำลองคุณภาพทั้งสิ้น 35 แบบ แต่มีเพียง 12 แบบเท่านั้นที่มีเครื่องมือสนับสนุนในการวัดคุณภาพของซอฟต์แวร์โอเพนซอร์ซ ซึ่งในปัจจุบัน (ปี ค.ศ. 2023) จากการตรวจสอบโดยละเอียดทำให้ทราบว่านอกจากเครื่องมือ deps.dev ก็ไม่มีเครื่องมือใดเลยที่สามารถเข้าใช้งานเพื่อนำไปวัดคุณภาพของซอฟต์แวร์โอเพนซอร์ซได้จริง (สามารถดูตารางเปรียบเทียบดังแสดงในตารางที่ 1.1)

งานวิจัยโอเอสเอสเพสโต (OSS PESTO) [4] ได้นำเสนอเครื่องมือที่ใช้วัดคุณภาพของโครงการโอเพนซอร์ซโดยดึงข้อมูลผ่านแอปพลิเคชันโปรแกรมมิงอินเทอร์เฟซ (Application Programming Interface) หรือ เอพีไอ (API) จากเว็บไซต์กิตฮับโดยผู้วิจัย [4] เลือกข้อมูลเพียงบางส่วนจากแบบจำลองคุณภาพโอเอสเอสพาล (OSSPAL) [5] และทำการดึงข้อมูลอย่างอัตโนมัติยกตัวอย่างข้อมูล เช่น อายุของโครงการ จำนวนของข้อบกพร่องที่เปิดอยู่ จำนวนกิตฮับสตาร์ (GitHub Star) และจากข้อมูลข้างต้นนำมาแสดงเป็นตารางเปรียบเทียบ ทำให้ผู้ประเมินสามารถเปรียบเทียบข้อมูลเองได้ แต่งานวิจัยนี้ทำการเลือกข้อมูลเพียงบางส่วนที่กิตฮับให้บริการเท่านั้น และไม่ได้วิเคราะห์ข้อมูลจากซอร์ซโค้ดของโครงการ ทำให้ไม่สามารถวัดคุณภาพตามแบบจำลองที่เลือกไว้อย่างสมบูรณ์

เครื่องมือ deps.dev เป็นเครื่องมือจากกูเกิล (Google) ที่พัฒนาโครงการโอเพนซอร์ซอินไซต์ (Open-Source Insights) เพื่อต้องการตรวจสอบความมั่นคงของโปรเจคที่เป็นไลบรารีต่าง ๆ เช่น npm, Go, PyPl, NuGet เป็นต้น ทั้งนี้เพื่อป้องกันช่องโหว่ความมั่นคงและป้องกันความเสี่ยงจากการนำไลบรารีไปใช้งานโดยทำการตรวจสอบอัตโนมัติในทุกอาทิตย์ เมื่อไลบรารีมีความเสี่ยงแล้วถูกนำไปใช้งานในระบบอาจจะทำให้ระบบมีความเสี่ยงตามไปด้วยเช่นกัน จากการตรวจสอบพบว่าตัววัดส่วนใหญ่เกี่ยวข้องกับช่องโหว่และความปลอดภัย โดยตัววัดที่น่าสนใจคือ การทดสอบด้วยวิธีฟัซซิ่ง

(Fuzzing), ไลเซนส์ (License), การบำรุงรักษา (Maintained), ช่องโหว่ความมั่นคง (Vulnerability) โดยเครื่องมือนี้มุ่งเน้นไปที่ซอร์ซโค้ดและตรวจสอบมาตรการความมั่นคงมากกว่าสนใจคุณภาพต่าง ๆ ของโอเพนซอร์ซทั้งหมด

จากงานวิจัยข้างต้นทำให้สรุปได้ว่า **ไม่มีแบบจำลองคุณภาพใดเลยที่มีเครื่องมือที่สามารถใช้งานได้ในปัจจุบัน และไม่มีเครื่องมือวัดคุณภาพใดเลยที่สามารถรวบรวมข้อมูลและวัดคุณภาพโดยรวมของโอเพนซอร์ซได้แบบอัตโนมัติอย่างครบถ้วนเพียงพอ** ดังนั้น งานวิจัยนี้มีเป้าหมายที่จะพัฒนาแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซใหม่สำหรับโอเพนซอร์ซที่อยู่บนเว็บไซต์กิตฮับ โดยการกำหนดปัจจัยและตัววัดที่ใช้คัดเลือกซอฟต์แวร์โอเพนซอร์ซซึ่งสามารถคำนวณได้อย่างอัตโนมัติ รวบรวมข้อมูลที่จำเป็น แปลงข้อมูลเป็นคะแนน เพื่อให้ได้คะแนนรวมที่สามารถวัดและเปรียบเทียบได้ โดยจุดมุ่งหมายทั้งหมดนี้เพื่อช่วยให้ผู้ประเมิน ผู้ตรวจสอบคุณภาพของซอฟต์แวร์โอเพนซอร์ซ และนักพัฒนาซอฟต์แวร์ สามารถนำค่าคะแนนคุณภาพเฉพาะส่วนที่วัดค่าได้อัตโนมัติไปเปรียบเทียบคุณภาพของโครงการโอเพนซอร์ซต่าง ๆ เพื่อที่จะเป็นข้อมูลส่วนหนึ่งที่ช่วยประกอบการตัดสินใจเลือกใช้โอเพนซอร์ซได้อย่างรวดเร็ว

ตารางที่ 1.1 รายละเอียดแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซที่สำรวจจากงานวิจัยต่าง ๆ [1],[2]

1.1 แบบจำลองคุณภาพ	ชื่อย่อ	ปรับปรุงจาก	ผู้สนับสนุน	ปีที่ตีพิมพ์	เปิดเผยแบบจำลอง	จำนวนปัจจัยหลัก/ปัจจัยย่อย	จำนวนตัววัด	วัดค่าอัตโนมัติ	วัดค่าด้วยผู้ประเมิน	นำเสนอเครื่องมือ	เครื่องมือใช้งานได้หรือไม่ ?
Capgemini Open Source maturity model [6]	OSMM	-	Cap Gemini	2003	/	4/12	12	/ (5)	/ (7)	-	-
Navicasoft Open source maturity model [9]	N-OSMM	-	Navicasoft	2004	/	6/0			/	/ (templates)	-
Qualification and Selection of Open Source software model [7]	QSOS	-	Atos Origin	2004	/	4/0	16	/ (4)	/ (12)	/ (O3S)	-
Open Business Readiness Rating [3]	OpenBRR	-	Intel, Carnegie Mellon, SpikeSource	2005	/	11/0	28	/ (13)	/ (15)	-	-
OpenBQR: A framework for the assessment of OSS [10]	OpenBQR	QSOS, OpenBRR	Università dell'Insubria	2007	/	4/9	14	/ (Not explicitly specified)	/ (Not explicitly specified)	/ (Open BQR)	-

ตารางที่ 1.1 รายละเอียดแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซที่สำรวจจากงานวิจัยต่าง ๆ [1],[2] (ต่อ)

แบบจำลองคุณภาพ	ชื่อย่อ	ปรับปรุงจาก	ผู้สนับสนุน	ปีที่ตีพิมพ์	เปิดเผยแบบจำลอง	จำนวนปัจจัยหลัก/ปัจจัยย่อย	จำนวนตัววัด	วัดค่าอัตโนมัติ	วัดค่าด้วยผู้ประเมิน	นำเสนอเครื่องมือ	เครื่องมือใช้งานได้หรือไม่ ?
Software Quality Observatory for Open Source Software [11]	SQO-OSS	-	Aristotle University, Athens University	2008	/	2/10	33	/ (33)	-	/ (Alitheia)	-
QualOSS Open Source Assessment Model [12]	QualOSS	ISO 9126	Fraunhofer Institute for Experimental Software Engineering	2009	/	2/11	14	-	/ (14)	-	-
QualiPSo Open Source Maturity Model [13]	QualiPSo OSS	QSOS, OpenBRR	Università degli Studi dell'Insubria	2009	/	3/11			/	/ (Checklist)	-
ISO/IEC 25010 System and software quality models [14]	ISO/IEC 25010:2011	-	ISO Standard	2011	/	2/13	40	/	/	-	-

ตารางที่ 1.1 รายละเอียดแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซที่สำรวจจากงานวิจัยต่าง ๆ [1],[2] (ต่อ)

แบบจำลองคุณภาพ	**ชื่อย่อ**	**ปรับปรุงจาก**	**ผู้สนับสนุน**	**ปีที่ตีพิมพ์**	**เปิดเผยแบบจำลอง**	**จำนวนปัจจัยหลัก/ปัจจัยย่อย**	**จำนวนตัววัด**	**วัดค่าอัตโนมัติ**	**วัดค่าด้วยผู้ประเมิน**	**นำเสนอเครื่องมือ**	**เครื่องมือใช้งานได้หรือไม่ ?**
OSSPAL [5]	OSSPAL	OpenBRR	Carnegie Mellon	2017	/	7/0	11	/ (Not explicitly specified)	/ (Not explicitly specified)	/ (OSSPal)	-
Open source software project evaluation and selection tool [4]	OSS PESTO	-	Tampere University	2021	-	-	4 factors from OSSPAL	/ (11)	-	/ (OSS-PESTO)	-
Open-source Insights [15]	-	OpenSSF Scorecard	Google	June, 2022	-	-	12	12	-	/ (deps.dev)	/
An Open-Source Software Quality Model for Automated Quality Measurement **(วิทยานิพนธ์ฉบับนี้)**	OSS-AQM	OpenBRR, OSS PESTO	Chulalongkorn University	2023	/	5/19	19 (Various Sources)	/ (19)	-	/	/

1.2 วัตถุประสงค์ของงานวิจัย

1) เพื่อนำเสนอแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซที่สามารถวัดคุณภาพและเปรียบเทียบโอเพนซอร์ซอย่างอัตโนมัติได้
2) เพื่อนำเสนอเครื่องมือสนับสนุนการวัดคุณภาพและเปรียบเทียบคุณภาพโอเพนซอร์ซตามแบบจำลองคุณภาพที่นำเสนอได้

1.3 ขอบเขตงานวิจัย

1) ใช้ข้อมูลโอเพนซอร์ซและซอร์ซโค้ดที่เปิดเป็นสาธารณะในเว็บไซต์กิตฮับเท่านั้น
2) โอเพนซอร์ซที่จะนำมาประเมินผลจะต้องมีฟังก์ชันการทำงานตรงตามที่ต้องการและมีอายุโครงการมากกว่า 6 เดือนขึ้นไป เนื่องจากข้อมูลอาจมีจำนวนน้อยเกินไปหากมีอายุโครงการน้อยกว่า 6 เดือน
3) ปัจจัยคุณภาพ (Quality Factors) และปัจจัยคุณภาพย่อย (Sub-quality Factors) ที่นำมาสร้างแบบจำลองคุณภาพนั้นพิจารณาจากการอ้างอิงมากกว่าร้อยละ 15 ขึ้นไปเป็นหลักตามงานวิจัย [1]
4) ตัววัดคุณภาพจะพิจารณาภายใต้ขอบเขตที่จะต้องหาค่ามาได้อัตโนมัติ
5) ตัววัดคุณภาพที่นำมาอ้างอิงทั้งหมดจะให้ความสำคัญกับตัววัดที่มีการอ้างอิงจากแบบจำลองคุณภาพ [2] ก่อน จากนั้นตัดตัววัดที่ไม่สามารถหาค่าได้อัตโนมัติ และถัดมาตัดตัววัดที่จำเพาะเจาะจงตามแบบจำลองคุณภาพ
6) เครื่องมือที่พัฒนาจะวัดค่าคุณภาพของโครงการโอเพนซอร์ซที่มีการนำเข้าสู่เครื่องมือ โดยยังไม่สามารถประมวลผลข้อมูลและแสดงผลลัพธ์แบบเรียลไทม์ แม้ข้อมูลโครงการต้นฉบับบนกิตฮับจะมีการเปลี่ยนแปลงไปแล้วก็ตาม
7) เครื่องมือที่พัฒนาสามารถบอกค่าคุณภาพได้ด้วยตัวเลข โดยค่าตัวเลขที่มากกว่าของโครงการหนึ่งจะถือว่าดีกว่าอีกโครงการหนึ่ง

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้แบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซสำหรับการประเมินคุณภาพโอเพนซอร์ซ แบบอัตโนมัติ

2) ได้เครื่องมือที่ใช้วัดคุณภาพโอเพนซอร์ซสำหรับวัดค่าคุณภาพที่สามารถคำนวณได้ อย่างอัตโนมัติ
3) ค่าคุณภาพที่วัดได้สามารถใช้เป็นข้อมูลประกอบในการตัดสินใจเลือกใช้โอเพนซอร์ซ ร่วมกับการประเมินคุณภาพตามปัจจัยและตัววัดอื่น ๆ ที่ไม่สามารถประเมินได้อย่างอัตโนมัติ

1.5 ขั้นตอนในการดำเนินงาน

1) ศึกษางานวิจัยที่เกี่ยวข้องรวมถึงทำความเข้าใจวิธีการต่าง ๆ และโอกาสในการต่อยอดของแต่ละงานวิจัย
2) ศึกษาวิธีการดึงข้อมูลจากแหล่งข้อมูลของโอเพนซอร์ซ และวิธีจัดเก็บข้อมูลที่จำเป็น
3) ออกแบบแบบจำลองซอฟต์แวร์โอเพนซอร์ซ และกำหนดตัววัดของโอเพนซอร์ซ
4) ออกแบบแบบสอบถาม ลงมือสำรวจ และประเมินผลแบบจำลองซอฟต์แวร์โอเพนซอร์ซ
5) ออกแบบ และพัฒนาเครื่องมือวัดคุณภาพและเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ
6) ทำการทดลองใช้งานเครื่องมือโดยผู้เชี่ยวชาญด้านซอฟต์แวร์โอเพนซอร์ซ
7) ประเมินผลการวิจัย และภัยคุกคามจากปัจจัยที่เกี่ยวข้อง
8) จัดทำรายงานวิทยานิพนธ์ และบทความวิชาการ

1.6 โครงสร้างของเนื้อหาในวิทยานิพนธ์

เนื้อหาของวิทยานิพนธ์เล่มนี้จะแบ่งออกเป็น 7 บท คือ บทที่ 1 เป็นบทที่กล่าวถึงที่มาและความสำคัญ แรงจูงใจ ของโจทย์ปัญหา วัตถุประสงค์ของงานวิจัย และขอบเขตของงานวิจัย บทที่ 2 กล่าวถึงทฤษฎี และงานวิจัยที่เกี่ยวข้อง บทที่ 3 นำเสนอแนวคิด และขั้นตอนการสร้างแบบจำลองคุณภาพ และวิธีการประเมิน บทที่ 4 กล่าวถึงผลการประเมินแบบจำลองคุณภาพที่นำเสนอ และแนวทางการปรับปรุง บทที่ 5 อธิบายความต้องการ รายละเอียดของการพัฒนาเครื่องมือในการประเมินคุณภาพโอเพนซอร์ซ บทที่ 6 จะกล่าวถึงการเปรียบเทียบผลงานวิจัยในการจัดลำดับโอเพนซอร์ซกับการจัดลำดับด้วยวิธีอื่น และบทที่ 7 จะกล่าวถึงบทสรุปรวมถึงข้อจำกัด และแนวทางการปรับปรุงในอนาคต

1.7 บทความที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของงานวิจัยของวิทยานิพนธ์เล่มนี้ได้รับการคัดเลือกตีพิมพ์เป็นบทความวิชาการชื่อ “OSS AQM: An Open-Source Software Quality Model for Automated Quality Measurement” ในการประชุมวิชาการระดับนานาชาติ “8th International conference on data and software engineering (ICoDSE) 2022” ซึ่งจัดงาน ณ เมือง เดนปาซาร์ จังหวัด บาหลี ประเทศอินโดนีเซีย ระหว่างวันที่ 2-3 พฤศจิกายน พ.ศ. 2565

บทที่ 2
ทฤษฎีและงานที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ซอฟต์แวร์โอเพนซอร์ซ (Open Source Software)

ซอฟต์แวร์โอเพนซอร์ซ หรือเรียกแบบย่อว่า โอเพนซอร์ซ (Open Source) หรือ โอเอสเอส (OSS) [16] เติบโตจากกลุ่มนักพัฒนาซอฟต์แวร์อิสระที่ต้องการพัฒนาซอฟต์แวร์ที่สามารถเปิดเผยซอร์ซโค้ด (Source Code) ของโปรแกรมได้ โดยอยู่บนพื้นฐานความตั้งใจที่จะพัฒนาซอฟต์แวร์เพื่อสังคมอย่างยั่งยืน และแบ่งปันความรู้ในกลุ่มของนักพัฒนา ทำให้เกิดนวัตกรรมใหม่ ๆ ของการพัฒนาซอฟต์แวร์ โดยในปัจจุบันความร่วมมือดังกล่าวได้ปรากฏเวเป็นรูปเป็นร่างจริงแล้วในภาคปฏิบัติ เช่น โครงการอะแพชี (Apache Project) หรือโครงการไพทอน (Python Project) เป็นต้น โดยโครงการโอเพนซอร์ซเหล่านี้เกิดจากกลุ่มของนักพัฒนาที่มีความสนใจและได้รวมกลุ่มกัน จนสามารถสร้างผลผลิตแห่งภูมิปัญญา และก่อให้เกิดซอฟต์แวร์ใหม่ ๆ ที่ใช้งานได้จริงจำนวนมากมาย และเปิดสิทธิ์ให้สามารถเข้าถึง และผู้ใช้ทั่วไปสามารถนำมาใช้ประโยชน์ได้ฟรี โดยไม่มีลิขสิทธิ์ผูกมัดใด ๆ โดยกลุ่มนักพัฒนาซอฟต์แวร์โอเพนซอร์ซยังคงเดินหน้าพัฒนาต่อไปอย่างต่อเนื่องในปัจจุบัน

2.1.2 กิตฮับ (GitHub)

กิตฮับ (GitHub) คือเว็บไซต์กิตฮับดอทคอม (Gitlab.com) ที่ให้บริการจัดเก็บ ซอร์ซโค้ดบนอินเทอร์เน็ต [17] และผู้ใช้สามารถปรับปรุงแก้ไขซอร์ซโค้ดและควบคุมเวอร์ชัน (Version Control) ที่จัดเก็บได้ โดยผู้ใช้ทั่วไปสามารถสมัครใช้งานผ่านหน้าเว็บไซต์ได้ฟรี เป็นพื้นที่ให้นักพัฒนาสามารถช่วยกันแก้ไขปรับปรุงซอร์ซโค้ดร่วมกัน ซึ่งโดยทั่วไปซอร์ซโค้ดเหล่านี้จะสามารถเข้าถึงได้แบบสาธารณะ แต่ปัจจุบันกิตฮับมีทางเลือกในการจัดเก็บ ซอร์ซโค้ดแบบส่วนตัวเพื่อปกปิดซอร์ซโค้ดไม่ให้ถูกแจกจ่ายออกไป ในปัจจุบันเว็บไซต์กิตฮับมีผู้ใช้งานมากกว่า 60 ล้านคนทั่วโลก และมีมากกว่า 100 ล้านโครงการบนระบบ

กิตฮับสตาร์ (GitHub Star) [18] เป็นคะแนนจากผู้ใช้งานในระบบที่โหวตให้กับโครงการที่สนใจ โดยผู้ใช้งานสามารถให้คะแนนได้ครั้งเดียว หรือ ไม่ให้ก็ได้ โดยกิตฮับสตาร์จะอยู่ในหน้าหลักของโครงการที่เปิดซอร์ซโค้ดเป็นสาธารณะ ซึ่งในปัจจุบันกิตฮับสตาร์สามารถนำมาเป็นส่วนหนึ่งของการตัดสินใจเลือกโครงการโอเพนซอร์ซ หรือเปรียบเทียบในแง่มุมจากความชอบของผู้ใช้งานจำนวนมากได้

2.1.3 โซนาร์คิวบ์ (SonarQube)

โซนาร์คิวบ์ เป็นเครื่องมือช่วยตรวจสอบคุณภาพของซอร์ซโค้ด และสามารถแนะนำข้อบกพร่อง (Defect) จุดบกพร่อง (Bug) การซ้ำของโค้ด (Code Duplication) ร่องรอยที่ไม่ดีของโค้ด (Code Smell) การทดสอบความครอบคลุมของเงื่อนไข (Code Coverage) และอีกหลากหลายฟังก์ชันงาน ทั้งนี้เพื่อให้นักพัฒนาสามารถแก้ไขปรับปรุงคุณภาพ ซอร์ซโค้ดให้ดียิ่งขึ้น ต้นกำเนิดของเครื่องมือโซนาร์คิวบ์เป็นโครงการซอฟต์แวร์โอเพนซอร์ซหนึ่งที่รองรับภาษาคอมพิวเตอร์ไม่กี่ภาษาเท่านั้น ต่อมาในปัจจุบันโซนาร์คิวบ์ได้รับการพัฒนาให้รองรับภาษาคอมพิวเตอร์มากถึง 27 ภาษาด้วยกัน [19] และอีกทั้งยังสามารถทำงานร่วมกับซอร์ซโค้ดที่จัดเก็บในกิตฮับได้ด้วยเช่นกัน โดยปัจจุบันสามารถรองรับภาษาตามเวอร์ชันของโซนาร์คิวบ์ที่ติดตั้ง โดยอ้างอิงจากรูปที่ 2.1 ซึ่งเวอร์ชัน Community Edition สามารถติดตั้งใช้งานได้โดยไม่มีค่าใช้จ่าย นอกจากนี้โซนาร์คิวบ์ได้นำเสนอเมตริกที่ได้จากการวัดเพื่อให้นักพัฒนาสามารถตรวจสอบคุณภาพของโค้ดผ่านเมตริกอีกด้วย ซึ่งมีเมตริกการวัดที่น่าสนใจ ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 ตัวอย่างเมตริกการวัดที่น่าสนใจ

ชื่อเมตริก	คำอธิบาย
Complexity	It is the Cyclomatic Complexity calculated based on the number of paths through the code. Whenever the control flow of a function splits, the complexity counter gets incremented by one. Each function has a minimum complexity of 1. This calculation varies slightly by language because keywords and functionalities do.
Maintainability Rating	(Formerly the SQALE rating.) Rating given to your project related to the value of your Technical Debt Ratio. The default Maintainability Rating grid is: A=0-0.05, B=0.06-0.1, C=0.11-0.20, D=0.21-0.5, E=0.51-1
Comments	Density of comment lines = Comment lines / (Lines of code + Comment lines) * 100 With such a formula: • 50% means that the number of lines of code equals the number of comment lines • 100% means that the file only contains comment lines

Language	Community Edition	Developer Edition	Enterprise Edition and Data Center Edition
ABAP		✓	✓
Apex			✓
C#	✓	✓	✓
C		✓	✓
C++		✓	✓
COBOL			✓
CSS	✓	✓	✓
Flex	✓	✓	✓
Go	✓	✓	✓
Java	✓	✓	✓
JavaScript	✓	✓	✓
Kotlin	✓	✓	✓
Objective-C		✓	✓
PHP	✓	✓	✓
PLI			✓
PLSQL		✓	✓
Python	✓	✓	✓
RPG			✓
Ruby	✓	✓	✓
Scala	✓	✓	✓
Swift		✓	✓
TypeScript	✓	✓	✓
TSQL		✓	✓
VB.NET	✓	✓	✓
VB6			✓
HTML	✓	✓	✓
XML	✓	✓	✓

รูปที่ 2.1 ภาษาที่รองรับตามเวอร์ชันของโซนาร์คิวบ์

2.1.4 สแต็กเอกซ์เชนจ์ (Stack Exchange)

สแต็กเอกซ์เชนจ์ เป็นระบบให้บริการถามและตอบปัญหาผ่านเว็บไซต์ โดยระบบสามารถให้ผู้ใช้งานตั้งคำถามขึ้นมาได้หลากหลายแนว [20] เช่น คำถามเชิงการเขียนโปรแกรม การทำอาหาร เกม ปัญหาทางคณิตศาสตร์ ภาษา และอื่น ๆ จากนั้นจะมีผู้รู้หรือผู้เชี่ยวชาญมาตอบคำถามนั้น เมื่อคำตอบนั้นเป็นคำตอบที่มีประโยชน์และสามารถแก้ไขปัญหาได้จริง คำตอบดังกล่าวจะได้รับคะแนนโหวตจากผู้ใช้งาน โดยทั่วไปปัญหาถามตอบต่าง ๆ จะเปิดเผยต่อสาธารณะ แต่หากต้องการตั้งคำถามหรือตอบคำถามใด ๆ จำเป็นจะต้องสมัครสมาชิกก่อน โดยตัวอย่างกลุ่มบริการของสแต็กเอกซ์เชนจ์แสดงในตารางที่ 2.2

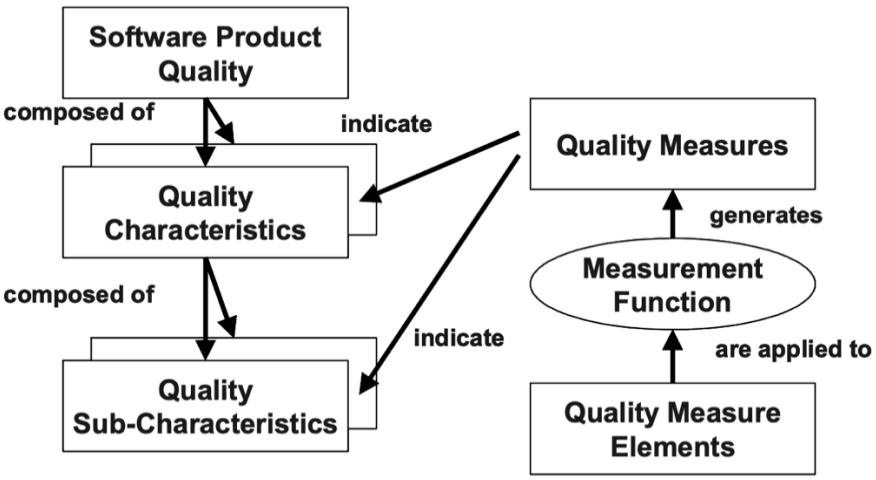
ตารางที่ 2.2 ตัวอย่างกลุ่มบริการของสแต็กเอกซ์เชนจ์

แหล่งบริการ	ประเภทถามตอบปัญหา
Stack OverFlow	ถามตอบปัญหาการเขียนโปรแกรม ซึ่งเป็นแหล่งชุมชนที่ใหญ่ที่สุดของสแต็กเอกซ์เชนจ์
English Stack Exchange	ถามตอบปัญหาเกี่ยวกับภาษาศาสตร์
MathOverFlow	ถามตอบปัญหาทางคณิตศาสตร์
PhysicsOverflow	ถามตอบปัญหาทางฟิสิกส์
Ask Ubuntu	ถามตอบปัญหาการใช้งานระบบปฏิบัติการอูบันตู

2.1.5 แบบจำลองคุณภาพซอฟต์แวร์ (Software Quality Models)

แบบจำลองคุณภาพซอฟต์แวร์ หรือแบบจำลองคุณภาพ ถูกนำเสนอขึ้นเพื่อแสดงความเกี่ยวข้องและเชื่อมโยงกันของคุณลักษณะเชิงคุณภาพกับมุมมองที่สนใจของซอฟต์แวร์ โดยส่วนใหญ่แบบจำลองคุณภาพมักนำเสนอในลักษณะแบบจำลองลำดับชั้น ซึ่งปัจจุบันมีงานวิจัยและมาตรฐานที่นำเสนอแบบจำลองคุณภาพจำนวนหนึ่ง ยกตัวอย่างเช่น แบบจำลองคุณภาพตามมาตรฐานไอเอสโอ 25010 (ISO 25010) [14] ที่ปรับปรุงมาจากมาตฐานไอเอสโอ 9126 (ISO 9126)

จากรูปที่ 2.2 แสดงให้เห็นถึงโครงสร้างวิธีการวัดคุณภาพของซอฟต์แวร์ตามแบบจำลองคุณภาพไอเอสโอ 25010 โดยเริ่มจากคุณภาพผลิตภัณฑ์ซอฟต์แวร์ (Software Product Quality) ที่สนใจ ประกอบด้วยคุณลักษณะเชิงคุณภาพ (Quality Characteristics) หลายคุณลักษณะประกอบกัน ซึ่งในแต่ละคุณลักษณะเชิงคุณภาพจะประกอบด้วย คุณลักษณะย่อยเชิงคุณภาพ (Quality Sub Characteristics) หลายคุณลักษณะ โดยตัววัดคุณภาพใด ๆ (Quality Measures) สามารถสะท้อนคุณลักษณะเชิงคุณภาพ หรือ คุณลักษณะเชิงคุณภาพย่อยได้ โดยที่การวัดเชิงคุณภาพจะต้องกำหนดฟังก์ชันในการวัด (Measurement Function) ให้ชัดเจนว่าประกอบด้วยองค์ประกอบของตัววัดคุณภาพ (Quality Measure Elements) ใดบ้าง



รูปที่ 2.2 แผนภาพแบบจำลองการวัดคุณภาพซอฟต์แวร์ตามมาตฐานไอเอสโอ 25010 [14]

แบบจำลองคุณภาพอย่างโอเพนบีอาร์อาร์ (OpenBRR) [3] มีลักษณะโครงสร้างที่คล้ายกันกับไอเอสโอ 25010 [14] ตรงที่มีโครงสร้างลำดับชั้น และมีตัววัดคุณภาพ แต่จะมีชื่อเรียกของแต่ละหน่วยแตกต่างกัน โดยแบบจำลองคุณภาพโอเพนบีอาร์อาร์ [3] ให้ความสนใจกับซอฟต์แวร์โอเพนซอร์ซ เป็นหลัก ดังนั้นจึงมีคุณลักษณะเชิงคุณภาพที่ต้องคำนึงแตกต่างออกไป เช่น แหล่งชุมชนโอเพนซอร์ซ (Open source community) และการสนับสนุนช่วยเหลือ (Support) ในขณะที่ไอเอสโอ 25010 [14] เน้นการวัดที่คุณภาพของผลิตภัณฑ์ซอฟต์แวร์เป็นหลัก

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 งานวิจัย Open Source Software Evaluation, Selection, and Adoption: a Systematic Literature Review

งานวิจัยนี้ [1] นำเสนอปัจจัยต่าง ๆ ที่ส่งผลต่อการนำโอเพนซอร์ซมาใช้งานมากที่สุด โดยงานวิจัยนี้ใช้วิธีการทบทวนวรรณกรรมอย่างเป็นระบบ (Systermetic Literature Review) จากรายการงานวิจัยที่มีการตีพิมพ์ในระหว่างปี ค.ศ. 1998-2019 ซึ่งเป้าหมายหลักของงานวิจัยนี้ เพื่อต้องการรู้ว่างานวิจัยแขนงนี้ (การนำโอเพนซอร์ซมาใช้งาน) ยังได้รับความสนใจอยู่อีกหรือไม่ และปัจจัยใดบ้างที่ส่งผลต่อการนำโอเพนซอร์ซมาใช้งานในมุมมองที่ผู้ใช้งานสนใจ โดยงานวิจัยนี้ได้คัดเลือกงานวิจัยประเภทแบบจำลองคุณภาพจำนวน 35 แบบ, 20 งานวิจัยประเภทสำรวจปัจจัยในการนำโอเพนซอร์ซมาใช้งาน และ 5 งานวิจัยประเภทถอดบทเรียนจากโครงการ ซึ่งผลลัพธ์ของงานวิจัยนี้ ประกอบไปด้วย 35 แบบจำลองคุณภาพที่ได้รับการอ้างอิงจากงานวิจัยอื่น ๆ, 12 เครื่องมือ

สนับสนุนในการวัดคุณภาพโอเพนซอร์ซ และ 8 ปัจจัยคุณภาพหลักและ 47 ปัจจัยคุณภาพย่อยที่ส่งผลต่อการประเมินผลและคัดเลือกโอเพนซอร์ซไปใช้งานมากที่สุด

ผู้วิจัยเล็งเห็นว่าปัจจัยคุณภาพต่าง ๆ ที่ระบุในงานวิจัย [1] มีความน่าสนใจเพราะเป็นปัจจัยที่ได้รับการรวบรวมมาจากงานตีพิมพ์ตลอดระยะเวลาเกือบ 21 ปี ผู้วิจัยจึงนำปัจจัยนี้มาเป็นข้อมูลนำเข้าของงานวิจัยนี้อีกด้วย

2.2.2 งานวิจัย Exploring information from OSS repositories and platforms to support OSS selection decisions

งานวิจัยนี้ [2] นำเสนอช่องทางการเก็บข้อมูลเพื่อนำไปเป็นข้อมูลให้กับแบบจำลองคุณภาพต่าง ๆ ที่ใช้สำหรับคัดเลือกโอเพนซอร์ซ โดยได้ระบุปัญหาว่ากระบวนการเก็บข้อมูลเพื่อใช้ในการคัดเลือกโอเพนซอร์ซนั้นใช้เวลานาน และหากได้ข้อมูลที่จำเป็นไม่เพียงพอจะทำให้การตัดสินใจเลือกโอเพนซอร์ซผิดพลาดได้ โดยงานวิจัยนี้มีวัตถุประสงค์เพื่อเติมช่องว่างระหว่างแบบจำลองคุณภาพและแหล่งข้อมูลที่มีอยู่ในปัจจุบัน โดยงานวิจัยนี้ได้เลือกแบบจำลองคุณภาพที่ได้รับการยอมรับ ได้แก่ โอเอสเอสพาล (OSSPAL) [5], โอเพนบีอาร์อาร์ (OpenBRR) [3], คิวเอสโอเอส (QSOS) [7], โอเพนบีคิวอาร์ (OpenBQR) [10], โอเอสเอ็มเอ็ม (OSMM) [6], เอสคิวโอโอเอสเอส (SQO_OSS) [11] และควอลโอเอสเอส (Qual_OSS) [12] ใช้ประกอบว่าข้อมูลใดบ้างที่จำเป็นต้องจัดหาเพื่อนำมาใช้ในแบบจำลองคุณภาพที่กล่าวในข้างต้น และงานวิจัยนี้ได้เสนอแหล่งข้อมูล 3 แห่ลง คือ กิตฮับ, โซนาร์คลาวด์ (โซนาร์คิวบ์ที่ใช้งานบนคลาวด์) และ สแต็กเอกซ์เชนจ์ โดยข้อมูลที่ได้จะต้องสามารถดึงมาได้อัตโนมัติผ่านเอพีไอ เพื่อลดระยะเวลาในกระบวนการเก็บรวบรวมข้อมูลที่จะใช้คัดเลือกโอเพนซอร์ซให้มากที่สุด

ผลลัพธ์ของงานวิจัยนี้ได้แสดงตารางเปรียบเทียบว่าแบบจำลองคุณภาพใดบ้างที่ต้องการข้อมูลโดยแบ่งเป็น 7 ด้านใหญ่ ๆ ได้แก่ ด้านแหล่งชุมชนและการสนับสนุนช่วยเหลือ (Community and Support), ด้านเศรษฐกิ (Economic), ด้านเอกสารต่าง ๆ (Documentation), ด้านลิขสิทธิ์ (License), ด้านวุฒิภาวะ (Maturity), ด้านคุณภาพ (Quality) และด้านอื่น ๆ (Others) โดยระบุว่าข้อมูลจำเป็นเหล่านี้สามารถดึงมาจากกิตฮับ โซนาร์คลาวด์ และสแต็กเอกซ์เชนจ์ ได้หรือไม่ โดยผู้วิจัยเล็งเห็นว่างานวิจัยนี้ควรมีการพัฒนาต่อยอด ดังนี้

1. งานวิจัยระบุข้อมูลที่มีจากทั้งสามแหล่ง คือ กิตฮับ โซนาร์คลาวด์ และ สแต็กเอกซ์เชนจ์ แต่ไม่ได้เจาะจงถึงตัววัดคุณภาพที่จับต้องได้ และไม่ได้กล่าวถึงวิธีการดึงข้อมูลจากแหล่งดังกล่าวข้างต้น

2. งานวิจัยไม่ได้พัฒนาเครื่องมือใด ๆ ที่ใช้สำหรับจัดเก็บข้อมูลตามที่แบบจำลองคุณภาพดังกล่าวข้างต้นต้องการ
3. ข้อมูลบางค่าจากแบบจำลองคุณภาพข้างต้นไม่สามารถหาค่าวัดได้จาก กิตฮับ โซนาร์คลาวด์ และสแต็กเอกซ์เชนจ์ จำเป็นต้องหาแหล่งข้อมูลอื่นเพิ่มเติม

จากข้อควรปรับปรุงดังกล่าว ผู้วิจัยเล็งเห็นประโยชน์ของการนำข้อมูลจากแหล่งข้อมูลกิตฮับ โซนาร์คลาวด์ และสแต็กเอกซ์เชนจ์มาเป็นข้อมูลนำเข้าของงานวิจัย และสามารถคิดค้นตัววัดใหม่ที่ยังขาดรวมถึงพัฒนาเครื่องมือที่ง่ายต่อการจัดเก็บข้อมูล ประมวลผล เพื่อแสดงผลข้อมูลเปรียบเทียบที่ให้ผู้ประเมินสามารถตัดสินใจเลือกโอเพนซอร์ซได้รวดเร็ว

2.2.3 งานวิจัย OpenBRR: The Business readiness rating model

งานวิจัยนี้ [3] นำเสนอแบบจำลองคุณภาพสำหรับการวัดคุณภาพและคัดเลือกโอเพนซอร์ซ โดยได้รับความร่วมมือจากมหาวิทยาลัยคาร์เนกีเมลลอน อินเทล และสไปก์ซอร์ซ ซึ่งจุดประสงค์หลักของแบบจำลองคุณภาพนี้เพื่อให้สามารถวัดคุณภาพและตัดสินใจเลือกโอเพนซอร์ซที่สนใจได้อย่างรวดเร็ว และผลลัพธ์การประเมินจะสามารถแชร์ให้กับคนอื่น ๆ ที่อยู่ในแหล่งชุมชนโอเพนซอร์ซได้เพื่อใช้อ้างอิงเป็นมาตรฐานเดียวกัน โดยงานวิจัยนี้ได้แบ่งขั้นตอนในการประเมินออกเป็น 4 เฟส ได้แก่ การประเมินแบบตัดตัวเลือกแบบรวดเร็ว (Quick Assessment Filter) การประเมินตามการใช้งานจริง (Target Usage Assessment) การเก็บข้อมูลและประมวลผล (Data Collection and Processing) และ การแปลข้อมูล (Data Translation) ซึ่งผลลัพธ์สุดท้ายได้ค่าคะแนนบีอาร์อาร์เรทติงสกอร์ (BRR Rating Score) โดยใช้วิธีการหาค่าเฉลี่ยถ่วงน้ำหนัก (Weighted Average Method)

งานวิจัยนี้ได้รับการอ้างอิงถึงมากที่สุดในงานวิจัยที่เกี่ยวข้อง เพราะมีความน่าเชื่อถือเป็นที่ยอมรับ [5] อธิบายขั้นตอนที่ละเอียด และสามารถตรวจสอบย้อนกลับถึงที่มาของคะแนนได้ แต่อย่างไรก็ตามการดำเนินการแต่ละขั้นตอนจะต้องมีข้อมูลนำเข้าจำนวนมากมาเกี่ยวข้องเพื่อสนับสนุนการตัดสินใจ โดยผู้วิจัยประมาณการว่าจะต้องมีผู้ประเมินอย่างน้อย 4-5 คนที่ต้องใช้เวลาในการรวบรวมข้อมูล และดำเนินการประเมินผล ดังนั้นเวลาส่วนใหญ่จึงตกไปอยู่ในกระบวนการเก็บข้อมูลเป็นหลัก

จากงานวิจัยนี้ [3] ผู้วิจัยได้นำวิธีการแปลงจากค่าช่วงเป็นคะแนนตัวเลข 1-5 ตามเกณฑ์ที่กำหนดไว้ และได้นำโครงสร้างแบบจำลองคุณภาพเป็นต้นแบบเพื่อใช้อ้างอิงในการพัฒนาแบบจำลองคุณภาพซอฟต์แวร์ตัวใหม่

2.2.4 งานวิจัย OSSPAL: Finding and Evaluating Open Source Software

งานวิจัยนี้ [5] เป็นงานวิจัยที่ต่อยอดมาจากงานวิจัยโอเพนบีอาร์อาร์ [3] โดยงานวิจัยนี้ได้ให้ความเห็นว่าแบบจำลองคุณภาพโอเพนบีอาร์อาร์ เน้นการวิเคราะห์ไปที่ตัวเลขเป็นหลัก แต่ในอีกแง่มุมหนึ่ง การเลือกโอเพนซอร์ซที่มาจากคำแนะนำจากผู้ใช้งานจริงมาแล้วก็มีส่วนสำคัญไม่แพ้กัน โดยผู้วิจัย [5] ได้ระบุจุดอ่อนของแบบจำลองคุณภาพ โอเพนบีอาร์อาร์หลายประการ ประการแรกคือการประเมินโครงการใด ๆ โดยแบบจำลองคุณภาพโอเพนบีอาร์อาร์สามารถคาดเดาได้ง่าย โดยสามารถคาดการจากการเคลื่อนไหวของโครงการ ความพร้อมใช้งานของเอกสาร และอัตราการสนับสนุนของโครงการนั้น ๆ ประการถัดไปคือ คะแนนโอเพนบีอาร์อาร์ที่ได้มาจากการประเมินไม่สามารถสะท้อนถึงความสามารถในการใช้งานจริงได้ในเชิงลึก ดังนั้นปัญหาจากการใช้งานต้องมาจากผู้ที่เคยใช้งานโอเพนซอร์ซเท่านั้นที่จะสามารถระบุปัญหาได้ และประการถัดมา การประเมินด้วยแบบจำลองดังกล่าวจะต้องคัดเลือกโอเพนซอร์ซมาจำนวนหนึ่งแล้ว ด้วยเหตุนี้อาจเกิดปัญหากับบริษัทที่ไม่มีผู้เชี่ยวชาญด้านโอเพนซอร์ซที่สามารถหาโครงการโอเพนซอร์ซเพื่อเข้าสู่กระบวนการประเมินได้อย่างเหมาะสม และเหตุผลสุดท้าย ถึงแม้การประเมินด้วยโอเพนบีอาร์อาร์จะสามารถกรองจำนวนโอเพนซอร์ซได้จำนวนมาก จนเหลือ 1-2 โครงการสุดท้าย แต่ท้ายที่สุดแล้วผู้ประเมินก็มักต้องการคำแนะนำจากเพื่อนนักพัฒนา หรือผู้เชี่ยวชาญที่มีประสบการณ์ใช้งานโอเพนซอร์ซมาก่อน

นอกจากนี้ผู้วิจัย [3] ได้พัฒนาแบบจำลองคุณภาพที่ชื่อว่า โอเอสเอสพาล (OSSPAL) โดยให้ความสำคัญกับคำแนะนำจากผู้ใช้งานจริงมาก่อน ซึ่งผู้วิจัย [3] ได้สร้างระบบที่ทำงานบนเว็บไซต์โดยผู้ใช้สามารถทำการรีวิว 1-5 คะแนนของแต่ละโครงการได้ ซึ่งรายชื่อโครงการมาจาก International Data Corporation (IDC) ที่ได้เปิดเผยรายชื่อใน ทุก ๆ ปี โดยผู้วิจัยมองว่าคำแนะนำที่มาจากผู้ใช้งานจริงนั้นมีประโยชน์และสามารถให้ผู้ประเมินได้รับรู้ในแง่ของการใช้งานและปัญหาที่เกิดขึ้นจริง แต่จากการไปสำรวจเว็บไซต์ของโอเอสเอสพาล ณ ปัจจุบัน (OSSpal.org) ไม่สามารถทำการสมัครสมาชิกและรีวิวได้อีกต่อไป อีกทั้งปัญหาอีกประการหนึ่งของโอเอเอสพาลคือผู้ให้ข้อมูลจะต้องเป็นผู้มีประสบการณ์ตรงและสมัครใจที่จะให้ความคิดเห็นและข้อเสนอแนะซึ่งหาได้ยาก

ด้วยเหตุผลดังกล่าวข้างต้น ผู้วิจัยมีความคิดที่จะสร้างตัววัดใหม่ที่สะท้อนปัญหาจริงจากผู้ใช้งานในปัจจุบันโดยสำรวจข้อมูลมาจากแหล่งชุมชนใหญ่ ๆ อย่างสแต็กโอเวอร์โฟลว์ (Stack Overflow) ที่ข้อมูลเหล่านี้เป็นปัญหาถามตอบจากผู้ใช้งานจริง

2.2.5 งานวิจัย Capgemini Expert Letter Open Source Maturity Model (OSMM)

งานวิจัยนี้ [6] นำเสนอแบบจำลองคุณภาพ และเผยแพร่ออกมาในปี ค.ศ. 2003 ถือเป็นแบบจำลองคัดเลือกและวัดคุณภาพโอเพนซอร์ซแรก ๆ ที่ให้ความสนใจโครงการโอเพนซอร์ซอย่างจริงจัง โดยแบบจำลองนี้ [6] ออกแบบตามโครงสร้างลำดับชั้น และแยกการประเมินออกเป็น 2 ส่วน โดยประกอบไปด้วยตัววัดเชิงประยุกต์ใช้งาน (Application Indicator) และตัววัดเชิงผลิตภัณฑ์ (Product Indicator) โดยที่ตัววัดเชิงประยุกต์ใช้งานจะประเมินจากผู้ใช้งาน และตัววัดผลิตภัณฑ์จะประเมินจากซอร์ซโค้ดของซอฟต์แวร์ โดยในงานวิจัยนี้ได้ระบุปัจจัยมากถึง 27 ปัจจัยในการนำไปวัดคุณภาพของโอเพนซอร์ซ

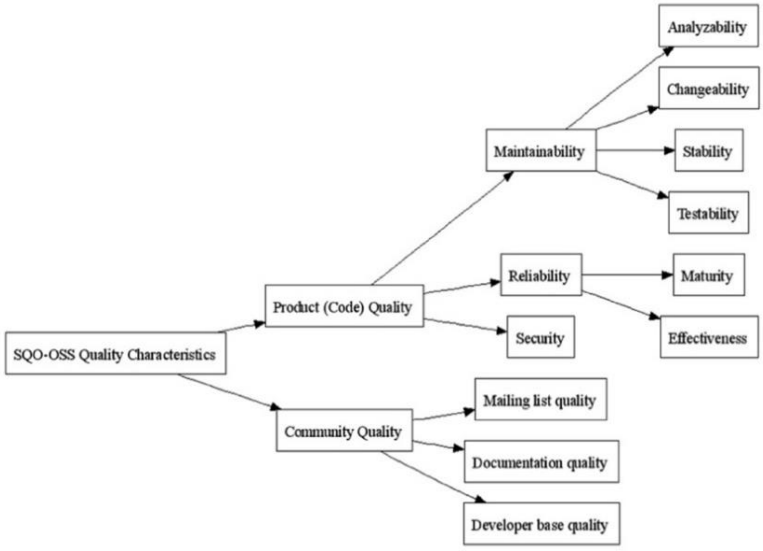
อย่างไรก็ตาม ผลลัพธ์สุดท้ายจากการประเมินแบบจำลองนี้ [3] ไม่มีการรวมค่าคะแนนรวมที่จะสามารถนำมาใช้เปรียบเทียบกับโอเพนซอร์ซอื่น ๆ อีกทั้งยังไม่สามารถกำหนดค่าถ่วงน้ำหนักของปัจจัยที่มีผลต่อผู้ประเมินได้ [8] รวมถึงตัววัดคุณภาพที่ระบุข้างต้นโดยส่วนใหญ่เป็นค่าวัดที่จำเป็นต้องใช้คนในการจัดหา ด้วยเหตุนี้ทำให้ไม่สามารถตอบโจทย์เป้าหมายของวิทยานิพนธ์นี้ที่ต้องการเปรียบเทียบโอเพนซอร์ซได้เร็ว และอัตโนมัติ แต่อย่างไรก็ตาม ตัววัดคุณภาพของงานวิจัยนี้ยังคงมีความน่าสนใจเพราะตัววัดบางค่าสามารถจัดหาข้อมูลได้แล้วในปัจจุบัน ด้วยเหตุนี้ ผู้วิจัยจึงได้นำตัววัดบางส่วนมาประยุกต์ใช้ในการสร้างแบบจำลองคุณภาพต่อไป

2.2.6 งานวิจัย SQO-OSS: The SQO-OSS Quality Model Measurement Based Open Source Software Evaluation

งานวิจัยนี้ [11] นำเสนอแบบจำลองคุณภาพที่ค่อนข้างแตกต่างจากแบบจำลองทั่วไป [3],[6],[5],[7],[14] ซึ่งแบบจำลองคุณภาพเหล่านี้ให้ความสำคัญกับผู้ประเมินเป็นหลัก แต่เนื่องจากแบบจำลองคุณภาพเอสคิวโอโอเอสเอส (SQO-OSS) ต้องการประเมินแบบอัตโนมัติ อีกทั้งงานวิจัยนี้ยังมองเห็นโอกาสที่จะประเมินจากซอร์ซโค้ดได้โดยตรง และต้องการลดคนในการมีส่วนร่วมในการประเมินให้น้อยที่สุดเพื่อความรวดเร็ว และนอกจากนี้แบบจำลองคิวโอโอเอสเอสได้คิดค้นค่าวัดที่ใช้ประเมินแหล่งชุมชนพัฒนาที่สามารถดึงได้อัตโนมัติอีกด้วย

แบบจำลองคุณภาพคิวโอโอเอสเอสแบ่งการประเมินออกเป็น 2 กลุ่ม ได้แก่ (1) ประเมินจากคุณภาพของโค้ด (Code Quality) และ (2) ประเมินจากคุณภาพแหล่งชุมชนพัฒนา (Community Quality) โดยผู้วิจัย [11] ได้ระบุแนวทางการออกแบบว่าตัววัดต่าง ๆ ที่คิดค้นจะต้องสามารถทำแบบ

อัตโนมัติเท่านั้น ดังนั้นค่าวัดอื่น ๆ เช่น การใช้งานง่าย (Usability) ที่จะต้องมีการทดลองใช้งานจากผู้ใช้จึงได้ถูกนำออกไป โดยโครงสร้างคุณภาพแบบจำลอง [11] สามารถแสดงดังในรูปที่ 2.3



รูปที่ 2.3 แบบจำลองคุณภาพเอสคิวโอโอเอสเอส (SQO-OSS) [11]

เนื่องจากงานวิจัยนี้ [11] เน้นการวัดคุณภาพของโอเพนซอร์ซจากซอร์ซโค้ดเป็นหลัก ซึ่งจากการตรวจสอบโดยถี่ถ้วนแล้ว ตัววัดคุณภาพที่ระบุข้างต้นโดยส่วนใหญ่เป็นค่าวัดที่จำเป็นต้องใช้เครื่องมือที่รองรับภาษาคอมพิวเตอร์นั้น ๆ ในการวัด ซึ่งในปัจจุบันยังคงไม่มีเครื่องมือใดที่สามารถวัดได้ในทุกภาษาคอมพิวเตอร์ตามค่าวัดที่ระบุ ยกตัวอย่างเช่น ตัววัดระดับความลึกของการสืบทอดคุณสมบัติของคลาส (Depth of inheritance tree: DIT) สามารถวัดได้จากเครื่องมือซีเคเมทริกส์ (CK Metrics) ที่ซอร์ซโค้ดถูกเขียนขึ้นจากภาษาจาวา (Java) แต่ไม่มีเครื่องมือรองรับในภาษาไพทอน (Python) หรือ ภาษาจาวาสคริปต์ (JavaScript) เป็นต้น ทำให้ไม่สามารถวัดค่าจากซอร์ซโค้ดที่เป็นภาษาข้างต้นได้ ดังนั้นผู้วิจัยจึงเห็นโอกาสในการต่อยอดด้วยการปรับปรุงแบบจำลองซอฟต์แวร์โอเพนซอร์ซต่อไป

ผู้วิจัยเล็งเห็นความสำคัญของการวัดคุณภาพจากซอร์ซโค้ดว่ามีความสำคัญไม่แพ้กับปัจจัยอื่น ๆ จึงได้นำตัววัดที่น่าสนใจจากงานวิจัยนี้ [11] มาเป็นส่วนหนึ่งของแบบจำลองซอฟต์แวร์โอเพนซอร์ซ และผู้วิจัยได้นำตัววัดต่าง ๆ จากซอร์ซโค้ดที่ได้จากเครื่องมือโซนาร์คิวบ์มาเพิ่มเติม เพื่อให้ตัววัดเหล่านี้เป็นตัววัดที่สะท้อนจากซอร์ซโค้ดของโครงการ โอเพนซอร์ซอย่างสมบูรณ์ยิ่งขึ้น

2.2.7 งานวิจัย OSS PESTO: An Open Source Software Project Evaluation and Selection Tool

งานวิจัยนี้ [4] นำเสนอเครื่องมือที่สนับสนุนการเปรียบเทียบและคัดเลือก โอเพนซอร์ซจากเว็บไซต์กิตฮับ โดยมีชื่อว่าโอเอสเอสเพสโต (OSS PESTO) โดยจะต้องเลือกแบบจำลองคุณภาพที่มีอยู่ในปัจจุบันเป็นข้อมูลนำเข้า เช่น โอเอสเอ็มเอ็ม (OSMM) [6], โอเพนบีอาร์อาร์ (OpenBRR) [3], คิวเอสโอเอส (QSOS) [7], โอเอสเอสพาล (OSSPAL) [5] เป็นต้น โดยงานวิจัยนี้ใช้วิธีการดึงข้อมูลจากกิตฮับแบบอัตโนมัติผ่านทางเอพีไอที่ทางกิตฮับได้ให้บริการไว้ ซึ่งผู้ใช้สามารถตั้งค่าเลือกข้อมูลที่ต้องการได้ และสามารถเลือกแบบจำลองที่จะใช้ในการประเมินและเปรียบเทียบได้ โดยระบบจะให้ผู้ใช้งานป้อนชื่อโครงการใดก็ได้ในกิตฮับที่เปิดเป็นสาธารณะอยู่ จากนั้นระบบจะทำการดึงข้อมูลผ่านทางเอพีไอ และนำมาเก็บไว้ที่ฐานข้อมูลของระบบโดยเก็บในรูปแบบ CSV (ซีเอสวี) และนำมาแสดงผลตารางให้ผู้ประเมินได้เปรียบเทียบดังรูปที่ 2.4

(A) Community aspect comparison

Project Name	#Watch	#Star	Age	Avg. Issue Active	Avg. Issue Comments	#Pull Request	#Issue Raiser
vuejs/vue	6356	176731	8	1623915.3355	3.1395	1926	6269
angular/angularjs	3999	59588	11	9195821.9838	4.7196	7960	7889
reduxjs/redux	1436	54891	6	1643166.0109	4.6450	2093	2435

(B) Support aspect comparison

Project Name	Avg. Issue Closed	#Contributor	Org Issue Raiser
vuejs/vue	832537.0758	382	4
angular/angularjs	7606054.5190	1324	16
reduxjs/redux	1516982.0928	824	0

(C) Software Technology Attributes aspect comparison

Project Name	Dependence	Open Issues
vuejs/vue	74	339
angular/angularjs	150	391
reduxjs/redux	39	33

รูปที่ 2.4 ผลลัพธ์การประเมินจากเครื่องมือโอเอสเอสเพสโต OSS Pesto [4]

ในกระบวนการทดสอบเครื่องมือ ผู้วิจัยได้เลือกโครงการแองกูลาร์ (Angular) วิว (Vue) และรีดักซ์ (Redux) ซึ่งเป็นโครงการโอเพนซอร์ซในกิตฮับ และเลือกแบบจำลองโอเอสเอสพาล (OSSPAL) เป็นแบบจำลองคุณภาพต้นแบบ โดยเลือกตัววัด 3 กลุ่มจากแบบจำลอง กลุ่มแรกคือ แหล่งชุมชน ได้แก่ จำนวนของผู้ติดตาม (Number of watches), จำนวนกิตฮับสตาร์ (Number of GitHub star), อายุของโครงการ (Project age) กลุ่มที่สอง คือ การสนับสนุน ได้แก่ จำนวนผู้พัฒนา (Number of contributors) เป็นต้น และสุดท้ายกลุ่มคุณลักษณะเทคโนโลยีซอฟต์แวร์ ได้แก่ จำนวนของข้อบกพร่องที่เปิด (Number of open issues), จำนวนโครงการที่ขึ้นต่อกัน (Number of dependence) เป็นต้น ผลลัพธ์ที่ได้ (อ้างอิงตามจากรูปที่ 2.4) คือ กลุ่มแหล่งชุมชน (A) โปรเจควิวมีคุณภาพสูงสุด ในขณะที่กลุ่มของการสนับสนุน (B) โปรเจคแองกูลาร์มีคุณภาพสูงสุด และ

สุดท้าย กลุ่มคุณลักษณะเทคโนโลยีซอฟต์แวร์ โปรเจครีดักซ์มีคุณภาพที่ดีที่สุด ทำให้ไม่สามารถสรุปได้ว่า โปรเจคใดเหมาะสมที่สุด

อย่างไรก็ตาม ผู้วิจัยเล็งเห็นข้อควรปรับปรุงหลายประการ ประการแรก เครื่องมือโอเอสเอสเพสโตใช้เพียงเอพีไอเพื่อดึงข้อมูล และนำข้อมูลดิบมาแสดงตารางเปรียบเทียบ ซึ่งทำให้ยากต่อการเปรียบเทียบ ประการที่สอง ผู้วิจัย [4] ละทิ้งตัววัดบางกลุ่มจากแบบจำลองคุณภาพโอเอสเอสพาลที่เลือกมา เพราะเอพีไอของกิตฮับไม่รองรับทำให้การวิเคราะห์จากการใช้แบบจำลองคุณภาพโอเอสเอสพาลที่นำมาอ้างอิงนั้นยังคงไม่สมบูรณ์ ประการสุดท้าย ข้อมูลที่นำมาจากกิตฮับมีน้ำหนักเท่ากัน โดยทางปฏิบัติจำเป็นต้องมีการถ่วงน้ำหนักซึ่งขึ้นอยู่กับความต้องการจากผู้ใช้งาน หรือจากผู้ประเมิน

จากงานวิจัยนี้ ผู้วิจัยได้รับแรงบันดาลใจในการนำข้อมูลที่ได้จากกิตฮับเพื่อคิดค้นตัววัดใหม่ ๆ และอีกทั้งยังได้รายละเอียดและขั้นตอนการสร้างเครื่องมือด้วยวิธีการดึงข้อมูลผ่านทางเอพีไอของกิตฮับ เพื่อนำไปปรับปรุงและพัฒนาเครื่องมือสนับสนุนใหม่ต่อไป

2.2.8 งานวิจัย OpenBQR: a framework for the assessment of OSS

งานวิจัย [10] นำเสนอโมเดลคุณภาพใหม่ที่ต่อยอดจากงานวิจัยโมเดลคุณภาพโอเพนบีอาร์อาร์ (OpenBRR) รวมกับโมเดลคุณภาพคิวเอสโอเอส (QSOS) โดยโมเดลคุณภาพโอเพนบีคิวอาร์ต้องการต่อยอดใน 3 แง่มุม ดังนี้

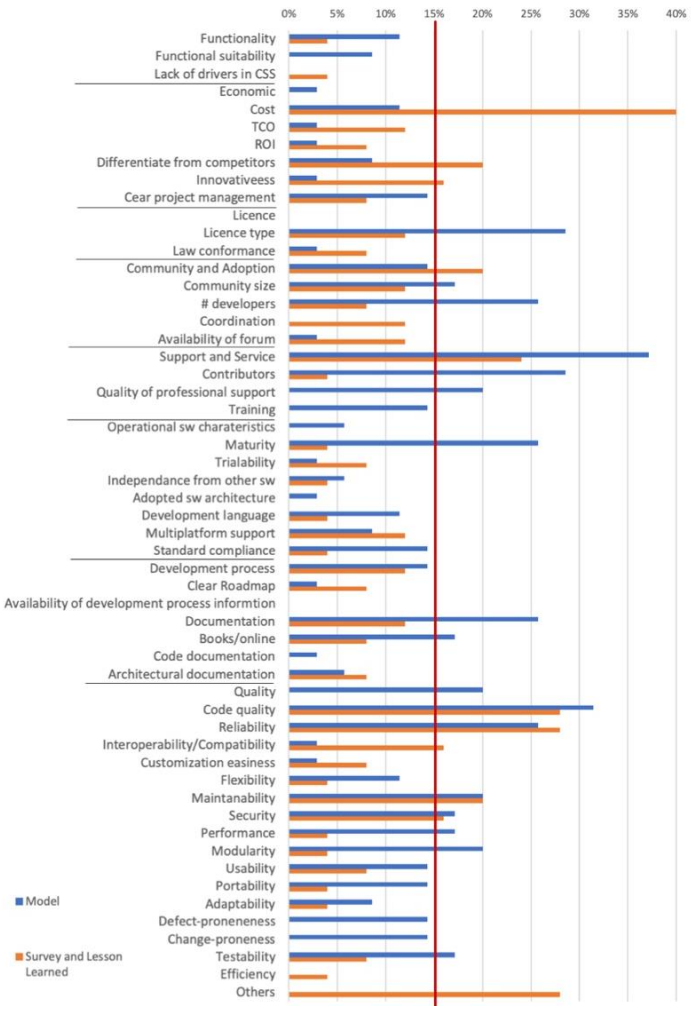
1. เป้าหมายในการใช้งาน (Target Usage) มีตัวชี้วัดคุณภาพประกอบไปด้วย ประเภทของไลเซนส์ (Type of license), การปฏิบัติตามมาตรฐานสากล (Standard compliance), ภาษาที่ใช้ในการพัฒนา (Implementation language), การรอบรับหลายภาษา (*Internationalization Support)*, หนังสือ (*Books) เป็นต้น*
2. คุณภาพภายนอก (External Quality) ตัวชี้วัดคุณภาพประกอบไปด้วย จำนวนบัค (Bug number), เวลาเฉลี่ยที่ใช้ในการแก้ไขข้อผิดพลาด (Average time for defect removal) เป็นต้น
3. คุณภาพภายใน (Internal Quality) ตัวชี้วัดคุณภาพประกอบไปด้วย ความซับซ้อน (Complexity), การนำไปใช้ซ้ำ (Reuse), ความเกี่ยวข้องกัน (Dependencies) เป็นต้น
4. การสนับสนุนในอนาคต (Future Support) ตัวชี้วัดคุณภาพประกอบไปด้วย จำนวนการปล่อย (Number of releases), จำนวนองค์กรที่ให้การสนับสนุน (Number of organizations supporting the software), จำนวนนักพัฒนาที่มาจากองค์กร (Number of programmers per organization) เป็นต้น

โมเดลคุณภาพอย่างโอเพนบีคิวอาร์นั้นจะเป็นต้องมีผู้ประเมินในการทำการประเมินคุณภาพของโอเพนซอร์ซโดยงานวิจัยได้นำเสนอเครื่องมือในการทำการประเมินคุณภาพโดยจะต้องให้ผู้ประเมินนำข้อมูลเข้าเป็นคะแนน 0-10 คะแนน สำหรับแต่ละตัวชี้วัดคุณภาพ และจะสามารถให้ค่าถ่วงน้ำหนักจาก 0-10 ด้วยเช่นกัน โดยทั้งหมดนี้จะใช้การรวมแบบค่าเฉลี่ยถ่วงน้ำหนักและมีผลลัพธ์คะแนนตั้งแต่ 0-100 คะแนนเป็นคะแนนสุดท้ายสำหรับแต่ละอเพนซอร์ซที่เลือกประเมิน โดยสรุปแล้ว โมเดลคุณภาพโอเพนบีคิวอาร์นั้นมีส่วนคล้ายกับความต้องการของผู้วิจัยโดยสามารถหาค่าและรวมเป็นคะแนนรวมได้ แต่มีส่วนที่ไม่คล้ายกันตรงที่ขั้นตอนการประเมินทั้งหมดนั้นจะใช้ความรู้จากผู้ประเมินเป็นหลัก

บทที่ 3
แบบจำลองคุณภาพสำหรับการประเมินซอฟต์แวร์โอเพนซอร์ซ

3.1 การกำหนดปัจจัยคุณภาพ และปัจจัยคุณภาพย่อยที่สะท้อนคุณภาพของโอเพนซอร์ซ

การพัฒนาแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซจำเป็นจะต้องคำนึงถึงปัจจัยคุณภาพต่าง ๆ ที่ผู้ประเมินหรือผู้นำโอเพนซอร์ซไปใช้ให้ความสนใจ จากงานวิจัยของ Davide Taibi, Valentina Lenarduzzi และคณะ [1] ได้รวบรวมปัจจัยคุณภาพที่มีส่วนสำคัญในการคัดเลือก และนำโอเพนซอร์ซไปใช้งานจากการทบทวนวรรณกรรม โดยในงานวิจัยของ [1] ได้ระบุปัจจัยทั้งหมด 47 ปัจจัยคุณภาพย่อย (Sub-quality Factor) และได้จัดกลุ่มเป็น 8 ปัจจัยคุณภาพ (Quality Factor) ดังรูปที่ 3.1



รูปที่ 3.1 ปัจจัยในการคัดเลือกโอเพนซอร์ซจัดตามหมวดหมู่ของงานวิจัย [1]

จากงานวิจัยนี้ [1] ถือเป็นประโยชน์อย่างมากต่อการนำปัจจัยคุณภาพต่าง ๆ มาใช้เป็นส่วนประกอบของแบบจำลองซอฟต์แวร์โอเพนซอร์ซ โดยผู้วิจัยได้คัดเลือกปัจจัยที่มีการรายงานมากกว่าร้อยละ 15 ขึ้นไปเป็นหลัก (เส้นสีแดง) อันเนื่องจากผู้วิจัยมองว่าปัจจัยที่มีอัตราการอ้างอิงจากหลาย ๆ งานวิจัย มีโอกาสที่มีประโยชน์ต่อผู้ประเมินโดยส่วนใหญ่ ในขณะที่ปัจจัยคุณภาพที่น้อยกว่าร้อยละ 15 ส่วนใหญ่จะมีลักษณะจำเพาะสำหรับบางงานวิจัยหรือสามารถรวมเข้าด้วยกัน กับปัจจัยอื่นได้ โดยจากรูปที่ 3.1 ได้แสดงให้เห็นว่าปัจจัยต่าง ๆ นำมาจากงานวิจัยประเภทใดบ้าง โดยแถบสีฟ้าแสดงงานวิจัยที่ตีพิมพ์แบบจำลองคุณภาพ แถบสีส้มแสดงถึงปัจจัยคุณภาพที่มาจากงานวิจัยประเภทแบบสำรวจและประสบการณ์การใช้งานโอเพนซอร์ซ โดยผลลัพธ์จากขั้นตอนนี้ จะได้ปัจจัยที่คัดเลือกดังจะแสดงในตารางที่ 3.1

ตารางที่ 3.1 ปัจจัยคุณภาพ 5 กลุ่ม และปัจจัยคุณภาพย่อย 19 ปัจจัยที่นำมาจากงานวิจัย [1]

ปัจจัยคุณภาพ / ปัจจัยคุณภาพย่อย	**ความหมาย**
Software License	
License	Open-source software license and permission to distribute software or source code
Community and Support	
Community Size	Open source community size
Availability of Forum	The question/answer forum is the first place where people go for free help [3]. Questions and answers are valuable resources.
Support Contributors	Code contributors, i.e. core team members and other contributors, usually promote the building of community around the project. The higher the number of code contributors, the better the community support [3].
Quality of Professional Support	Professional support that helps fine-tune for the local deployment and troubleshooting is always desirable [3].
Operational Software Characteristics	
Maturity	The degree to which a system, product, or component meets the needs for reliability under normal operation [14].
Development Language	Development language is a factor that can impact the ease of finding a developer to operate and maintain the open-source project.
Documentation	Good documentation should include documentation for several user groups in several formats [3].

ตารางที่ 3.1 ปัจจัยคุณภาพ 5 กลุ่ม และปัจจัยคุณภาพย่อย 19 ปัจจัยที่นำมาจากงานวิจัย [1] (ต่อ)

ปัจจัยคุณภาพ / ปัจจัยคุณภาพย่อย	**ความหมาย**
Operational Software Characteristics	
Book/Online	The availability of published materials such as books, videos, and courses about the project is a strong indicator of the software's level of maturity and adoption. [3]
Economics	
Cost	The ability of the software to contribute positively to the financial balance. [8]
Innovativeness	The ability to allow everyone equal opportunity to use, modify and re-design OSS applications and source-code in their own way. [21]
Competitiveness	The ability of the project to gain market share and respond to requirements requested from users [6].
Product Quality (เปลี่ยนชื่อจาก Quality)	
Code Quality	The open-source software having high quality code is easier to understand for long-term maintenance and evolution.
Reliability	The degree to which a system, product or component performs specified functions under specified conditions for a specified period [14].
Maintainability	The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [14].
Security	The degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization [14].
Testability	The degree to which a software artifact supports testing in a given test context [14].
Compatibility	The degree to which a product, system, or component can exchange information with other products, systems, or components, and/or perform its required functions, while sharing the same hardware or software environment [14].
Performance	The ability of the software to perform its function within given constraints concerning the consumption of resources and time [14].

3.2 การกำหนดตัววัดคุณภาพที่สะท้อนปัจจัยคุณภาพ

ตัววัดคุณภาพ (Quality Metric) ที่สะท้อนปัจจัยคุณภาพย่อยเป็นตัววัดที่สามารถนำไปวัดได้จากแหล่งข้อมูลที่มีอยู่ในปัจจุบัน หรือจากซอร์ซโค้ดของโครงการ ผู้วิจัยได้กำหนดให้มีตัววัดซึ่งนิยามด้วยสมการการวัด (Measurement Fuction) โดยที่แต่ละสมการการวัดจะประกอบด้วยองค์ประกอบของตัววัด (Quality Metric Element) หรือตัวแปร 1-3 ตัว โดยตัววัดต่าง ๆ นั้นได้นำมาจากงานวิจัยแบบจำลองคุณภาพอย่างโอเพนบีอาร์อาร์ (OpenBRR) [3], โอเอสเอ็มเอ็ม (OSMM) [6], เอสคิวโอโอเอสเอส (SQO-OSS) [11] และโอเพนบีคิวอาร์ (OpenBQR) [10] รวมทั้งแหล่งข้อมูลอื่น ๆ ซึ่งในอดีตค่าคุณภาพเหล่านี้จะต้องจัดหาหรือพิจารณาจากความรู้และประสบการณ์โดยผู้ประเมินเองซึ่งใช้ระยะเวลานาน แต่ในปัจจุบันสามารถจัดหาข้อมูลได้จากแหล่งข้อมูลต่าง ๆ ดังแสดงในตารางที่ 3.2 ภายใต้เงื่อนไขที่จะต้องสามารถหาค่าได้อย่างอัตโนมัติจากแหล่งข้อมูลข้างต้น ดังนั้นตัววัดที่ไม่สามารถหาค่าได้อัตโนมัติจะไม่นำมาระบุในงานวิจัยนี้

ตารางที่ 3.2 แหล่งข้อมูลสำหรับโอเพนซอร์ซที่สามารถดึงข้อมูลอัตโนมัติ

แหล่งข้อมูล	คำย่อ	ข้อมูลที่สามารถจัดเก็บ
กิตฮับ (GitHub)	GH	• ข้อมูลจากกิตฮับเอพีไอ (GitHub API) • ซอร์ซโค้ดของโครงการ รวมถึงไฟล์ Markdown (*.md) ต่าง ๆ
โซนาร์คิวบ์ (SonarQube)	SQ	• ค่าที่วัดได้จากการวิเคราะห์ซอร์ซโค้ด
สแต็กเอกซ์เชนจ์ (Stack Exchange)	SE	• ข้อมูลความคิดเห็นจากผู้ใช้งาน • ปัญหาจากการใช้งานโครงการโอเพนซอร์ซที่กำลังสนใจ • ข้อมูลถามตอบที่เป็นประโยชน์ต่อการแก้ปัญหาจากการใช้งาน
กูเกิลบุ๊กเอพีไอ (Google Book API)	Goog	• ข้อมูลหนังสือที่ตีพิมพ์เกี่ยวกับโครงการโอเพนซอร์ซ
ยูทิวบ์เอพีไอ (Youtube API)	YT	• แหล่งเรียนรู้ประเภทวีดีโอที่สอนการใช้งาน

นอกเหนือจากงานวิจัยแบบจำลองคุณภาพ [3],[6] และ [11] ผู้วิจัยได้นำตัววัดคุณภาพมาจากงานวิจัย [2] อีกด้วย ซึ่งได้ระบุตัววัดเพิ่มเติมที่ได้จากเครื่องมือโซนาร์คิวบ์ ซึ่งเป็นค่าที่ได้จากการวิเคราะห์ข้อมูลจากซอร์ซโค้ดของโครงการโอเพนซอร์ซ นอกจากนี้ปัจจัยอื่น ๆ ที่ยังไม่สามารถหาตัววัดมารองรับได้ ผู้วิจัยจึงได้เสนอตัววัดใหม่ขึ้นมาบางส่วนโดยให้อยู่ในขอบเขตของแหล่งข้อมูลที่รองรับ โดยผลลัพธ์จากขั้นตอนนี้จะสามารถแสดงดังตารางที่ 3.3 โดยในรายละเอียดของแต่ละตัววัดจะกล่าวต่อไปในขั้นตอนการพัฒนาแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซ

ตารางที่ 3.3 ตัววัดและองค์ประกอบของตัววัดที่สะท้อนปัจจัยคุณภาพย่อยของแบบจำลองฯ

รหัส	ปัจจัยคุณภาพย่อย	ตัววัด	องค์ประกอบของตัววัด	ความหมาย	แหล่งข้อมูล	ที่มาขององค์ประกอบของตัววัด
V1	License	OSS License	License Type	The open-source software license and permission to distribute software or source code.	GH	ปรับปรุงจาก OSMM [6]
V2	Community Size	Size of Community	Number of Core Team Members, Contributors, and Watchers	The number of members in the open-source community.	GH	ปรับปรุงจาก Eurostat Glossary: Enterprise Size Website
V3	Availability of Forum	Q&A Volume	Average Q&A	Average number of questions and answers in the forum per month in the past 6 months.	SE	ปรับปรุงจาก OpenBRR [3]
V4	Support Contributors	Number of Support Contributors	Number of Contributors and Core Team Members	The number of open-source project members who have contributed to the commit history of the project, and the number of core members of the open-source project, including owner and dedicated members, who determine the direction and evolution of the project.	GH	ปรับปรุงจาก OpenBRR [3]

ตารางที่ 3.3 ตัววัดและองค์ประกอบของตัววัดที่สะท้อนปัจจัยคุณภาพย่อยของแบบจำลองฯ (ต่อ)

รหัส	ปัจจัยคุณภาพย่อย	ตัววัด	องค์ประกอบของตัววัด	ความหมาย	แหล่งข้อมูล	ที่มาขององค์ประกอบของตัววัด
V5	Quality of Professional Support	Support Activity	Issue Support Activity	The proportion of issues that have been responded in the past 6 months. If there are no reported issues, there will be no evidence to show whether the support is active recently.	GH	นำเสนอใหม่
			Pull Request Support Activity	The proportion of pull requests that have been responded in the past 6 months. If there are no pull requests, there will be no evidence to show whether the support is active recently.	GH	นำเสนอใหม่
V6	Maturity	Maturity Level	Age	The age of project repository from created date to last update.	GH	ปรับปรุงจาก OSMM [6]
			Issueless Code	A characteristic of the code with small number of issues in the past 6 months. It indicates that the open-source software is likely to be mature and work correctly under normal operation.	GH	ปรับปรุงจาก SQO-OSS [11] และ OpenBRR [3]
			Minor Releases	The number of minor releases in the past 12 months. It typically indicates that the project has planned updates and bug fixes.	GH	นำมาจาก OpenBRR [3]

ตารางที่ 3.3 ตัววัดและองค์ประกอบของตัววัดที่สะท้อนปัจจัยคุณภาพย่อยของแบบจำลองฯ (ต่อ)

รหัส	ปัจจัยคุณภาพย่อย	ตัววัด	องค์ประกอบของตัววัด	ความหมาย	แหล่งข้อมูล	ที่มาขององค์ประกอบของตัววัด
V7	Development Language	Development Language Popularity	Computer Language	How commonly the languages are used by professional developers.	GH	นำเสนอใหม่
V8	Documentation	Code Documentation	Code Comments	The amount of code comments, compared with the total amount of code. Some of the comments may be used to generate API documents.	SQ	นำมาจาก SonarQube
			Markdown Files	The proportion of markdown files in the project.	GH	นำเสนอใหม่
V9	Book/Online	Learning Materials	Books and Courses	The number of book titles about the software and the number of videos about the software.	Goog, YT	ปรับปรุงจาก OpenBRR [3]
V10	Cost	Cost of Ownership Reduction	Maintainability Level	The degree of ease with which the open-source software can be maintained and is defined in terms of the Technical Debt Ratio of the software, i.e. the ratio between the cost to fix all code smells in the software and the cost to develop the software. When maintainability is high, maintenance costs can be reduced.	SQ	นำเสนอใหม่

ตารางที่ 3.3 ตัววัดและองค์ประกอบของตัววัดที่สะท้อนปัจจัยคุณภาพย่อยของแบบจำลองฯ (ต่อ)

รหัส	ปัจจัยคุณภาพย่อย	ตัววัด	องค์ประกอบของตัววัด	ความหมาย	แหล่งข้อมูล	ที่มาขององค์ประกอบของตัววัด
V10 (ต่อ)	Cost	Cost of Ownership Reduction	Support Activity	The degree of support activity for issues and pull requests in the past 6 months. If there are no reported issues and pull requests, there will be no evidence to show whether the support is active recently. With active support from The contributors and core team members, development costs and maintenance costs can be reduced.	GH	นำเสนอใหม่
			Learning Materials	The number of published learning materials for the open-source software, i.e. books and courses. If there are none of these materials, learning about the software will have to rely on the published source code and code documentation. The materials can reduce training costs.	Goog, YT	นำเสนอใหม่
V11	Innovativeness	New Features Focus	New Feature Pull Requests	The proportion of pull requests labeled with new features in the past 6 months.	GH	นำเสนอใหม่
V12	Competitiveness	Continuing Change	Pull Request Frequency	The frequency of pull requests that have proposed change to the project in the past 30 days.	GH	นำเสนอใหม่

ตารางที่ 3.3 ตัววัดและองค์ประกอบของตัววัดที่สะท้อนปัจจัยคุณภาพย่อยของแบบจำลองฯ (ต่อ)

รหัส	ปัจจัยคุณภาพย่อย	ตัววัด	องค์ประกอบของตัววัด	ความหมาย	แหล่งข้อมูล	ที่มาขององค์ประกอบของตัววัด
V13	Code Quality	Code Quality Level	Uncomplex Code	A characteristic of the code with less degree of cyclomatic complexity which indicates that the code has better quality.	SQ	นำมาจาก Code Metrics Cyclomatic Complexity Website
			Unduplicated Code	A characteristic of the code with less degree of duplications which indicates that the code has better quality.	SQ	นำเสนอใหม่
V14	Reliability	Reliability Level	Reliability Rating	The degree of quality of the code that is bug-free or has less severe bugs.	SQ	นำมาจาก SonarQube
V15	Maintainability	Maintainability Level	Maintainability Rating	The degree of quality of the code having less technical debt ratio.	SQ	นำมาจาก SonarQube
V16	Security	Security Level	Security Rating	The degree of quality of the code that is vulnerability-free or has less severe vulnerabilities.	SQ	นำมาจาก SonarQube
V17	Testability	Testability Level	Uncomplex Code	A characteristic of the code with less degree of cyclomatic complexity which indicates that there is a smaller number of paths through the code, making it less complex to test.	SQ	นำมาจาก Code Metrics Cyclomatic Complexity Website

ตารางที่ 3.3 ตัววัดและองค์ประกอบของตัววัดที่สะท้อนปัจจัยคุณภาพย่อยของแบบจำลองฯ (ต่อ)

รหัส	ปัจจัยคุณภาพย่อย	ตัววัด	องค์ประกอบของตัววัด	ความหมาย	แหล่งข้อมูล	ที่มาขององค์ประกอบของตัววัด
V18	Compatibility	Co-existence	Development Environment	The types of development environment, i.e. web, mobile, desktop, or embedded development environment, which the primary programming language of the project can support.	GH	นำเสนอใหม่
V19	Performance	Lack of Performance Issues Level	Lack of Performance Issues	The proportion of non-performance issues in relation to other reported issues in the past 6 months.	GH	นำเสนอใหม่

3.3 การพัฒนาแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซ

ในขั้นตอนการพัฒนาแบบจำลองคุณภาพนี้จะต้องนำข้อมูลจากขั้นตอนก่อนหน้า ได้แก่ ปัจจัยคุณภาพ ปัจจัยคุณภาพย่อย ตัววัด และองค์ประกอบของตัววัด มาประกอบกันเพื่อให้ได้แบบจำลองคุณภาพที่สมบูรณ์ อีกทั้งในขั้นตอนนี้ยังต้องกำหนดสมการการวัดของตัววัดคุณภาพแต่ละรายการที่ประกอบขึ้นจากองค์ประกอบของตัววัดคุณภาพที่สนใจ และนำทุกค่าตัววัดมาหาค่าคะแนนรวมโดยใช้วิธีการหาค่าเฉลี่ยถ่วงน้ำหนัก (Weighted Average Method) [3] เพื่อให้ได้ค่าคะแนนคุณภาพโดยรวมของโอเพนซอร์ซ โดยสมการจะแสดงรายละเอียดดังตารางที่ 3.4

ตารางที่ 3.4 ตัววัดคุณภาพของโครงการโอเพนซอร์ซ

ชื่อตัววัด	คุณภาพของโครงการโอเพนซอร์ซ (Open-Source Quality)
คุณลักษณะเชิงคุณภาพ	คุณภาพของโครงการโอเพนซอร์ซโดยรวม
วัตถุประสงค์ของการวัด	เพื่อสะท้อนคุณภาพของโครงการโอเพนซอร์ซโดยรวม และค่าคะแนนสามารถนำไปเปรียบเทียบกับโครงการโอเพนซอร์ซอื่น ๆ ได้ ทำให้ผู้ประเมินสามารถคัดเลือกโอเพนซอร์ซได้ตามคุณภาพจริง
สมการการวัด	$$Open\ Source\ Quality = \frac{\sum_{i=1}^{n}(W_i \times V_i)}{\sum_{i=1}^{n} W_i}$$ โดยที่ Open-Source Quality คือค่าคะแนนคุณภาพของโอเพนซอร์ซโดยรวม W_i ค่าถ่วงน้ำหนักของแต่ละปัจจัยคุณภาพย่อย โดยที่ $W_i > 0$ กำหนดให้ค่าโดยปริยายเป็น 1 V_i คะแนนของตัววัดคุณภาพของแต่ละปัจจัยคุณภาพย่อย n จำนวนปัจจัยคุณภาพย่อยทั้งหมดที่นำมาคำนวณ
เกณฑ์การตีความผลลัพธ์	ผลลัพธ์ที่ได้ตามสมการข้างต้นจะเป็นตัวเลขทศนิยมในช่วง (0,100] โดยเป็นคะแนนเฉลี่ยที่บอกคุณภาพของโครงการโอเพนซอร์ซโดยรวม และสามารถนำค่าคะแนนที่ได้ไปเปรียบเทียบกับอีกโครงการหนึ่งได้

จากสมการตัววัดคุณภาพของโครงการโอเพนซอร์ซ ค่า W_i เป็นค่าที่ผู้ประเมินจะต้องกำหนดค่าให้ โดยให้น้ำหนักตามความสำคัญของปัจจัยคุณภาพย่อยที่ผู้ประเมินสนใจ แต่เพื่อให้ง่ายต่อการคำนวณ ดังนั้นแบบจำลองคุณภาพจึงได้กำหนดค่าถ่วงน้ำหนักโดยปริยายเท่ากับ 1 ทุกตัววัด

ลำดับต่อไปเป็นการอธิบายรายละเอียดตัววัดคุณภาพแต่ละรายการ นอกจากนี้โครงสร้างลำดับชั้นที่สะท้อนเป็นการเรียงระดับจากชั้นบนสุดลงมาล่างสุด ซึ่งระดับชั้นล่างจะสะท้อนระดับชั้นก่อนหน้า ซึ่งจะเขียนตามรูปแบบ [ปัจจัยคุณภาพ] > [ปัจจัยคุณภาพย่อย] > [ตัววัดคุณภาพ] โดยตัววัดคุณภาพมีทั้งหมด 20 รายการ ดังแสดงตัวอย่างไว้ในตารางที่ 3.5 และ ตารางที่ 3.6 ทั้งนี้ภาพรวมของแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซแสดงไว้ในรูปที่ 3.2 (นิยามของตัววัดคุณภาพทั้งหมดแสดงไว้ในภาคผนวก ก.)

ตารางที่ 3.5 ตัวอย่างตัววัดคุณภาพ V1 – OSS License

Metric name	V1 – OSS License
Hierarchy structure	Software License > License > OSS License
Description	OSS License refers to the open-source software license and permission to distribute software or source code. It consists of one quality metric element: • License Type refers to different types of open-source licenses that define how the open source can be used, modified, and shared.
Equation	$V_1 = \frac{M}{5} \text{x} 100$ where V1 OSS License M License Type M = 1 Undefined/Unclear M = 2 LGPL-2.1, GPL-2.0 M = 3 AGPL-3.0, EPL-2.0, GPL-3.0, MPL-2.0 M = 4 BSD-2-Clause, BSD-3-Clause, BSL-1.0, CC0-1.0, The Unlicensed M = 5 Apache 2.0, MIT
Reference model	Metric element M adapted from OSMM [6]

ตารางที่ 3.5 ตัวอย่างตัววัดคุณภาพ V1 – OSS License (ต่อ)

Information source	GH – License API (https://docs.github.com/en/rest/reference/licenses) License types reference (https://choosealicense.com/appendix/)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ 3.6 ตัวอย่างตัววัดคุณภาพ V11 – New Features Focus

Metric name	V11 – New Features Focus
Hierarchy structure	Economics > Innovativeness > New Features Focus
Description	New Features Focus refers to the degree to which the change as new feature enhancement has been introduced to the open-source software in relation to the effort to effect change to the open source. This indicates the degree of the contributors' focus on innovativeness of the open source. The focus on enhancing the open source with new features helps keep the software that utilizes the open source upgraded and modernized, making it beneficial for economic reasons. The metric consists of one quality metric element: • New Feature Pull Requests refers to the proportion of pull requests labeled with new features in the past 6 months.
Equation	(1) $$V_{11} = M \times 100$$ where V_{11} New Features Focus M New Feature Pull Requests

ตารางที่ 3.6 ตัวอย่างตัววัดคุณภาพ V11 – New Features Focus (ต่อ)

Equation	(2) $$M = \begin{cases} \left(\dfrac{Number\ of\ new\ feature\ pull\ requests\ in\ the\ past\ 6\ months}{N} \right), & N > 0 \\ 0, & N = 0 \end{cases}$$ where M New Feature Pull Requests N Total pull requests in the past 6 months
Reference model	Metric element M newly introduced
Information source	GH – Pull requests with features label (https://docs.github.com/en/rest/reference/pulls#list-pull-requests) Search keywords: feature, new feature, user request, redesign, new function, new item, new component.
Output	The result of the equation is a decimal number in the range [0,100].

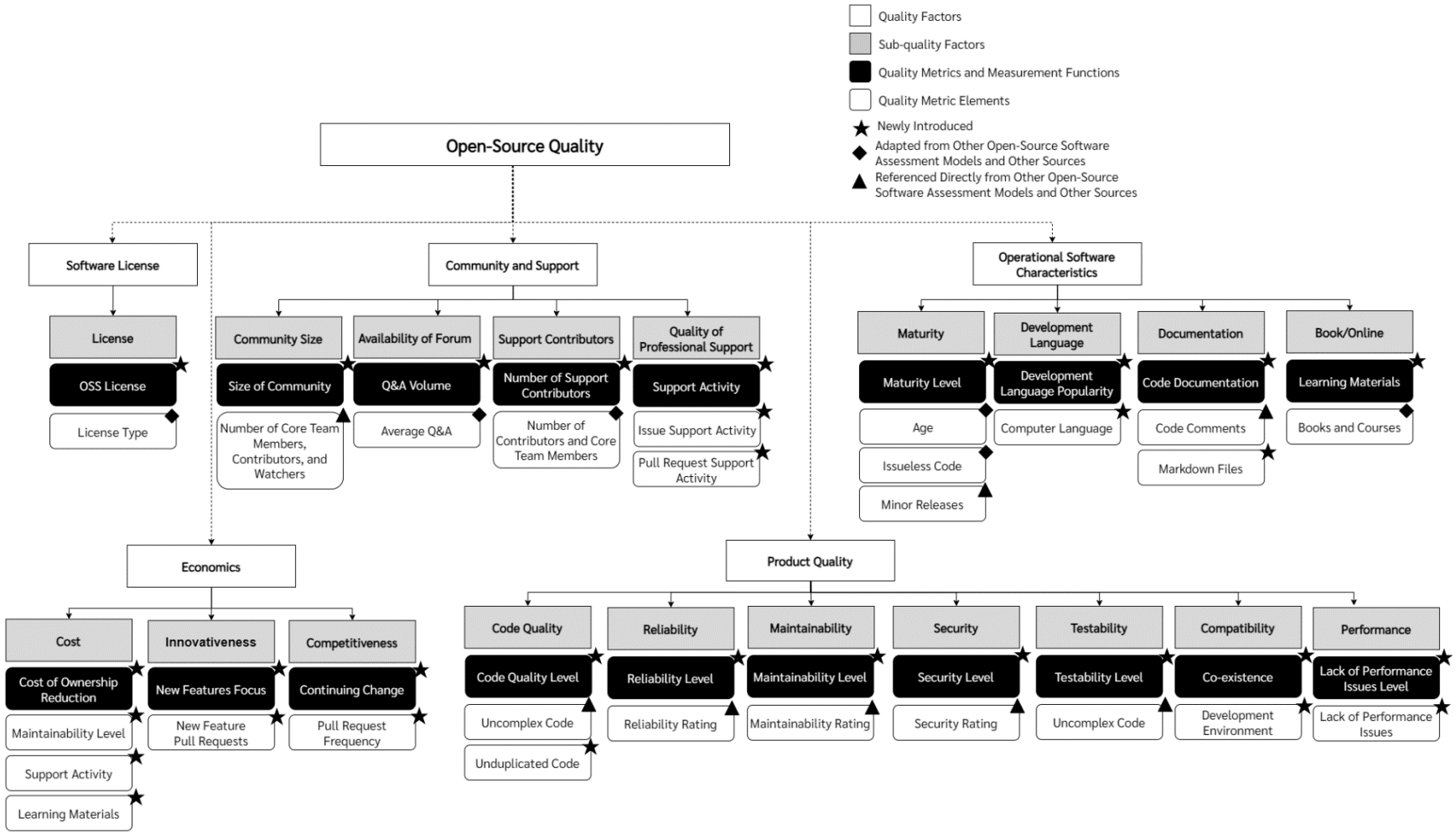
3.4 สรุปโครงสร้างแบบจำลองแบบสมบูรณ์

ในขั้นตอนสุดท้ายของการสร้างแบบจำลองคุณภาพฯ นี้ ผลลัพธ์ของแต่ละขั้นตอนก่อนหน้านี้จึงถูกนำมารวมกันประกอบเป็นแบบจำลองโดยสมบูรณ์ โดยสรุปแล้ว แบบจำลองคุณภาพฯ ประกอบด้วย ปัจจัยคุณภาพ (Quality Factor) และปัจจัยคุณภาพย่อย (Sub-quality Factor) โดยแต่ละปัจจัยมีตัววัดคุณภาพ (Quality Metrics) เพื่อสะท้อนคุณภาพของแต่ละปัจจัยย่อย แต่ละตัววัดคุณภาพจะประกอบด้วยสมการการวัด องค์ประกอบของตัววัด และแหล่งข้อมูลที่ต้องใช้ โดยแบบจำลองนี้จะสามารถคำนวณคะแนนของซอฟต์แวร์โอเพนซอร์ซ และได้คะแนนรวมตามที่ต้องการ ภาพรวมแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซเป็นดังรูปที่ 3.2 ปัจจัยคุณภาพ (5 ปัจจัย) และปัจจัยคุณภาพย่อย (19 ปัจจัย) ทั้งหมดอ้างอิงมาจากแบบจำลองคุณภาพอื่นที่รวบรวมโดย [1] ในขณะที่ตัววัดคุณภาพทั้งหมด (19 ตัว) ผู้วิจัยจะนิยามสมการการวัดขึ้นมาใหม่ทั้งหมดเพื่อให้สามารถวัดค่าปัจจัยคุณภาพย่อยที่เกี่ยวข้องเป็นค่าเชิงปริมาณที่สามารถนำมาใช้คำนวณร่วมกันได้

ส่วนองค์ประกอบของตัววัด (25 ตัว) จะมีทั้งที่ 1) นำมาโดยตรงจากแบบจำลองคุณภาพอื่นหรือแหล่งข้อมูลอื่น (6 ตัว) 2) นำเสนอใหม่โดยปรับปรุงจากแบบจำลองคุณภาพโอเพนซอร์ซอื่นหรือแหล่งข้อมูลอื่น (7 ตัว) 3) นำเสนอใหม่โดยผู้วิจัย (12 ตัว)

3.5 ข้อจำกัดของแบบจำลองคุณภาพสำหรับประเมินซอฟต์แวร์โอเพนซอร์ซ

แบบจำลองคุณภาพนี้เป็นแบบจำลองที่ได้รวบรวมสมการการวัดและองค์ประกอบของตัววัดที่ศึกษาจากแบบจำลองก่อนหน้า และบางส่วนก็ได้นำเสนอใหม่ โดยสมการการวัดแบ่งออกเป็น 2 ประเภทคือ 1) สมการการวัดแบบช่วง (Category) โดยมีค่าช่วงตั้งแต่ 5 ช่วง (เช่น ตัววัด V1 OSS License) มีค่าคะแนนที่ได้คือ [20, 40, 60, 100], 4 ช่วง (เช่น ตัววัด V17 Testibility Level) มีค่าคะแนนที่ได้คือ [25, 50, 75, 100] และ 3 ช่วง (เช่น องค์ประกอบของตัววัด Minor Releases ของ V6 Maturity) มีค่าคะแนนที่ได้คือ [33, 66, 100] และ 2) สมการการวัดแบบสัดส่วน (Ratio) (เช่น ตัววัด V5 Support Activity) มีค่าคะแนนที่เป็นไปได้ตั้งแต่ (0-100] ทั้งนี้ขึ้นอยู่กับแบบจำลองคุณภาพอื่นหรือแหล่งข้อมูลอื่นที่อ้างอิงมา และความสมเหตุสมผลของตัววัดคุณภาพที่ใช้สะท้อนปัจจัยคุณภาพย่อยที่ต้องการ โดยผู้วิจัยพยายามแปลงค่าคะแนนที่ได้จากสมการการวัดของแต่ละตัววัดคุณภาพให้อยู่ในช่วง (0-100] เพื่อให้สามารถคำนวณคะแนนรวมตามวัตถุประสงค์ของแบบจำลองคุณภาพได้ แต่จะเห็นได้ว่าตัววัดที่คำนวณค่าแบบช่วงมีโอกาสที่จะให้ค่าคะแนนที่สูงหรือมีความแตกต่างกันของค่าคะแนนอย่างก้าวกระโดด ซึ่งจะส่งผลต่อคะแนนรวมของโอเพนซอร์ซมากกว่าค่าคะแนนที่ได้จากตัววัดที่คำนวณค่าแบบสัดส่วน อย่างไรก็ตามโอเพนซอร์ซทุกตัวที่ถูกนำมาเปรียบเทียบกันตามแบบจำลองคุณภาพจะได้รับผลกระทบดังกล่าวในลักษณะเดียวกัน จึงสามารถนำค่าคะแนนมาเปรียบเทียบกันได้



รูปที่ 3.2 ภาพรวมแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซ

บทที่ 4
การประเมินแบบจำลองคุณภาพ

หลังจากที่ได้พัฒนาแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซตามขั้นตอนข้างต้นแล้ว ผู้วิจัยจะทำการออกแบบสอบถามเพื่อให้ผู้มีประสบการณ์ในการเลือกใช้โอเพนซอร์ซได้ประเมินความสมเหตุสมผลของแบบจำลองคุณภาพ โดยต้องการสอบถาม 2 ประเด็น คือ 1) ความครอบคลุมของปัจจัยที่สามารถประเมินได้อย่างอัตโนมัติและมีผลต่อการคัดเลือกโอเพนซอร์ซในมุมมองของผู้ประเมิน 2) ความสมเหตุสมผลในการกำหนดตัววัดคุณภาพที่สามารถวัดค่าได้อัตโนมัติเพื่อสะท้อนถึงปัจจัยที่สนใจ

4.1 การออกแบบแบบสอบถามเพื่อประเมินแบบจำลองคุณภาพ

ในส่วนของการออกแบบแบบสอบถามนี้ ผู้วิจัยต้องการให้ผู้ประเมินได้ให้ความเห็นในทุกส่วนของแบบจำลองคุณภาพทั้งภาพรวม และรายละเอียดปลีกย่อย จึงแบ่งแบบสอบถามออกเป็น 3 ส่วนด้วยกันคือ 1) ข้อมูลทั่วไป 2) คำถามภาพรวม 3) รายละเอียดของตัววัดคุณภาพ โดยขออธิบายรายละเอียดของแต่ละส่วนดังตารางที่ 4.1 และตารางที่ 4.2 (ตัวอย่างแบบประเมินดูเพิ่มเติมได้ที่ภาคผนวก ข)

ตารางที่ 4.1 คำถามข้อมูลทั่วไป

ข้อที่	คำถาม
1	**วัตถุประสงค์:** เพื่อทราบประสบการณ์ทำงานทั้งหมดที่เกี่ยวข้องกับโอเพนซอร์ซทั้งนี้เพื่อให้น้ำหนักคำแนะนำที่ได้จากผู้ทำแบบประเมิน **คำถาม:** คุณมีประสบการณ์เกี่ยวกับประสบการณ์โอเพนซอร์ซกี่ปี
2	**วัตถุประสงค์:** เพื่อจำแนกประเภทของแหล่งที่มาของประสบการณ์ว่ามาจากที่ใดบ้าง ทั้งนี้ทำให้ผู้วิจัยทราบว่าโอเพนซอร์ซนั้นได้รับความสนใจจากแหล่งที่มาใดมากน้อยเพียงใด **คำถาม:** คุณมีประสบการณ์เกี่ยวกับโครงการโอเพนซอร์ซจากที่ใด

ตารางที่ 4.2 คำถามภาพรวมของแบบจำลองคุณภาพ

ข้อที่	คำถาม
1	**วัตถุประสงค์:** เพื่อต้องการทราบว่าระดับปัจจัยคุณภาพ (Quality Factors) ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสม สมเหตุสมผล และทำให้สามารถสะท้อนคุณภาพของซอฟต์แวร์โอเพนซอร์ซได้เพียงพอหรือไม่ **คำถาม:** คุณคิดว่าระดับ**ปัจจัยคุณภาพ (Quality Factors)** ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสม และทำให้สามารถสะท้อนคุณภาพของซอฟต์แวร์โอเพนซอร์ซได้
2	**วัตถุประสงค์:** เพื่อต้องการทราบว่าระดับปัจจัยคุณภาพ (Quality Factors) ที่ระบุในแบบจำลองคุณภาพครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้ หรือแก้ไขเพิ่มเติมให้สมบูรณ์ยิ่งขึ้น **คำถาม:** คุณคิดว่าระดับ**ปัจจัยคุณภาพ (Quality Factors)** ที่ระบุในแบบจำลองคุณภาพ ครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้
3	**วัตถุประสงค์:** เพื่อต้องการทราบว่าปัจจัยคุณภาพย่อย (Sub-quality factors) ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสมทำให้สามารถสะท้อนคุณภาพของระดับก่อนหน้าได้ และเป็นการจับคู่ที่สมเหตุสมผล **คำถาม:** คุณคิดว่าระดับ**ปัจจัยคุณภาพย่อย (Sub-quality factors)** ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสมทำให้สามารถสะท้อนคุณภาพของระดับก่อนหน้าได้
4	**วัตถุประสงค์:** เพื่อต้องการทราบว่าปัจจัยคุณภาพย่อย (Sub-quality factors) ที่ระบุในแบบจำลองคุณภาพครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้ มีคำอธิบาย และแหล่งที่มาที่ชัดเจน **คำถาม:** คุณคิดว่าระดับ**ปัจจัยคุณภาพย่อย (Sub-quality factors)** ที่ระบุในแบบจำลองคุณภาพ ครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้
5	**วัตถุประสงค์:** เพื่อต้องการทราบว่าวิธีการคำนวณโดยใช้ค่าเฉลี่ยถ่วงน้ำหนักสามารถใช้งานได้จริง สามารถสะท้อนคุณภาพซอฟต์แวร์โอเพนซอร์ซที่กำลังสนใจได้จริง และค่าคะแนนสามารถนำไปเปรียบเทียบกับซอฟต์แวร์โอเพนซอร์ซอื่น ๆ ได้ **คำถาม:** คุณคิดว่า**ค่าคะแนนที่เป็นผลลัพธ์จากแบบจำลองคุณภาพ (Overall Quality Score)** จากวิธีการคำนวณที่ระบุไว้สามารถสะท้อนคุณภาพซอฟต์แวร์โอเพนซอร์ซที่กำลังสนใจได้จริง และค่าคะแนนสามารถนำไปเปรียบเทียบกับซอฟต์แวร์โอเพนซอร์ซอื่น ๆ ได้

คำถามรายละเอียดจะถามไปที่แต่ละตัววัดคุณภาพทั้งหมด 19 ตัววัดโดยแต่ละตัววัดมีรายการคำถามย่อยดังตารางที่ 4.3

ตารางที่ 4.3 รายละเอียดของแต่ละตัววัดคุณภาพ

ข้อที่	คำถาม
1	**วัตถุประสงค์:** เพื่อต้องการทราบว่าคุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว ผู้ประเมินสามารถอ่านแล้วเข้าใจได้เป็นอย่างดี **คำถาม:** คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว
2	**วัตถุประสงค์:** เพื่อต้องการทราบว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว และผู้ประเมินมีวิธีการคำนวณที่แตกต่างจากนี้ **คำถาม:** คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว
3	**วัตถุประสงค์:** เพื่อต้องการทราบว่าแหล่งข้อมูลที่นำมาใช้คำนวณตัววัดนี้เพียงพอหรือไม่ หรือผู้ทำแบบประเมินมีแหล่งข้อมูลที่เหมาะสมกว่านี้ **คำถาม:** คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว
4	**วัตถุประสงค์:** เพื่อต้องการทราบว่าตัววัดคุณภาพนี้มีความสำคัญในการตรวจสอบคุณภาพของซอฟต์แวร์โอเพนซอร์ซมากน้อยเท่าใด ทั้งนี้หากมีความสำคัญน้อยจะพิจารณาปรับปรุงแก้ไข **คำถาม:** คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ

ทั้งนี้ผู้ประเมินจะให้คะแนนความคิดเห็นตามข้อคำถามในช่วงคะแนนตั้งแต่ 0 ถึง 4 โดยเกณฑ์การให้คะแนนจะแสดงดังตารางที่ 4.4

ตารางที่ 4.4 เกณฑ์การให้คะแนน

คะแนน	คำอธิบาย	หลักการให้คะแนน
4	เห็นด้วยอย่างยิ่ง	ผู้ประเมินเห็นด้วยกับข้อคำถามนั้น ๆ อย่างยิ่ง ว่าแบบจำลองคุณภาพมีคุณภาพตามข้อคำถามนั้น ๆ และไม่จำเป็นต้องแก้ไขใด ๆ
3	ค่อนข้างเห็นด้วย	ผู้ประเมินค่อนข้างเห็นด้วยกับข้อคำถามนั้น ๆ ว่าแบบจำลองมีคุณภาพเพียงพอ แต่ยังไม่ชัดเจน หรืออาจจะพิจารณาแก้ไขปรับปรุงเพิ่มเติม (โปรดแสดงความคิดเห็นเพิ่มเติม)
2	ค่อนข้างไม่เห็นด้วย	ผู้ประเมินค่อนข้างไม่เห็นด้วยกับข้อคำถามนั้น ๆ ว่าแบบจำลองคุณภาพมีคุณภาพตามข้อคำถาม ผู้วิจัยควรตรวจสอบและแก้ไขตามข้อคำถามนั้น ๆ (โปรดแสดงความคิดเห็นเพิ่มเติม)
1	ไม่เห็นด้วยอย่างยิ่ง	ผู้ประเมินไม่เห็นด้วยกับข้อคำถามนั้น ๆ อย่างยิ่ง ผู้วิจัยจำเป็นต้องแก้ไขตามข้อนั้น ๆ ตามคำแนะนำ (โปรดแสดงความคิดเห็นเพิ่มเติม)
0	ไม่สามารถประเมินได้	ผู้ประเมินไม่สามารถประเมิน/ตัดสินใจเนื่องจากไม่มีความรู้เรื่องตัววัดนั้น ๆ

นอกจากนี้ผู้วิจัยได้กำหนดคุณสมบัติของผู้ทำการประเมินด้วยเพื่อให้สอดคล้องกับแบบจำลองคุณภาพที่จะได้รับการประเมินจำเป็นจะต้องเป็นบุคคลที่มีประสบการณ์ในการใช้ซอฟต์แวร์โอเพนซอร์ซมากกว่า 2 ตัวขึ้นไป หรือทำงานเกี่ยวกับเกี่ยวกับโครงการโอเพนซอร์ซอย่างน้อย 1 ปี

4.2 ผลการดำเนินการประเมินด้วยแบบสอบถาม

การหาผู้ประเมินที่มีคุณสมบัติดังกล่าวข้างต้นนั้นผู้วิจัยได้ทำการประกาศหาผู้ประเมินในกลุ่มบริษัทที่ทำงาน และกลุ่มนิสิตปริญญาโทภาคนอกเวลาที่เป็นพนักงานจากบริษัทในประเทศไทย โดยมีบริษัทขนาดใหญ่ ขนาดกลาง และบริษัทสตาร์ตอัพ โดยเบื้องต้นได้สอบถามความสมัครใจ และสอบถามความรู้พื้นฐานที่เกี่ยวข้องกับโปรเจคโอเพนซอร์ซก่อนการนัดหมายครั้งต่อไปเพื่ออธิบายรายละเอียดของการประเมินด้วยแบบสอบถาม

ผู้วิจัยได้นำเสนอแบบสอบถามโดยการนัดหมายเพื่ออธิบายเพิ่มเติมให้แก่ผู้ประเมินก่อนการทำการประเมินผ่าน Google Meet และชี้แจงข้อสงสัยในทุกคำถามในแต่ละผู้ตอบแบบสอบถาม โดยผู้วิจัยได้เก็บข้อมูล 2 วิธี คือ ผ่านการสัมภาษณ์ หรือส่งแบบสอบถามพร้อมรายละเอียดและแนบวิดีโออธิบายเพิ่มเติม โดยผู้วิจัยให้น้ำหนักกับการสัมภาษณ์โดยตรงหากผู้ประเมินยินดี โดยได้มีผู้ร่วมตอบแบบสอบถามมีทั้งหมด 26 คน ดังผลลัพธ์ตามตารางที่ 4.5

ตารางที่ 4.5 ข้อมูลหน่วยทดลอง

หลักเกณฑ์	ช่วงข้อมูล	จำนวน (คน)
ประสบการณ์เกี่ยวกับโอเพนซอร์ซ	1-2 ปี	3
	> 2-3 ปี	3
	> 3-5 ปี	5
	> 5 ปี	15
ประสบการณ์ทำงาน	บริษัทกลาง/ใหญ่	21
	บริษัทสตาร์ตอัพ	9
	องค์กรพัฒนาโอเพนซอร์ซ	2
	บุคคลทั่วไป	8
ผู้ประเมินทั้งหมด		26

จากผลการประเมินทั้งสองส่วนได้แก่ 1) ผลการทดลองคะแนนภาพรวม (อ้างอิงตารางที่ 4.6) และ 2) ผลการทดลองในแต่ละตัววัดคุณภาพ (อ้างอิงตารางที่ 4.7) ทั้งสองส่วนได้คะแนนเกณฑ์มากกว่าร้อยละ 70 หรือมากกว่า 2.8 คะแนนโดยคะแนนที่ได้อยู่ในระดับสูง ผู้ประเมินเห็นด้วยอย่างมากกับแบบจำลองคุณภาพ แต่อย่างไรก็ตามผู้วิจัยได้รับข้อเสนอแนะจากผู้ประเมินที่มีประโยชน์อย่างยิ่งที่มีผลต่อการปรับปรุงแบบจำลองคุณภาพ โดยให้รายละเอียดเพิ่มเติมดังความเห็นต่อไปนี้

ตารางที่ 4.6 ผลการทดลองคะแนนภาพรวม

ข้อที่	ข้อคำถาม	คะแนนเฉลี่ย
1	คุณคิดว่าระดับ**ปัจจัยคุณภาพ (Quality Factors)** ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสม และทำให้สามารถสะท้อนคุณภาพของซอฟต์แวร์โอเพนซอร์ซได้	**3.69**

ตารางที่ 4.6 ผลการทดลองคะแนนภาพรวม (ต่อ)

ข้อที่	ข้อคำถาม	คะแนนเฉลี่ย
2	คุณคิดว่าระดับ**ปัจจัยคุณภาพ (Quality Factors)** ที่ระบุในแบบจำลองคุณภาพ ครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้	**3.72**
3	คุณคิดว่าระดับ**ปัจจัยคุณภาพย่อย (Sub-quality factors)** ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสมทำให้สามารถสะท้อนคุณภาพของระดับก่อนหน้าได้	**3.58**
4	คุณคิดว่าระดับ**ปัจจัยคุณภาพย่อย (Sub-quality factors)** ที่ระบุในแบบจำลองคุณภาพ ครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้	**3.48**
5	คุณคิดว่า**ค่าคะแนนที่เป็นผลลัพธ์จากแบบจำลองคุณภาพ (Overall Quality Score)** จากวิธีการคำนวณที่ระบุไว้สามารถสะท้อนคุณภาพซอฟต์แวร์โอเพนซอร์ซที่กำลังสนใจได้จริง และค่าคะแนนสามารถนำไปเปรียบเทียบกับซอฟต์แวร์โอเพนซอร์ซอื่น ๆ ได้	**3.69**

ตารางที่ 4.7 ผลการทดลองในแต่ละตัววัดคุณภาพ (Quality Metrics)

ข้อที่	ข้อคำถาม	คะแนนเฉลี่ย
1	OSS License	3.68
2	Size of Community	3.61
3	Q&A Volume	3.46
4	Number of Support Contributors	3.65
5	Support Activity	3.72
6	Maturity Level	3.53
7	Development Language Popularity	3.27
8	Code Documentation	3.58
9	Learning Materials	3.61
10	Cost of Ownership Reduction	3.42
11	New Features Focus	3.38

ตารางที่ 4.7 ผลการทดลองในแต่ละตัววัดคุณภาพ (Quality Metrics) (ต่อ)

ข้อที่	ข้อคำถาม	คะแนนเฉลี่ย
12	Continuing Change	3.59
13	Code Quality Level	3.70
14	Reliability Level	3.76
15	Maintainability Level	3.79
16	Security Level	3.85
17	Testability Level	3.57
18	Co-existence	3.40
19	Lack of Performance Issues Level	3.62

ผู้ประเมินได้ให้ความเห็นเพิ่มเติมเกี่ยวกับ **ตัววัดคุณภาพที่ 7 Development Language Popularity** ว่าตัววัดคุณภาพภาษาที่ได้รับความนิยมนั้นมีความเกี่ยวข้องกับภาษาของโครงการโดยตรง โดยหากโครงการนั้นได้พัฒนามาก่อนหน้าแล้ว ตัววัดคุณภาพที่เกี่ยวข้องกับภาษาจะไม่จำเป็น เพราะเป็นไปไม่ได้ที่จะเป็นภาษาอื่น ต้องเป็นภาษาเดียวกันเท่านั้น แต่กระนั้นผู้ประเมินบางส่วนได้ให้ความเห็นว่าการเลือกภาษาที่นิยมนั้นจำเป็นต่อการตัดสินใจตอนสร้างโครงการในช่วงแรก ๆ เท่านั้น เพราะจำเป็นที่จะต้องเลือกภาษาที่มีความนิยม เพื่อให้สามารถหานักพัฒนามาร่วมงานในโครงการได้ง่าย นอกจากนี้ **ตัววัดคุณภาพที่ 18 Co-existence** ผู้ประเมินได้ให้ความเห็นว่าตัววัดคุณภาพที่สะท้อนการทำงานร่วมกันกับสภาพแวดล้อมอื่นๆ หรือ Cross Platform ไม่มีความสำคัญกับโครงการที่ดำเนินการพัฒนาไปแล้ว เพราะความต้องการที่จะต้องสามารถทำงานร่วมกับอุปกรณ์ในแพลตฟอร์มใดนั้นจำเป็นต้องระบุตั้งแต่ช่วงแรกเริ่มของโครงการ ดังนั้นเพื่อตอบสนองต่อความคิดเห็นของผู้ประเมินในตัววัดคุณภาพที่ 7 และ 18 ผู้วิจัยเห็นว่าควรที่จะคงตัววัดทั้งสองนี้ไว้ในแบบจำลองคุณภาพแต่ต้องเพิ่มความต้องการเพิ่มเติมในส่วนของเครื่องมือ คือ เครื่องมือจะต้องสามารถให้ผู้ใช้นำตัววัดคุณภาพบางตัวออกได้เพื่อให้เหมาะสมกับสถานะโครงการ ซึ่งตัววัดคุณภาพดังกล่าวจะไม่รวมอยู่ในการคำนวณค่าคะแนนรวมที่เป็นผลลัพธ์จากแบบจำลองคุณภาพ (Overall Quality Score)

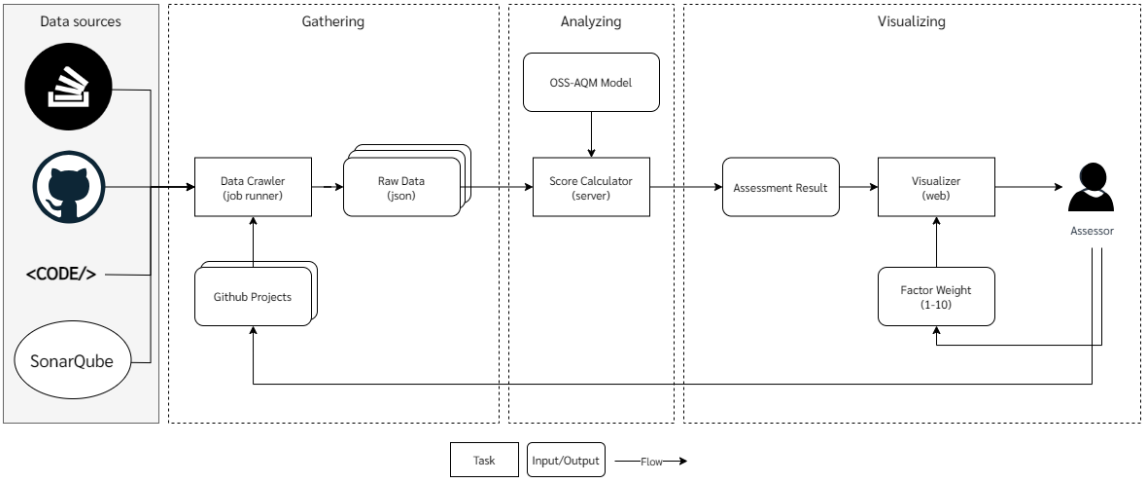
นอกจากนี้ผู้ประเมินให้ความเห็นว่า **ตัววัดคุณภาพที่ 11 New Features Focus** ซึ่งเป็นตัววัดคุณภาพที่แสดงถึงอัตราการเพิ่มขึ้นของฟีเจอร์ในโครงการโอเพนซอร์ซนั้นมีความสำคัญน้อยกว่าความสามารถของตัวโอเพนซอร์ซ กล่าวคือ ถ้าโอเพนซอร์ซที่เลือกเข้ามาไม่มีฟีเจอร์ที่ต้องการตั้งแต่แรกก็จะไม่สนใจนำมาใช้งาน อีกเหตุผลหนึ่งผู้ใช้งานบางส่วนไม่ต้องการฟีเจอร์ใหม่ ๆ เช่น

โอเพนซอร์ซที่เกี่ยวข้องกับฟังก์ชันทางคณิตศาสตร์ที่ต้องการความถูกต้อง คงที่ มากกว่าสนใจฟีเจอร์ใหม่ๆ ที่เพิ่มเข้ามา ทำให้ทราบว่าผู้ประเมินบางส่วนไม่ต้องการใช้ตัววัดคุณภาพที่ 11 ในบางสถานการณ์ ดังนั้นการเพิ่มความต้องการเครื่องมือดังกล่าวข้างต้นเพื่อให้สามารถนำตัววัดคุณภาพนี้ออกจากการคำนวณคะแนนรวมได้จะเป็นประโยชน์ต่อการใช้งานในสถานการณ์นี้เช่นกัน

บทที่ 5
การออกแบบ พัฒนา และทดสอบเครื่องมือสำหรับการประเมินโอเพนซอร์ซ

5.1 ภาพรวมการใช้งานเครื่องมือ

การออกแบบเครื่องมือเพื่อใช้สำหรับคัดเลือกโอเพนซอร์ซนั้นก็เพื่อสนับสนุนการใช้งานแบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซที่ได้รับการประเมินจากผู้ประเมินแล้วจากบทก่อนหน้า โดยผู้วิจัยได้พัฒนาโปรแกรมประยุกต์บนเว็บไซต์ที่สามารถรองรับทุกระบบปฏิบัติการ ซึ่งเครื่องมือดังกล่าวจะมีความสามารถและฟังก์ชันการทำงานในภาพรวมตามแผนภาพแนวคิดในรูปที่ 5.1



รูปที่ 5.1 ภาพรวมการทำงานของเครื่องมือวัดคุณภาพโอเพนซอร์ซ

จากรูปที่ 5.1 สามารถอธิบายขั้นตอนการใช้งานเครื่องมือดังต่อไปนี้

1. ผู้ใช้ ได้แก่ นักพัฒนา ผู้ดูแลโครงการ หัวหน้าโครงการ ผู้เชี่ยวชาญโอเพนซอร์ซ สามารถนำรายการโอเพนซอร์ซที่สนใจเข้าสู่ระบบ รวมถึงกำหนดค่าถ่วงน้ำหนักปัจจัยคุณภาพย่อยที่สนใจ
2. ระบบรับคำขอจากผู้ใช้แล้วส่งข้อมูลไปยังระบบประมวลผลกลาง โดยระบบจะทำหน้าที่ดึงข้อมูลที่จำเป็นในการประเมินผลตามแบบจำลองที่ออกแบบไว้ โดยส่งคำร้องไปยังแหล่งข้อมูลต่าง ๆ
3. แหล่งข้อมูลส่งข้อมูลกลับมายังระบบส่วนกลาง จากนั้นระบบจะทำการประมวลผลเพื่อให้ได้ข้อมูลการประเมิน และคะแนนคุณภาพของโอเพนซอร์ซแต่ละรายการตามแบบจำลองคุณภาพ

4. ระบบส่วนกลางทำการบันทึกข้อมูลการประเมินทั้งหมด รวมถึงคะแนนไปยังฐานข้อมูลของระบบ
5. ระบบส่วนกลางส่งรายงานผลการประเมิน และคะแนนโอเพนซอร์ซไปยังผู้ใช้ รวมถึงจัดลำดับโอเพนซอร์ซให้ตามผลลัพธ์จากการประเมิน

5.2 การระบุความต้องการของเครื่องมือ

หลังจากอธิบายภาพรวมการใช้งานเครื่องมือสนับสนุนแบบจำลองฯ จะสามารถจำแนกรายการความต้องการออกเป็น 6 รายการดังตารางที่ 5.1 จากนั้นจึงจะวิเคราะห์ความต้องการแบบละเอียดในหัวข้อถัดไป

ตารางที่ 5.1 รายการความต้องการของเครื่องมือ

รหัส	รายละเอียดความต้องการ	ประเภทความต้องการ	ความสำคัญ
RQ-1	ผู้ประเมินสามารถนำรายการโอเพนซอร์ซที่สนใจเข้าสู่ระบบได้	เชิงฟังก์ชัน	มาก
RQ-2	ผู้ประเมินสามารถเปรียบเทียบผลการประเมินของโอเพนซอร์ซได้	เชิงฟังก์ชัน	มาก
RQ-3	ผู้ประเมินสามารถให้ค่าถ่วงน้ำหนักในปัจจัยคุณภาพย่อยที่สนใจได้	เชิงฟังก์ชัน	ปานกลาง
RQ-4	ผู้ประเมินสามารถค้นหารายการโอเพนซอร์ซที่ได้รับการประเมินผลแล้วได้	เชิงฟังก์ชัน	มาก
RQ-5	ผู้ประเมินสามารถลบโอเพนซอร์ซที่ไม่ต้องการเปรียบเทียบได้	เชิงฟังก์ชัน	ปานกลาง
RQ-6	ผู้ประเมินสามารถลบปัจจัยคุณภาพย่อยที่ไม่สนใจออกได้	เชิงฟังก์ชัน	มาก

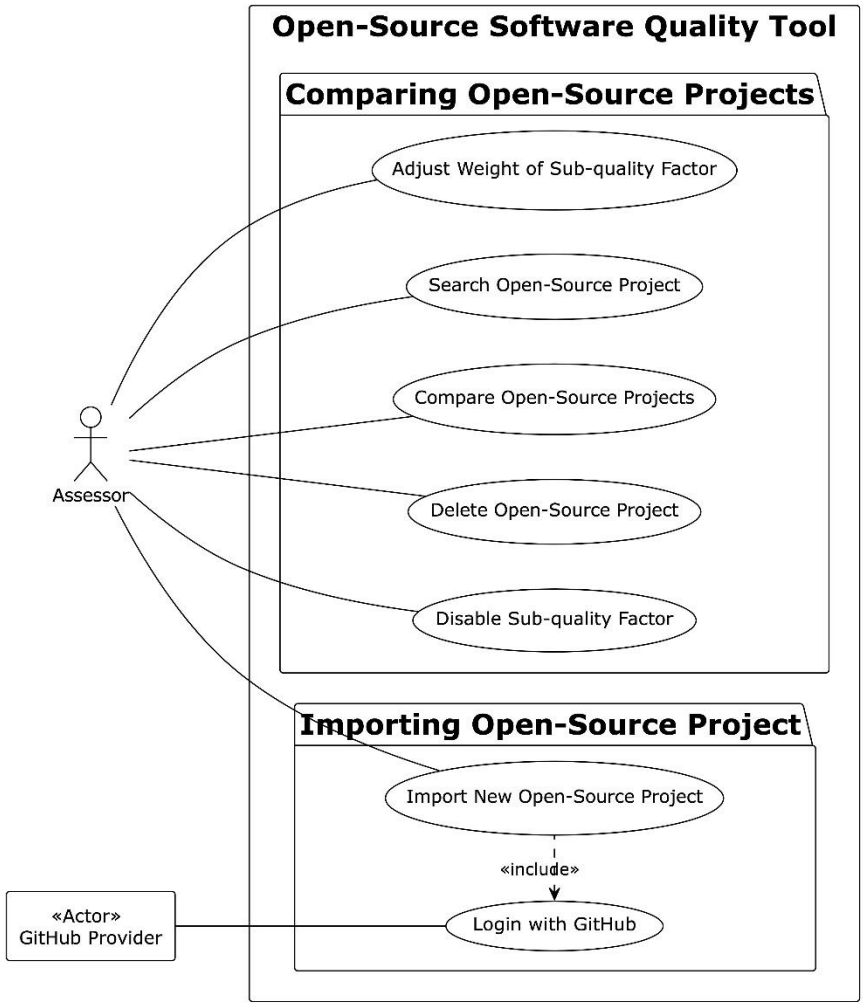
5.3 การออกแบบ การพัฒนา และการทดสอบเครื่องมือสนับสนุน

การวิเคราะห์และออกแบบเครื่องมือสนับสนุนแบบจำลองคุณภาพจะออกแบบโดยใช้แผนภาพยูเอ็มแอล (UML Diagrams) โดยจะออกแบบในภาพรวมก่อน และเจาะลึกลงในรายละเอียด ผู้วิจัยใช้แนวคิด (Top-Down Approach) ซึ่งการออกแบบในแต่ละระดับนั้นจำเป็นต้องมีรายการความต้องการเป็นข้อมูลนำเข้า และนำมาออกแบบแผนภาพยูสเคสเพื่อแสดงภาพรวมที่ผู้ใช้งานจะติดต่อประสานกับระบบ จากนั้นจึงลงรายละเอียดของแต่ละยูสเคสด้วยแผนภาพลำดับ รวมถึงออกแบบคลาสที่สามารถแสดงความสามารถของแต่ละคลาสว่ามีอะไรบ้าง จากนั้นออกแบบ

สถาปัตยกรรมระบบของเครื่องมือด้วยแผนภาพการติดตั้ง แผนภาพทั้งหมดที่นำเสนอใช้เครื่องมือแพลนต์ยูเอ็มแอล (PlantUML) ในการเขียนและสร้างแผนภาพ โดยจะขออธิบายแยกเป็นหัวข้อย่อยดังต่อไปนี้

5.3.1 แผนภาพยูสเคส

จากการวิเคราะห์ผลรายการความต้องการจะสามารถนำมาเขียนแผนภาพยูเคสเพื่อแสดงถึงฟังก์ชันงานกับบุคคลได้ตามรูปที่ 5.2



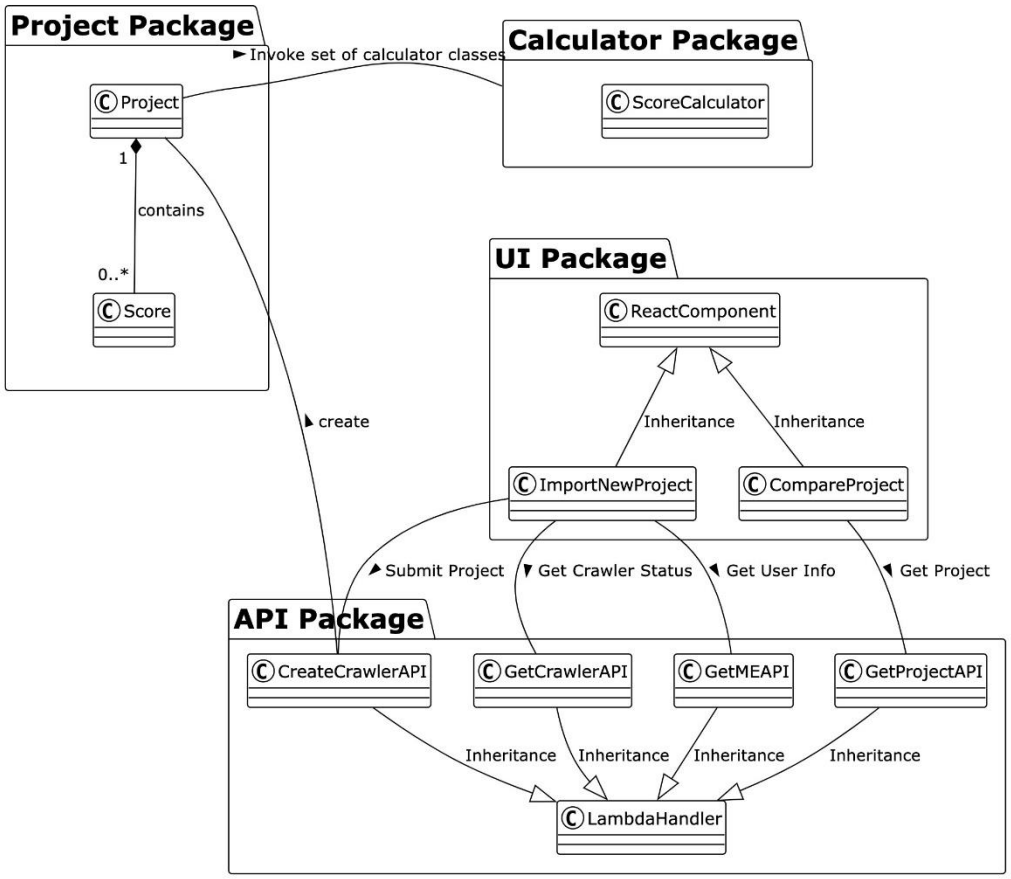
รูปที่ 5.2 แผนภาพยูเคสของเครื่องมือ

จากรูปที่ 5.2 แผนภาพยูเคสแสดงให้เห็นว่าเครื่องมือประกอบไปด้วยฟังก์ชันหลักอยู่ 2 กลุ่ม โดยกลุ่มแรก คือ ยูสเคสสำหรับการเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซที่แสดงให้เห็นว่าผู้ใช้สามารถเลือกโอเพนซอร์ซมาเปรียบเทียบ สามารถลบ แก้ไขค่าถ่วงน้ำหนักของปัจจัยคุณภาพย่อย หรือปิด

ปัจจัยคุณภาพย่อยที่ไม่ได้ใช้ได้ โดยผู้ใช้จะสามารถเป็นผู้พัฒนาซอฟต์แวร์ เจ้าของโครงการ หรือผู้ประเมินที่สนใจในรายละเอียดของคุณภาพโอเพนซอร์ซ ต่อมากลุ่มยูสเคสที่สองคือ ยูสเคสการเพิ่มโปรเจคโอเพนซอร์ซที่ต้องนำเข้าสู่เครื่องมือเพื่อนำมาประเมินคุณภาพ จะเห็นได้ว่าผู้ใช้จะต้องล็อกอินผ่านกิตฮับก่อนจะใช้งานส่วนนี้ทุกครั้ง เพื่อให้ผู้ใช้คนดังกล่าวได้ติดตามสถานะการประมวลผลจากเครื่องมือว่าโปรเจคที่ร้องขอไปมีสถานะเสร็จสิ้น หรือกำลังประมวลผลนั้นเอง

5.3.2 แผนภาพคลาส

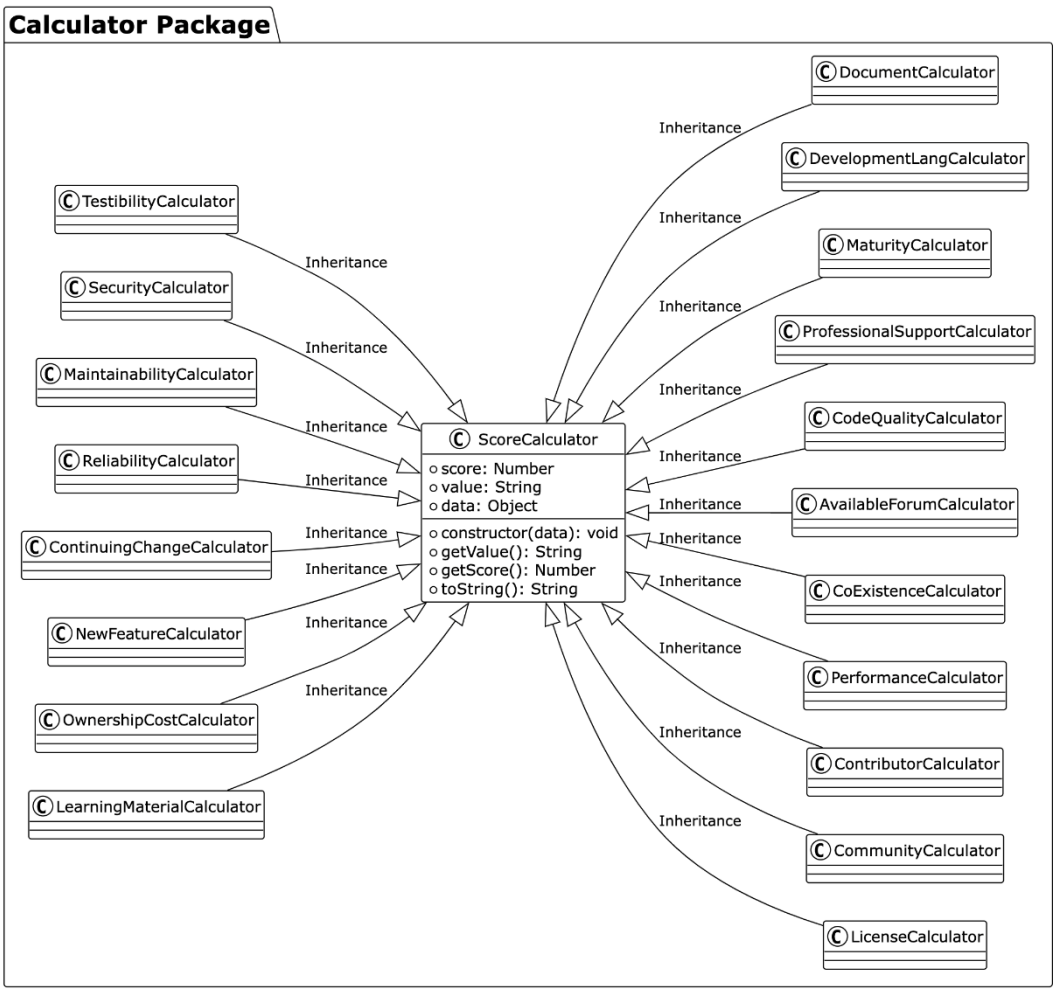
แผนภาพคลาสจะแสดงให้เห็นถึงโครงสร้างของข้อมูลและหน้าที่ของแต่ละคลาสที่ทำงานร่วมกันภายในระบบของเครื่องมือ จากรูปที่ 5.3 แสดงให้เห็นถึงความสัมพันธ์ของคลาสต่าง ๆ ซึ่งถูกแบ่งออกเป็น 4 กลุ่ม คือ 1) กลุ่มคลาสสำหรับการคำนวณคะแนนซอฟต์แวร์โอเพนซอร์ซ (Calculator Package) 2) กลุ่มคลาสโปรเจค (Project Package) 3) กลุ่มคลาสสำหรับเอพีไอ (API Package) 4) กลุ่มคลาสสำหรับเว็บแอปพลิเคชัน (UI Package) โดยมีรายละเอียดของแต่ละกลุ่มคลาสดังต่อไปนี้



รูปที่ 5.3 ภาพรวมคลาสต่าง ๆ ที่ติดต่อประสานงานกันระหว่างกลุ่ม

5.3.2.1 กลุ่มคลาสสำหรับการคำนวณคะแนนซอฟต์แวร์โอเพนซอร์ซ

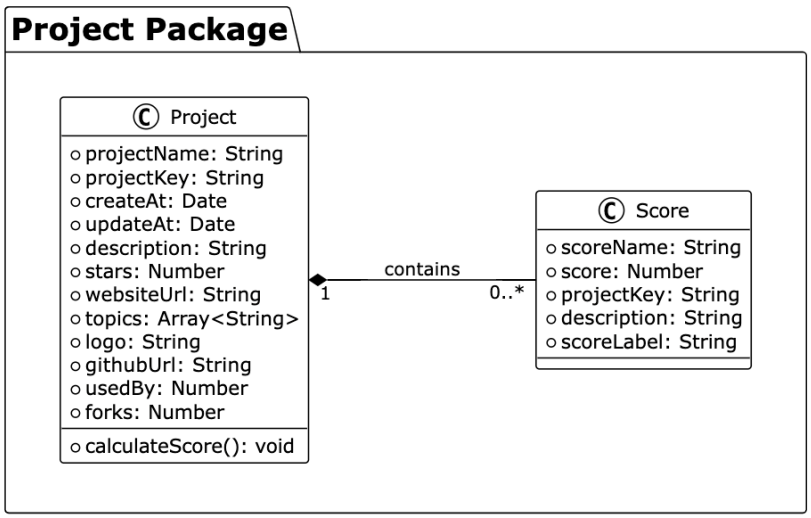
จากรูปที่ 5.4 แสดงให้เห็นกลุ่มคลาสตัวคำนวณคะแนน Calculator Package โดยมีคลาสแม่คือ คลาส ScoreCalculator ทำหน้าที่เป็นคลาสนามธรรมที่กำหนดข้อมูลและเมท็อดสำหรับการคำนวณคะแนนที่คลาสลูกทุกตัวต้องมีและสืบทอดคุณสมบัติ คลาสลูก ๆ ที่ทำหน้าที่สืบทอดนั้นจะสามารถมีหน้าที่อื่น ๆ ที่เฉพาะเจาะจงของตัวเองได้ เช่น คลาส LicenseCalculator จะต้องสามารถเก็บข้อมูลไลเซนส์จากข้อมูลดิบที่ได้มาได้ และคำนวณคะแนนออกมาตามรายละเอียดของตัววัดคุณภาพ V1 เป็นต้น โดยคลาสลูกของ ScoreCalculator จะมีทั้งหมด 19 คลาส



รูปที่ 5.4 กลุ่มคลาสสำหรับการคำนวณคะแนนซอฟต์แวร์โอเพนซอร์ซ

5.3.2.2 กลุ่มคลาสโปรเจค

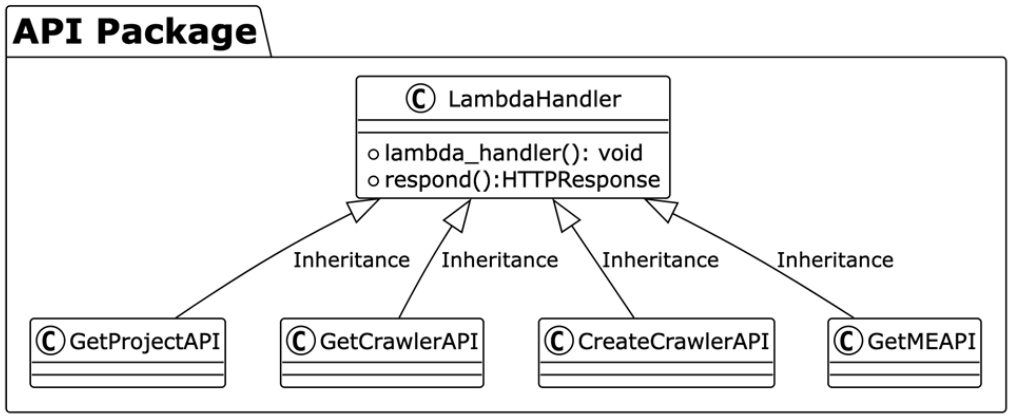
จากรูปที่ 5.5 แสดงกลุ่มคลาสโปรเจค (Project Package) โดยมีคลาส Project ทำหน้าที่ในการเก็บรายละเอียดข้อมูลของโปรเจคหนึ่ง ๆ ที่ใช้ในการคำนวณโดยมีรายละเอียด เช่น ชื่อ วันที่สร้าง จำนวนกิตฮับสตาร์ คำอธิบาย ที่อยู่เว็บไซต์ และข้อมูลอื่น ๆ ที่จำเป็น โดยคลาสโปรเจค 1 คลาส จะต้องมีคลาส Score หลายตัว (19 คลาสอ้างอิงจากโมเดลแบบจำลองฯ) ซึ่งเป็นคะแนนที่ได้จากการคำนวณจากกลุ่มคลาสตัวคำนวณคะแนน (Calculator Package) ดังนั้นคลาสดังกล่าวจึงเป็นความสัมพันธ์ 1 คลาส Project ไปหลายคลาส Score (One-to-Many)



รูปที่ 5.5 กลุ่มคลาสโปรเจคสำหรับเก็บรายละเอียดของโปรเจค

5.3.2.3 กลุ่มคลาสสำหรับเอพีไอ

กลุ่มคลาสสำหรับเอพีไอออกแบบมาเพื่อใช้ติดต่อไคลเอนต์ (Client) ที่มาจากเว็บเบราว์เซอร์ (Web Browser) โดยผ่านโปรโตคอลเอชทีทีพีเอส (HTTPS Protocol) ทั้งนี้เป็นวิธีสมัยใหม่ที่นิยมและมีความมั่นคงสูง โดยมีรายละเอียดดังรูปที่ 5.6

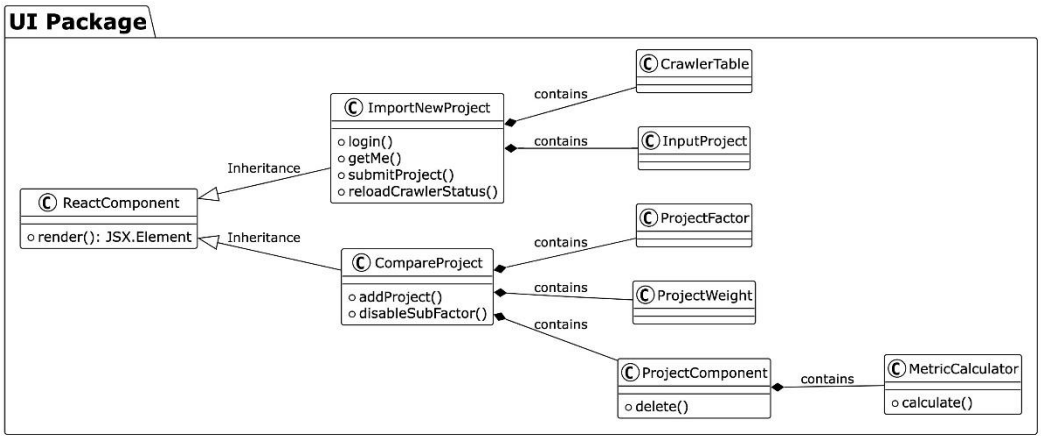


รูปที่ 5.6 กลุ่มคลาสสำหรับเรียกใช้เอพีไอ

จากรูปที่ 5.6 แสดงให้เห็นกลุ่มคลาสสำหรับเรียกใช้เอพีไอ โดยมีคลาสนามธรรม (Abstract Class) คือ คลาส LambdaHandler กำหนดฟังก์ชันที่จำเป็น และให้คลาสลูกสืบทอดคุณสมบัติซึ่งได้แก่ คลาส GetProjectAPI ใช้สำหรับค้นหาข้อมูลโปรเจคที่อยู่ในระบบ คลาส GetCrewlerAPI ใช้สำหรับดึงสถานะการเก็บข้อมูลของโปรเจคใหม่ คลาส CreateCrawlerAPI ใช้สำหรับร้องขอโปรเจคใหม่เพื่อเข้าสู่ระบบ และคลาส GetMEAPI ใช้สำหรับดึงข้อมูลผู้ใช้งานปัจจุบันเมื่อผู้ใช้ทำงานล็อกอินผ่านระบบกิตฮับ โดยกลุ่มคลาสเหล่านี้แสดงให้เห็นเอพีไอที่เครื่องมือจากฝั่งไคลเอนต์สามารถร้องขอได้

5.3.2.4 กลุ่มคลาสสำหรับเว็บแอปพลิเคชัน

จากรูปที่ 5.7 แสดงให้เห็นกลุ่มคลาสที่ใช้ในการพัฒนาส่วนต่อประสานกับผู้ใช้โดยใช้แนวคิดแบบคลาส และนำมาแสดงผลด้วย HTML ซึ่งแต่ละคลาสจะรับผิดชอบเป็นส่วนย่อย ๆ ของทั้งหน้าจอ โดยเว็บแอปพลิเคชันที่นำมาพัฒนาประกอบด้วย 2 หน้า หน้าแรกเกี่ยวข้องกับคลาส CompareProject ซึ่งทำการเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ ประกอบไปด้วยคลาส ProjectFactor ใช้แสดงรายการปัจจัยคุณภาพย่อย คลาส ProjectWeight ใช้กำหนดค่าถ่วงน้ำหนักของแต่ละปัจจัยคุณภาพย่อย คลาส MetricCalculator ใช้คำนวณคะแนนของซอฟต์แวร์โอเพนซอร์ซของโปรเจค และคลาส ProjectComponent แสดงรายละเอียดของโปรเจค ส่วนหน้าที่สองเกี่ยวข้องกับคลาส ImportNewProject สำหรับส่งคำร้องขอสร้างโปรเจคใหม่ ประกอบด้วยคลาส 2 คลาส คือคลาส InputProject ใช้สำหรับให้ผู้ใช้ป้อนยูอาร์แอลของโปรเจค และคลาส CrawlerTable ใช้สำหรับแสดงสถานะโปรเจคที่กำลังประมวลผลอยู่ในปัจจุบัน



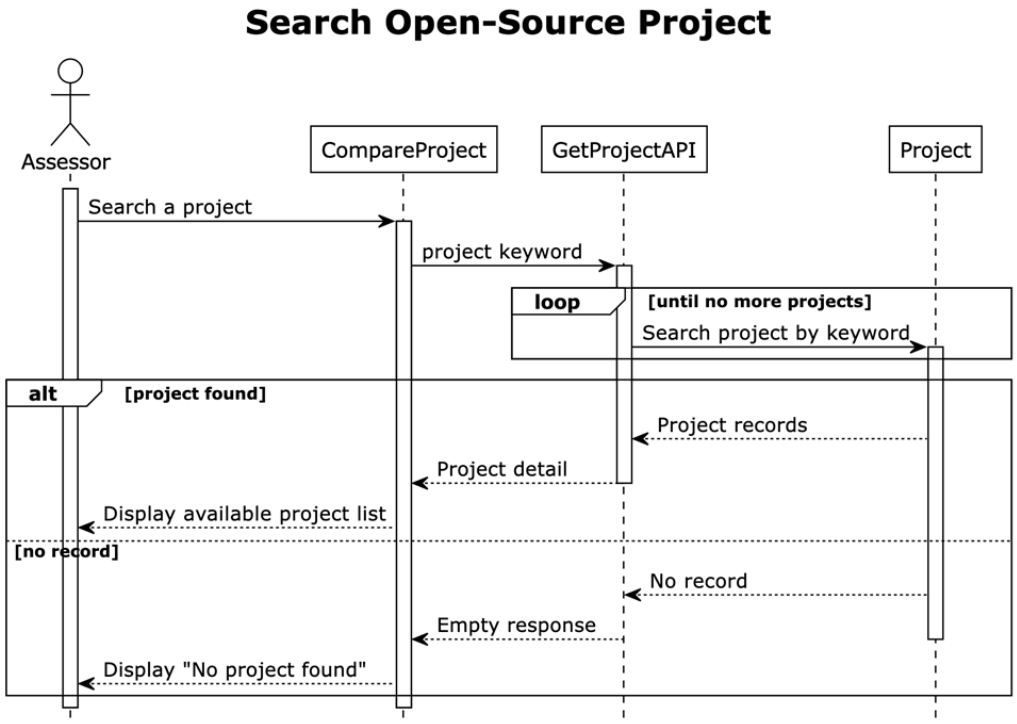
รูปที่ 5.7 กลุ่มคลาสสำหรับเว็บแอปพลิเคชัน

5.3.3 แผนภาพลำดับ

การออกแบบแผนภาพยูสเคสก่อนหน้านี้จะทำให้ทราบรายการฟีเจอร์ รวมถึงทราบจำนวนคลาสจากการออกแบบแผนภาพคลาสที่แสดงหน้าที่ความรับผิดชอบของคลาสต่าง ๆ ทำให้สามารถลงรายละเอียดชัดเจนยิ่งขึ้นในการออกแบบแผนภาพลำดับ โดยมีแผนภาพลำดับประกอบไปด้วย 6 แผนภาพโดยแสดงรายละเอียดดังต่อไปนี้

5.3.3.1 แผนภาพลำดับการค้นหาโปรเจคโอเพนซอร์ซที่สนใจ

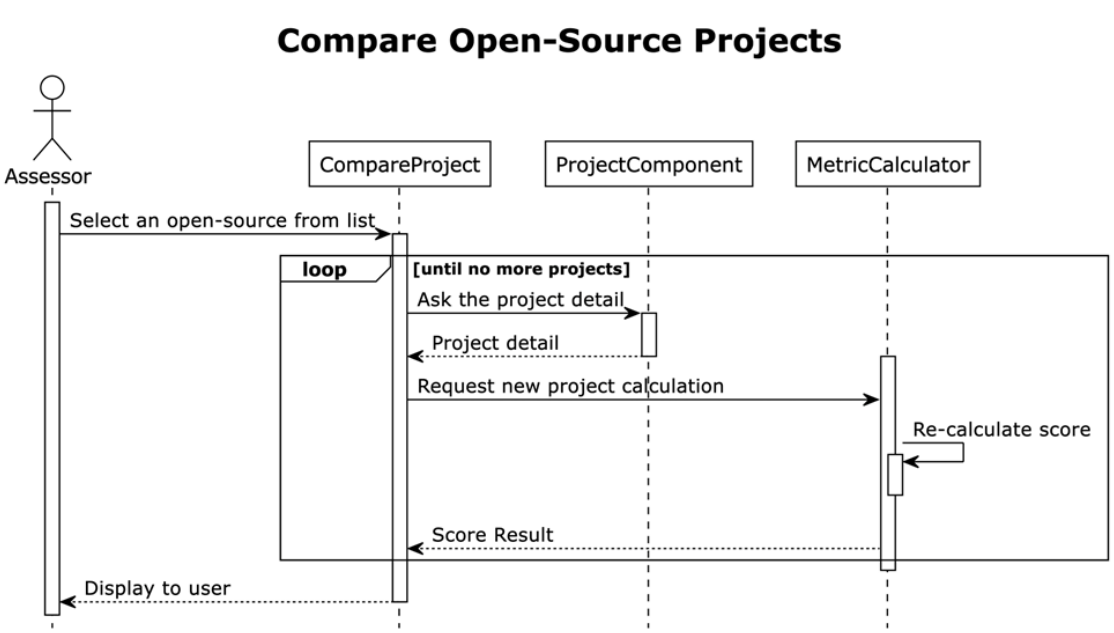
จากรูปที่ 5.8 แสดงให้เห็นลำดับการทำงานของการค้นหาโปรเจคโอเพนซอร์ซที่ผู้ประเมินให้ความสนใจ โดยเริ่มต้นจากผู้ใช้ป้อนชื่อของโปรเจคโอเพนซอร์ซ จากนั้นคลาส CompareProject จะทำการส่งคำค้นหาส่งต่อไปยังคลาส GetProjectAPI โดยจะส่งผ่านพารามิเตอร์ search ไปค้นหาในฐานข้อมูลของระบบซึ่งจะต้องไปค้นหาในตาราง Project โดยหากพบโปรเจคที่ตรงกันจะส่งข้อมูลรายละเอียดของโปรเจคกลับมา ซึ่งข้อมูลที่ได้มาจากฐานข้อมูลอาจจะมีมากกว่าหนึ่งรายการ หรือรายการเดียว โดยแสดงโลโก้ ชื่อโปรเจค และจำนวนกิตฮับสตาร์ แต่ถ้าค้นหาในฐานข้อมูลแล้วไม่พบจะส่งค่าว่างกลับมาที่คลาส GetProjectAPI และส่งต่อไปยังผู้ใช้โดยแจ้งว่า “No project found” ให้ผู้ใช้



รูปที่ 5.8 แผนภาพลำดับการค้นหาโปรเจคโอเพนซอร์ซที่สนใจ

5.3.3.2 แผนภาพลำดับการเปรียบเทียบโปรเจคโอเพนซอร์ซ

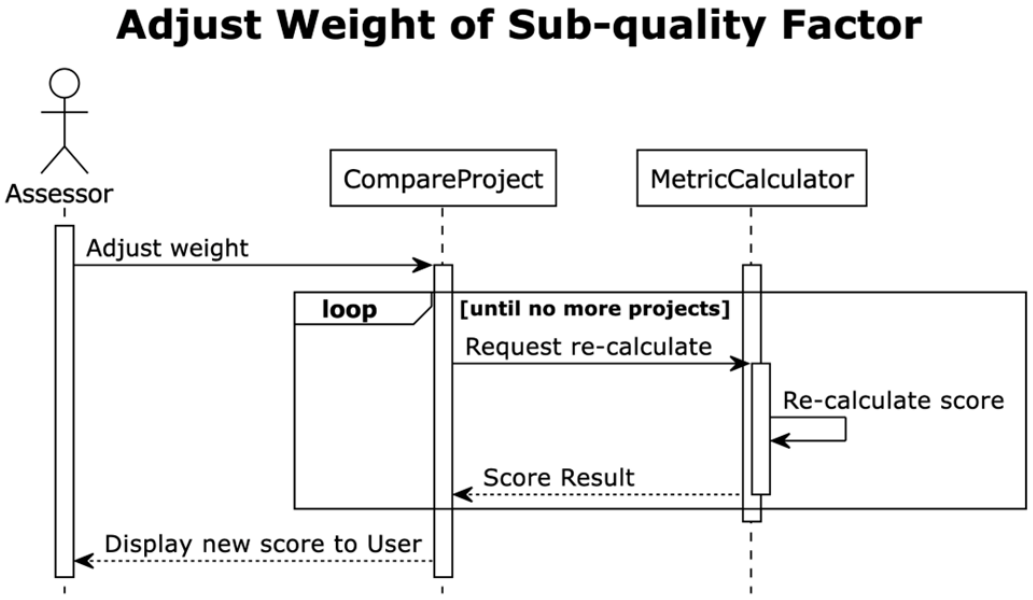
จากรูปที่ 5.9 แสดงให้เห็นลำดับการทำงานของการนำรายการโอเพนซอร์มาเปรียบเทียบ โดยต่อเนื่องจากแผนภาพลำดับการค้นหาโปรเจคโอเพนซอร์ซที่สนใจ โดยเริ่มจากผู้ใช้งานกดเลือกเพนซอร์ซจากรายการโอเพนซอร์ซที่ได้จากการค้นหาไปยังคลาส CompareProject โดยส่งไปให้คลาส ProjectComponent จากนั้นจะทำการร้องขอการคำนวณคะแนนจากคลาส MetricCalculator และนำข้อมูลคะแนนมาแสดงผลให้ผู้ใช้งาน



รูปที่ 5.9 แผนภาพลำดับการเปรียบเทียบโปรเจคโอเพนซอร์ซ

5.3.3.3 แผนภาพลำดับการแก้ไขค่าถ่วงน้ำหนักปัจจัยคุณภาพย่อย

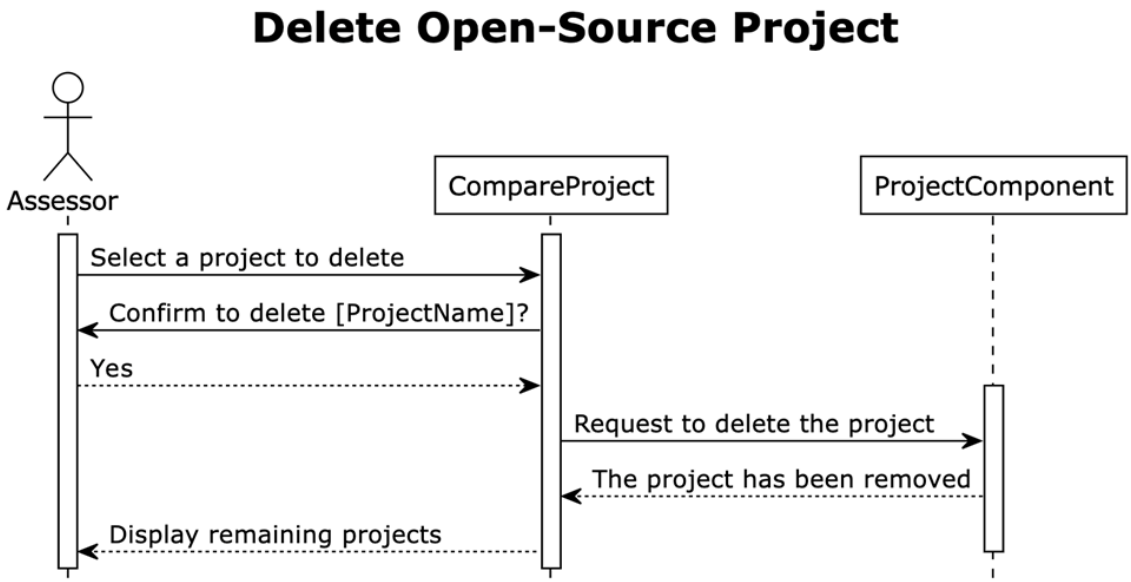
จากรูปที่ 5.10 แสดงให้เห็นลำดับการทำงานของการแก้ไขค่าถ่วงน้ำหนักสำหรับปัจจัยคุณภาพย่อยที่สนใจ โดยเริ่มจากผู้ใช้แก้ไขค่าถ่วงน้ำหนักจากหน้าจอ โดยมีค่าถ่วงน้ำหนักเริ่มต้นที่ 1 ไปจนถึง 10 และส่งข้อมูลไปที่คลาส CompareProject จากนั้นคลาสดังกล่าวจะทำการส่งคำร้องเพื่อขอให้คำนวณคะแนนใหม่ตามค่าถ่วงน้ำหนักใหม่ของทุก ๆ โปรเจคที่ผู้ใช้กำลังเปรียบเทียบอยู่ ณ ตอนนั้น โดยการขอไปที่คลาส MetricCalculator ของทุกโปรเจคจากนั้นคลาสดังกล่าวทำการคำนวณใหม่และส่งค่ากลับมาแสดงให้ผู้ใช้



รูปที่ 5.10 แผนภาพลำดับการแก้ไขค่าถ่วงน้ำหนักปัจจัยคุณภาพ

5.3.3.4 แผนภาพลำดับการลบโปรเจคโอเพนซอร์ซ

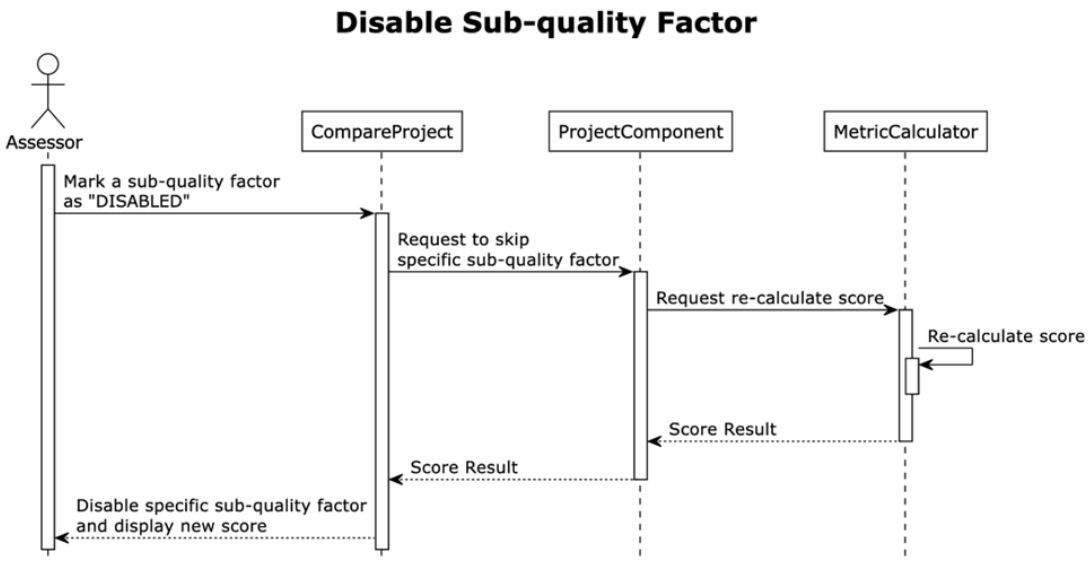
จากรูปที่ 5.11 แสดงลำดับขั้นตอนการลบโปรเจคโอเพนซอร์ซออกจากหน้าเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซในกรณีที่ผู้ประเมินไม่สนใจที่จะเปรียบเทียบอีกต่อไป โดยเริ่มจากผู้ใช้ต้องการลบโปรเจคที่มีการเพิ่มมาแล้วก่อนหน้านี้ ไปยังคลาส CompareProject โดยก่อนลบจะแสดงข้อความแจ้งเตือนว่า “Confirm to delete [ProjectName]” เมื่อผู้ใช้ยืนยันแล้วจะทำการลบโปรเจคออกจากหน้าเปรียบเทียบโอเพนซอร์ซโดยส่งคำขอลบไปยังคลาส ProjectComponent และส่งผลลัพธ์กลับมาให้ผู้ใช้



รูปที่ 5.11 แผนภาพลำดับการลบโปรเจคโอเพนซอร์ซ

5.3.3.5 แผนภาพลำดับการปิดปัจจัยคุณภาพย่อย

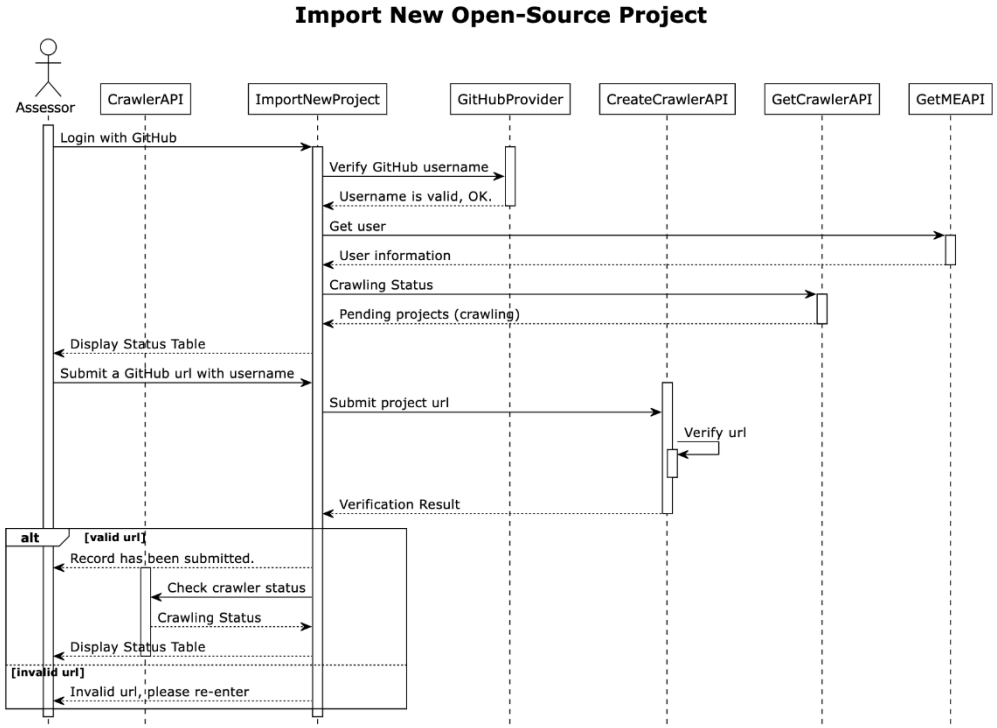
จากรูปที่ 5.12 แสดงให้เห็นลำดับการปิดปัจจัยคุณภาพย่อยที่ไม่ต้องการ โดยความต้องการนี้เกิดขึ้นมาเพื่อรองรับในกรณีที่ผู้ใช้ต้องการปิดปัจจัยคุณภาพย่อยบางตัวทำให้ปัจจัยคุณภาพย่อยนั้นไม่นำมารวมในการคำนวณคะแนนรวมของซอฟต์แวร์โอเพนซอร์ซ ทำให้คะแนนรวมสามารถสะท้อนคุณภาพของซอฟต์แวร์โอเพนซอร์ที่ตรงตามความต้องการของผู้ใช้ โดยเริ่มจากผู้ใช้ทำการเลือกปัจจัยคุณภาพย่อยที่ไม่ต้องการโดยเลือก “DISABLED” ไปยัง CompareProject จากนั้นจะส่งคำร้องไปยังคลาส ProjectComponent และส่งต่อไปยัง MetricCalculator เพื่อขอให้คำนวณคะแนนโปรเจคใหม่ และแสดงผลคะแนนรวมล่าสุดให้ผู้ใช้ทราบ



รูปที่ 5.12 แผนภาพลำดับการปิดตัววัดคุณภาพ

5.3.3.6 แผนภาพลำดับการนำเข้าโปรเจคโอเพนซอร์ซ

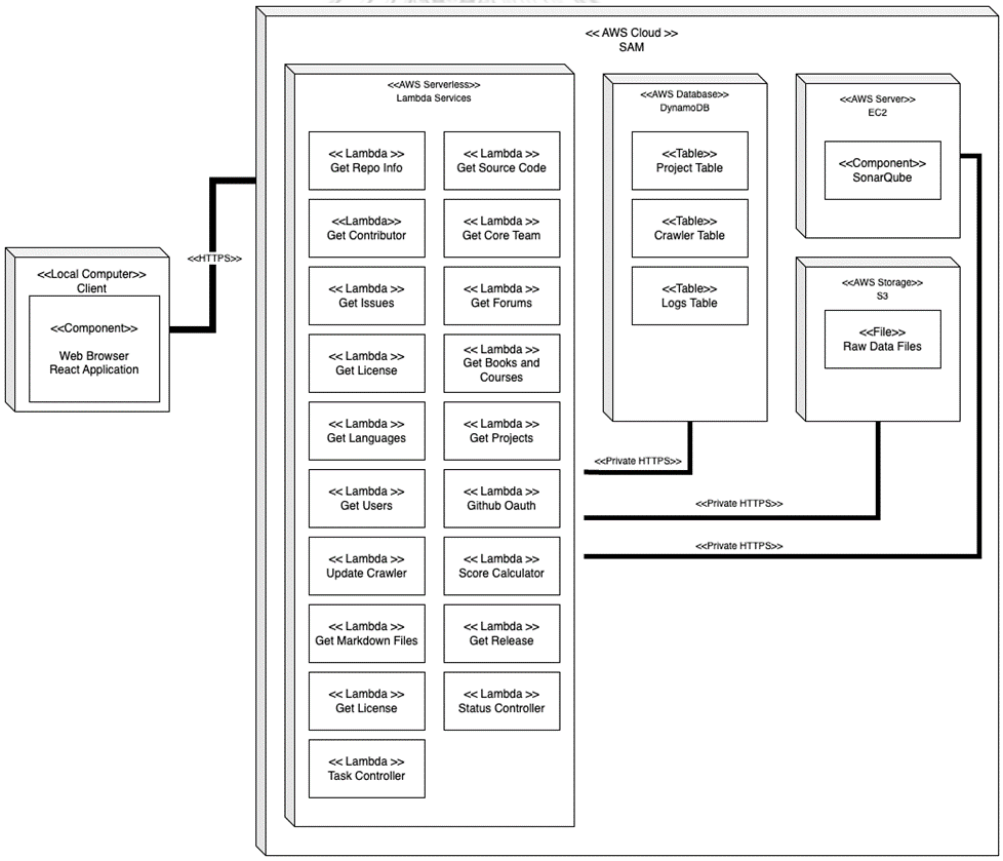
จากรูปที่ 5.13 แสดงให้เห็นลำดับการทำงานในการนำโปรเจคซอฟต์แวร์โอเพนซอร์ซใหม่เข้าระบบของเครื่องมือ หรืออาจจะเป็นโปรเจคเดิมที่เคยมีอยู่แต่ต้องการเก็บข้อมูลใหม่ล่าสุด ซึ่งอาจจะเกิดจากผู้ใช้ได้เคยทำการค้นหาโปรเจคในหน้าเปรียบเทียบโปรเจค (อ้างอิงรูปที่ 5.8 แผนภาพลำดับการค้นหาโปรเจคโอเพนซอร์ซที่สนใจ) แต่ค้นหาไม่พบ ผู้ใช้สามารถเริ่มโดยการล็อกอินเข้าสู่ระบบโดยผ่านการยืนยันตัวผ่านกิตฮับ GitHub Provider ทั้งนี้ระบบจำเป็นต้องรู้ว่าใครคือผู้ใช้ โดยดึงมาจากคลาส GetMEAPI เพื่อที่จะสามารถแสดงสถานะของโปรเจคที่เคยร้องขอมาแล้วผ่าน GetCrawlerAPI เมื่อผู้ใช้เข้าสู่ระบบสำเร็จแล้วเครื่องมือจะทำการแสดงตารางสถานะของโปรเจคที่เคยส่งคำร้อง จากนั้นผู้ใช้สามารถนำยูอาร์แอลที่อยู่โปรเจคกิตฮับมาระบุ เช่น โปรเจคแองกูลาร์ (Angular) อยู่ที่ https://github.com/angular/angular (ค้นหาเมื่อ 22 มิถุนายน 2566) แล้วกด “Submit Request” จากนั้นระบบจะส่งไปยัง CreateCrawlerAPI เพื่อตรวจสอบว่ายูอาร์แอลถูกต้องก่อนจะนำไปค้นหาจากแหล่งข้อมูลสำหรับโอเพนซอร์ซที่สามารถดึงข้อมูลอัตโนมัติตามตารางที่ 3.2 ถ้าทำรายการสำเร็จจะแจ้งว่า “Your request has been submitted” แต่ถ้ายูอาร์แอลที่นำส่งมาผิดจะแสดงผลว่า “In-valid URL, please re-enter” ให้ผู้ใช้ทราบ



รูปที่ 5.13 แผนภาพลำดับการนำเข้าโปรเจคโอเพนซอร์ซ

5.3.4 แผนภาพการติดตั้ง

ภาพรวมการติดตั้งเครื่องมือฯ บนคลาวด์เอดับเบิลยูเอส (AWS Cloud) นี้ใช้แทมเพลตแซม (SAM Template ย่อมาจาก AWS Serverless Application Model) สามารถทำให้การออกแบบโครงสร้าง การจัดวาง และการอัปโหลดโค้ดขึ้นระบบคลาวด์ทำได้ง่ายและรวดเร็ว เนื่องจากเทคโนโลยีตอนนี้ทำให้ไม่ต้องลงทุนเพื่อซื้อเครื่องเซิร์ฟเวอร์ (Server) สำหรับการติดตั้งแล้ว สามารถใช้ระบบคลาวด์แล้วจ่ายตามการใช้งานจริงได้เลย (Pay as you go) ซึ่งมีค่าใช้จ่ายที่ลดลงอย่างมากเมื่อเทียบกับวิธีดั้งเดิม จากความต้องการระบุว่าต้องพัฒนาเครื่องมือสนับสนุนฯ ผ่านโปรแกรมประยุกต์บนเว็บไซต์ ดังนั้นผู้วิจัยได้เลือกใช้วิธีไคลเอนต์เซิร์ฟเวอร์ (Client-Server) โดยสื่อสารกันผ่านโปรโตคอลเอชทีทีพีเอส (HTTPS Protocol) ระหว่างผู้ใช้กับเครื่องมือดังแสดงในแผนภาพการติดตั้ง (Deploy Diagram) ดังรูปที่ 5.14 โดยผลลัพธ์สุดท้ายเครื่องมือได้ติดตั้งบนเว็บไซต์ http://gomstory.github.io/oss-aqm สำเร็จ



รูปที่ 5.14 แผนภาพการติดตั้งเครื่องมือฯ

จากรูปที่ 5.14 แสดงให้เห็นสถาปัตยกรรม 2 กลุ่มใหญ่ กลุ่มแรกคือ ฝั่งไคลเอนต์เป็นเครื่องมือที่ติดตั้งฝั่งผู้ใช้โดยจะต้องเรียกใช้ผ่านเว็บเบราว์เซอร์ เช่น Google Chrome, Safari, Microsoft Edge เป็นต้น เครื่องมือฯ ฝั่งไคลเอนต์เป็นเว็บแอปพลิเคชันที่พัฒนาด้วยภาษาจาวาสคริปต์ (JavaScript) โดยใช้ไลบรารี React.js เขียนสไตล์ชีตด้วย SASS จัดการสถานะข้อมูลด้วย Redux State Management และใช้ Axios.js ในการติดต่อระบบผ่านเอพีไอที่ให้บริการ ส่วนกลุ่มที่สองคือ ฝั่งเซิร์ฟเวอร์ โดยมีรายละเอียดดังต่อไปนี้

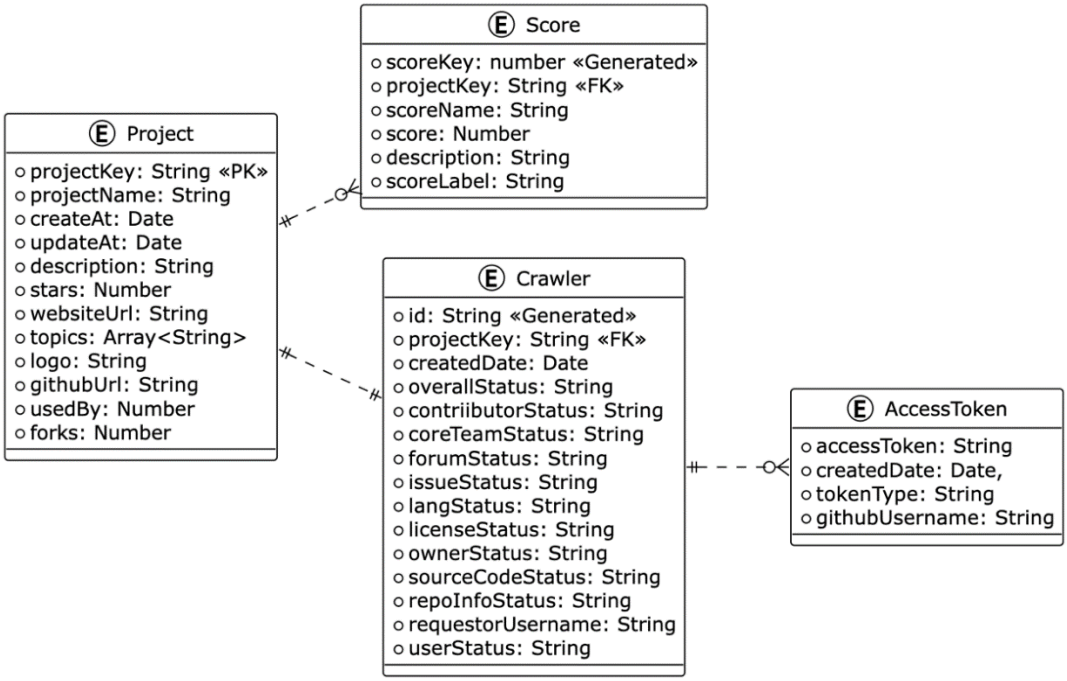
- **AWS Lambda Serverless** ใช้สำหรับให้บริการเอพีไอเพื่อเก็บข้อมูลจากแหล่งข้อมูลต่าง ๆ ลง AWS S3 และใช้คำนวณคะแนนตามตัววัดคุณภาพของแต่ละโปรเจค ผู้วิจัยพัฒนาส่วนนี้ด้วยภาษาไพทอน (Python)
- **Amazon Simple Queue Service (SQS)** ใช้สำหรับกำหนดลำดับคิวในการร้องขอนำเข้าโปรเจคโอเพนซอร์ซ ซึ่งมีความสำคัญเนื่องจากผู้ใช้อาจส่งมาหลายคำร้องขอ ทำให้ต้องจัดการลำดับเข้าออกของข้อมูล
- **Amazon DynamoDB Database (DynamoDB)** ใช้เป็นฐานข้อมูลและจัดเก็บรายละเอียดของโปรเจคโอเพนซอร์ซและคะแนนตัววัดคุณภาพที่ผ่านการประมวลผลแล้ว
- **Amazon Simple Storage Service (AWS S3)** ใช้เก็บข้อมูลดิบที่ได้มาจากแหล่งข้อมูลต่าง ๆ (อ้างอิงตารางที่ 3.2 แหล่งข้อมูลสำหรับโอเพนซอร์ซที่สามารถดึงข้อมูลอัตโนมัติ) ก่อนนำไปประมวลผล โดยจะเก็บข้อมูลในรูปแบบไฟล์ json และ text
- **Amazon Elastic Compute Cloud (EC2)** ใช้สำหรับติดตั้งโซนาร์คิวบ์ (SonarQube Server) เพื่อคัดลอก สแกนโค้ดจากโปรเจคโอเพนซอร์ซ และนำข้อมูลพื้นฐานที่ได้ไปหาค่าตัววัดคุณภาพที่เกี่ยวกับโค้ด

5.4 การพัฒนาเครื่องมือสนับสนุน

5.4.1 การกำหนดโครงสร้างฐานข้อมูล

ในส่วนนี้จะลงรายละเอียดข้อมูลที่จำเป็นต้องเก็บไว้ในฐานข้อมูลโดยวิเคราะห์มาจากแผนภาพคลาส และแผนภาพลำดับที่ได้มาจากการวิเคราะห์ก่อนหน้า เนื่องจากแผนภาพคลาสสามารถเก็บข้อมูลได้ในช่วงเวลาหนึ่งเท่านั้นทำให้จำเป็นต้องเก็บข้อมูลที่จำเป็นไว้ในฐานข้อมูลเพื่อให้เครื่องมือ และผู้ใช้เข้าถึงข้อมูลอย่างรวดเร็ว

จากรูปที่ 5.15 การกำหนดโครงสร้างฐานข้อมูล แสดงคลาสเอ็นทิตี หรือผู้วิจัยเรียกว่า ตาราง โดยมีตารางทั้งหมด 4 ตารางด้วยกัน ตารางแรกคือตาราง Project มีคอลัมน์ ProjectKey เป็นคีย์หลัก (Primary Key) ทำหน้าที่เก็บข้อมูลรายละเอียดของโปรเจคที่ได้มาจากกิตฮับ เช่น ชื่อโปรเจค ที่อยู่ รายละเอียด โลโก้ และจำนวนกิตฮับสตาร์ เป็นต้น โดยตาราง Project มีความสัมพันธ์กับตาราง Score เนื่องจากว่า 1 Project จะต้องมีคะแนนจากตัววัดคุณภาพเก็บไว้อยู่อย่างมาก 19 ตัววัด โดยให้ตาราง Score มี ProjectKey ของตาราง Project นั้นเอง เป็นคีย์รอง (Foreign Key) ถัดมาตาราง Project อาจจะอยู่ในสถานะกำลังเก็บข้อมูลและประมวลผล ซึ่งเกี่ยวข้องกับตาราง Crawler ที่ทำหน้าที่เก็บสถานะของโปรเจคที่กำลังประมวลผลโดยข้อมูลจะถูกลบหลังจากประมวลผลเสร็จสิ้น การเก็บข้อมูลในกิตฮับต้องอาศัย Access Key เพื่อดึงจากกิตฮับเอพีไอทำให้ต้องมีตาราง AccessToken เพื่อเก็บ AccessToken ที่ได้มาจากการล็อกอินของผู้ใช้ทุกครั้ง



รูปที่ 5.15 การกำหนดโครงสร้างฐานข้อมูล

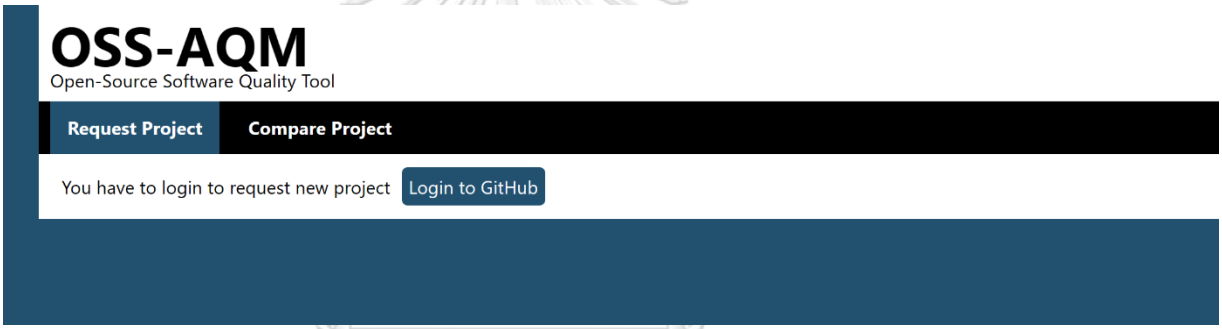
5.4.2 การพัฒนาส่วนผู้ใช้

ในขั้นตอนนี้เป็นการนำความต้องการ การออกแบบด้วยแผนภาพทั้งหมด รวมทั้งโครงสร้างฐานข้อมูล มาพัฒนาเครื่องมือทั้งส่วนต่อประสานกับผู้ใช้ และระบบในการจัดการข้อมูลของเครื่องมือ การพัฒนาเครื่องมือวัดคุณภาพโอเพนซอร์ซนี้ได้รับแรงบันดาลใจจากเว็บไซต์โอเพนฮับดอทเน็ต

(OpenHub.net) ซึ่งเป็นเว็บไซต์เปรียบเทียบข้อมูลโปรเจคโอเพนซอร์ซแบบง่ายที่นำข้อมูลมาจากกิตฮับเท่านั้น โดยผู้วิจัยนำมาออกแบบต่อยอดให้สามารถแสดงข้อมูลมากขึ้น แสดงข้อมูลมาจากหลายแหล่ง โดยส่วนต่อประสานกับผู้ใช้มีทั้งหมด 2 หน้าจอได้แก่ 1) หน้าจอสำหรับเพิ่มโปรเจคโอเพนซอร์ซใหม่ 2) หน้าจอสำหรับเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ โดยขออธิบายรายละเอียดแยกเป็น 2 หัวข้อย่อยดังต่อไปนี้

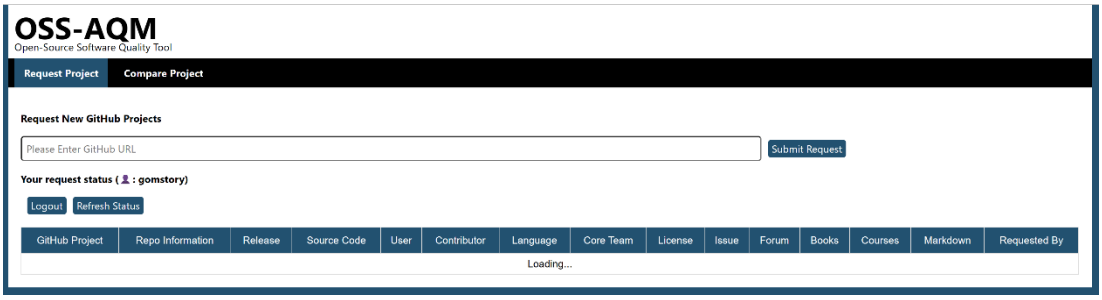
5.4.2.1 หน้าจอสำหรับเพิ่มโปรเจคโอเพนซอร์ซใหม่

เมื่อผู้ใช้มีความต้องการเพิ่มโปรเจคโอเพนซอร์ซใหม่สามารถเข้ามาเมนู Request Project โดยทำการกดแท็บซ้ายของหน้าจอดังรูปที่ 5.16 เมื่อเข้ามาครั้งแรกผู้ใช้จำเป็นจะต้องทำการล็อกอินด้วยรหัสผู้ใช้งานของกิตฮับเพื่อที่จะสามารถเข้าสู่ระบบได้ หากผู้ใช้ยังไม่เคยมีรหัสผู้ใช้งานกิตฮับจะต้องทำการลงทะเบียนก่อนใช้งาน



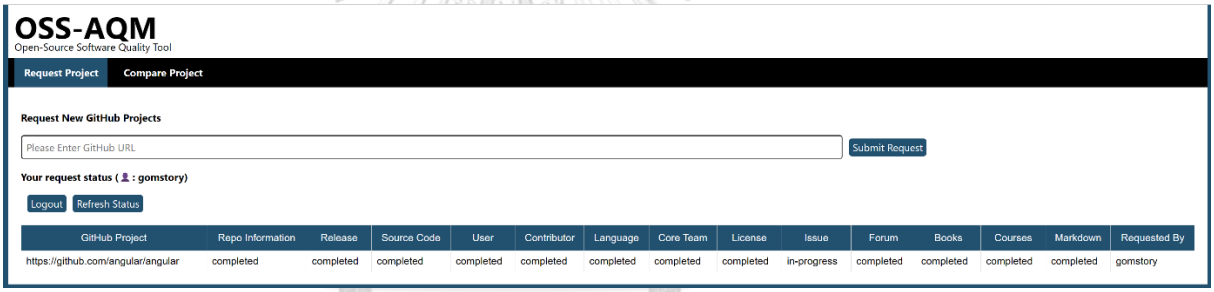
รูปที่ 5.16 หน้าจอเพิ่มโปรเจคโอเพนซอร์ซใหม่ ก่อนล็อกอินใช้งาน

เมื่อทำการยืนยันตัวตนด้วกิตฮับซึ่งในครั้งแรกทางกิตฮับจะถามสิทธ์ในการเข้าถึงข้อมูลจำเป็น เช่น ยูเซอร์เนม และอีเมล โดยผู้ใช้ต้องทำการอนุญาต จากนั้นเมื่อระบบทำการยืนยันตัวตนจะกลับเข้าสู่หน้าจอของเครื่องมือดังรูปที่ 5.17 ซึ่งผู้ใช้จะสามารถทำการเพิ่มโปรเจคกิตฮับโดยกรอกยูอาร์แอลที่อยู่โปรเจคกิตฮับที่ต้องการ ตัวอย่างเช่น เช่น โปรเจคแองกูลาร์ (Angular) อยู่ที่ https://github.com/angular/angular (ค้นหาเมื่อ 22 มิถุนายน 2566) แล้วกด “Submit Request”



รูปที่ 5.17 หน้าจอเพิ่มโปรเจคโอเพนซอร์ซใหม่ หลังล็อกอินใช้งาน

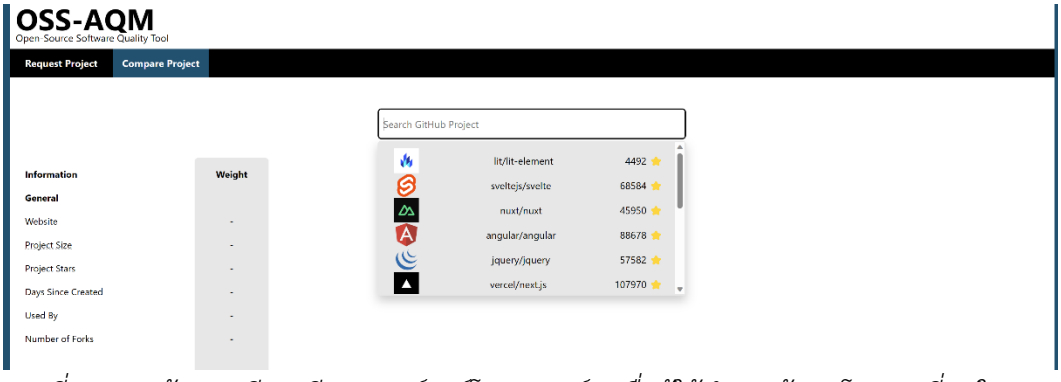
เมื่อทำการส่งคำร้องขอเสร็จสิ้น รายการที่เพิ่งส่งคำร้องขอจะแสดงในตารางดังรูปที่ 5.18 ผู้ใช้งานสามารถทราบสถานะของโปรเจคใหม่โดยการส่งครั้งแรกจะขึ้นสถานะเป็น “In-Queue” เพื่อรอดำเนินการเก็บข้อมูลสำคัญจากแหล่งข้อมูลที่เกี่ยวข้อง (อ้างอิงตารางที่ 3.2 แหล่งข้อมูลสำหรับโอเพนซอร์ซที่สามารถดึงข้อมูลอัตโนมัติ) โดยเมื่อระบบอยู่ในระหว่างสืบค้นข้อมูลจะมีสถานะ “In-Progress” และเมื่อเสร็จสิ้นจะมีสถานะ “Completed” จากโปรเจคโอเพนซอร์ซจะสามารถค้นหาในส่วนของหน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ โดยจะขออธิบายในหัวข้อต่อไป



รูปที่ 5.18 หน้าจอเพิ่มโปรเจคโอเพนซอร์ซใหม่ หลังล็อกอินใช้งาน กรณีผู้ใช้เคยส่งคำร้องขอมาแล้ว

5.4.2.2 หน้าจอสำหรับเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ

เมื่อผู้ใช้ต้องการตรวจสอบคุณภาพโอเพนซอร์ซที่สนใจสามารถทำการค้นหารายชื่อโอเพนซอร์ซโดยการกรอกชื่อ หรือพิมพ์ “[ชื่อองค์กร]/[ชื่อโปรเจค]” จากนั้นเครื่องมือจะทำการค้นหารายชื่อที่มีอยู่ในระบบและทำการแสดงผลลัพธ์การค้นหาดังรูปที่ 5.19 โดยแสดงรายละเอียดบางส่วน ได้แก่ โลโก้ ชื่อ และจำนวนกิตฮับสตาร์ของโปรเจคโอเพนซอร์ซที่ตรงกันกับคำค้นหาข้างต้น



รูปที่ 5.19 หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ เมื่อผู้ใช้ทำการค้นหาโปรเจคที่สนใจ

ผู้ใช้สามารถเลือกรายการโอเพนซอร์ซที่แสดงไว้ในรายการค้นหาโดยทำการกดเลือกที่ชื่อโอเพนซอร์ซ จากนั้นรายละเอียดของโอเพนซอร์ซจะนำมาแสดงรายละเอียดดังรูปที่ 5.20 โดยมีรายละเอียดใน 2 ส่วนได้แก่ 1) ข้อมูลทั่วไปของโปรเจคโอเพนซอร์ซ เช่น ที่อยู่เว็บไซต์ ขนาดของโปรเจค จำนวนกิตฮับสตาร์ และอื่น ๆ และ 2) รายละเอียดของแบบจำลองคุณภาพโดยมีรายการปัจจัยคุณภาพ และปัจจัยคุณภาพย่อยตามรูปที่ 3.2

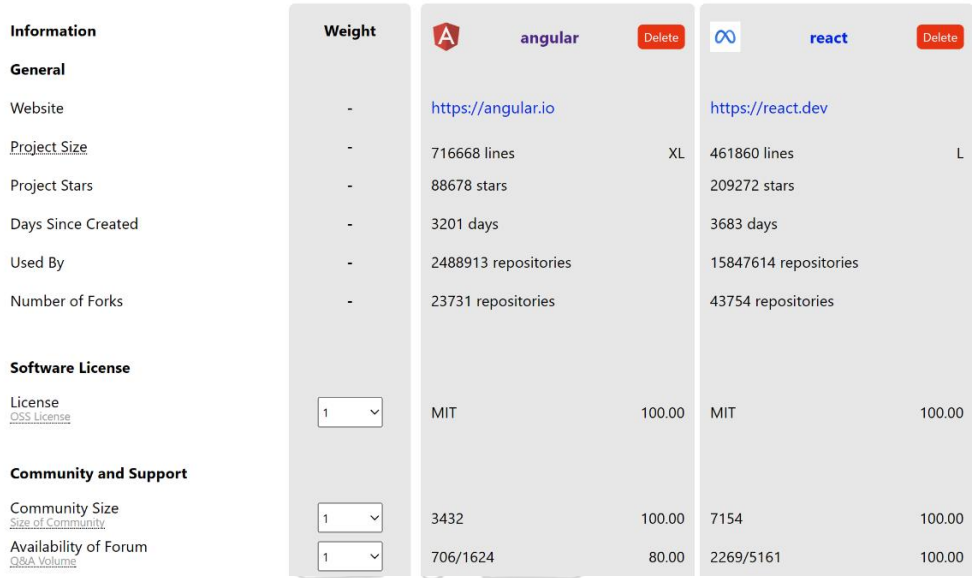
Information	Weight	angular Delete	angular.js Delete
General			
Website	-	https://angular.io	https://angularjs.org
Project Size	-	744433 lines XL	305112 lines L
Project Stars	-	84097 stars	59383 stars
Days Since Created	-	2935 days	4651 days
Used By	-	2085056 repositories	130292 repositories
Number of Forks	-	22237 repositories	28237 repositories

รูปที่ 5.20 หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ
เมื่อผู้ใช้เลือกโปรเจคที่สนใจ คือโปรเจค angular และโปรเจค angular.js

ผู้ใช้สามารถเลือกดูนิยามตัววัดคุณภาพซึ่งมีรายละเอียดของแต่ละตัววัดคุณภาพโดยการนำเมาส์ไปอยู่เหนือตัววัดคุณภาพที่สนใจดังรูปที่ 5.21 เป็นตัวอย่างรายละเอียดของตัววัดคุณภาพ V2 Size of Community

รูปที่ 5.21 หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ
เมื่อผู้ใช้ดูรายละเอียดตัววัดคุณภาพ V2 Size of Community

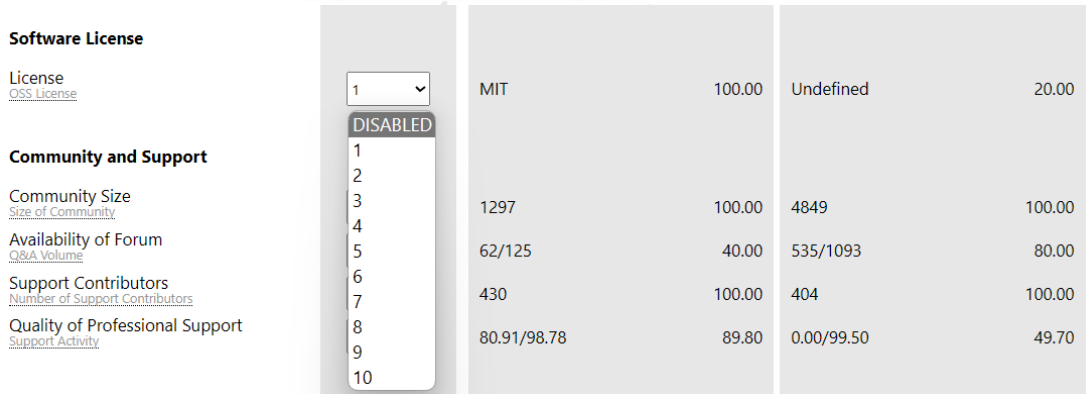
เมื่อผู้ใช้เลือกรายชื่อโปรเจคโอเพนซอร์ซมากกว่า 1 รายการขึ้นไป เครื่องมือจะแสดงรายละเอียดโอเพนซอร์ซแบบคอลัมน์ทำให้ผู้ใช้สามารถเทียบข้อมูลรายบรรทัดได้ง่าย ดังรูปที่ 5.22 แสดงรายละเอียดการเปรียบเทียบโปรเจคโอเพนซอร์ซ angular และ react



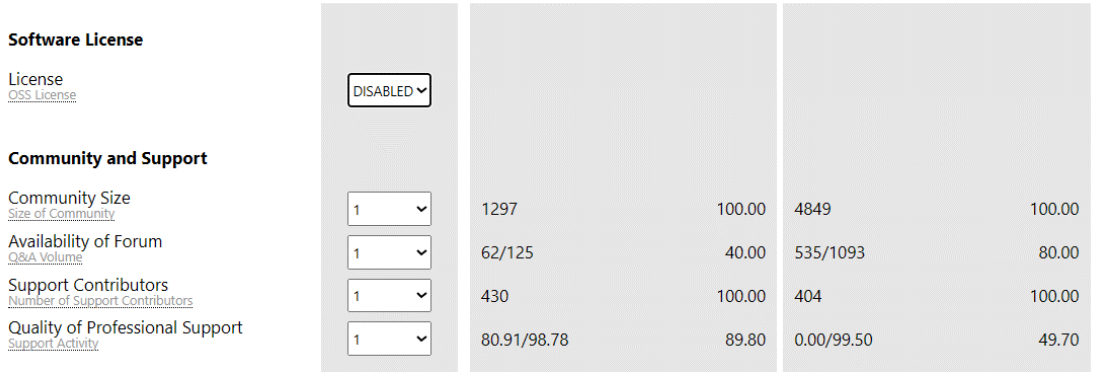
Information	Weight	angular Delete		react Delete	
General					
Website	-	https://angular.io		https://react.dev	
Project Size	-	716668 lines	XL	461860 lines	L
Project Stars	-	88678 stars		209272 stars	
Days Since Created	-	3201 days		3683 days	
Used By	-	2488913 repositories		15847614 repositories	
Number of Forks	-	23731 repositories		43754 repositories	
Software License					
License OSS License	1	MIT	100.00	MIT	100.00
Community and Support					
Community Size Size of Community	1	3432	100.00	7154	100.00
Availability of Forum Q&A Volume	1	706/1624	80.00	2269/5161	100.00

รูปที่ 5.22 หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ
เมื่อผู้ใช้นำโอเพนซอร์ซ angular และ react มาเปรียบเทียบ

เมื่อผู้ใช้เห็นว่าปัจจัยคุณภาพย่อยบางตัวไม่ได้มีความจำเป็นในสถานการณ์ของตนเองหรือในบริบทขององค์กร สามารถเลือกที่จะปิดปัจจัยคุณภาพย่อยต่าง ๆ ทั้ง 19 ตัว โดยทำการเลือกคอลัมน์น้ำหนัก “Weight” และทำการเลือก “DISABLED” ดังรูปที่ 5.23 และเมื่อกดตัวเลือกดังกล่าวจะทำให้บรรทัดของปัจจัยคุณภาพย่อยที่เลือกทำการปิดในทุกโปรเจคโอเพนซอร์ที่กำลังทำการเปรียบเทียบและเครื่องมือจะทำการคำนวณคะแนนรวมใหม่ดังแสดงในรูปที่ 5.23



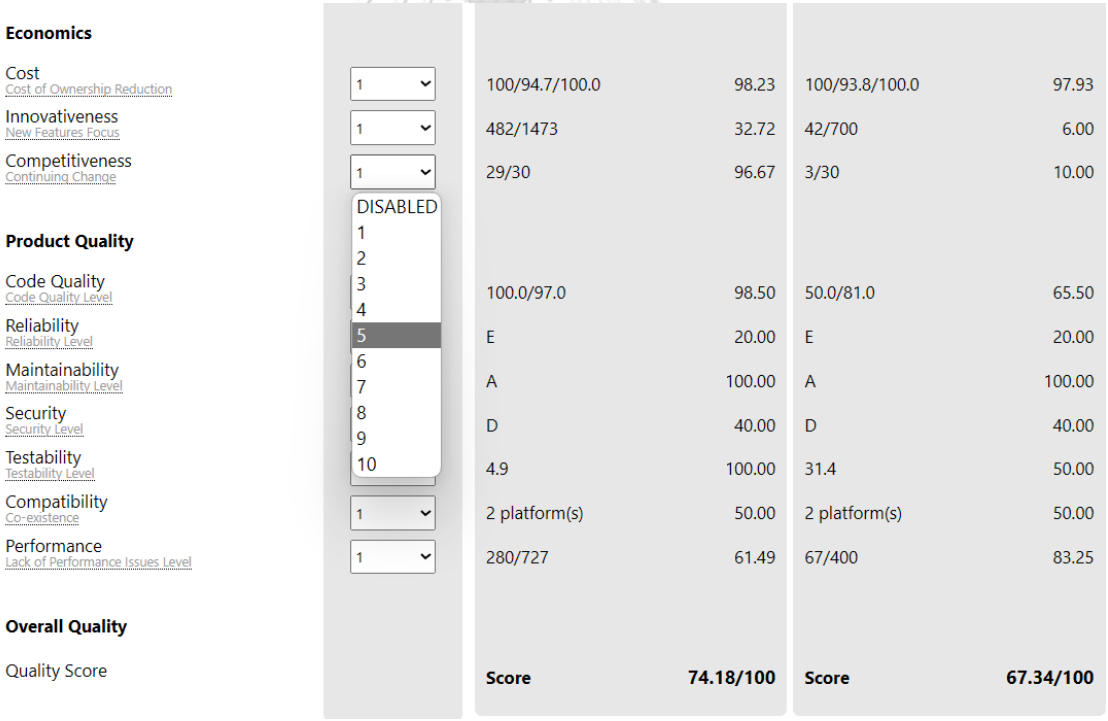
รูปที่ 5.23 หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ
เมื่อผู้ใช้ไม่ต้องการปัจจัยคุณภาพย่อยบางตัวจะสามารถเลือก "DISABLED" เพื่อปิดได้



Software License					
License OSS License	DISABLED				
Community and Support					
Community Size Size of Community	1	1297	100.00	4849	100.00
Availability of Forum Q&A Volume	1	62/125	40.00	535/1093	80.00
Support Contributors Number of Support Contributors	1	430	100.00	404	100.00
Quality of Professional Support Support Activity	1	80.91/98.78	89.80	0.00/99.50	49.70

รูปที่ 5.24 หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ เมื่อผู้ใช้งานเลือกปิดปัจจัยคุณภาพย่อย ปัจจัยดังกล่าวในทุกโปรเจคจะถูกปิด และทำการคำนวณคะแนนใหม่

ในบางสถานการณ์ผู้ใช้ต้องการปรับค่าถ่วงน้ำหนักจากเดิมเป็นค่าเริ่มต้นที่ 1 เป็นค่าอื่น สามารถทำได้โดยการไปที่คอลัมน์ “Weight” และทำการปรับค่าถ่วงน้ำหนักได้ตั้งแต่ 1-10 ตามความสำคัญในมุมมองของผู้ใช้จากนั้นเครื่องมือจะทำการคำนวณคะแนนรวมใหม่ของทุกโปรเจค จากรูปที่ 5.25 ผู้ใช้ทำการปรับค่าถ่วงน้ำหนักของปัจจัยคุณภาพย่อย Competitiveness จาก 1 เป็น 5

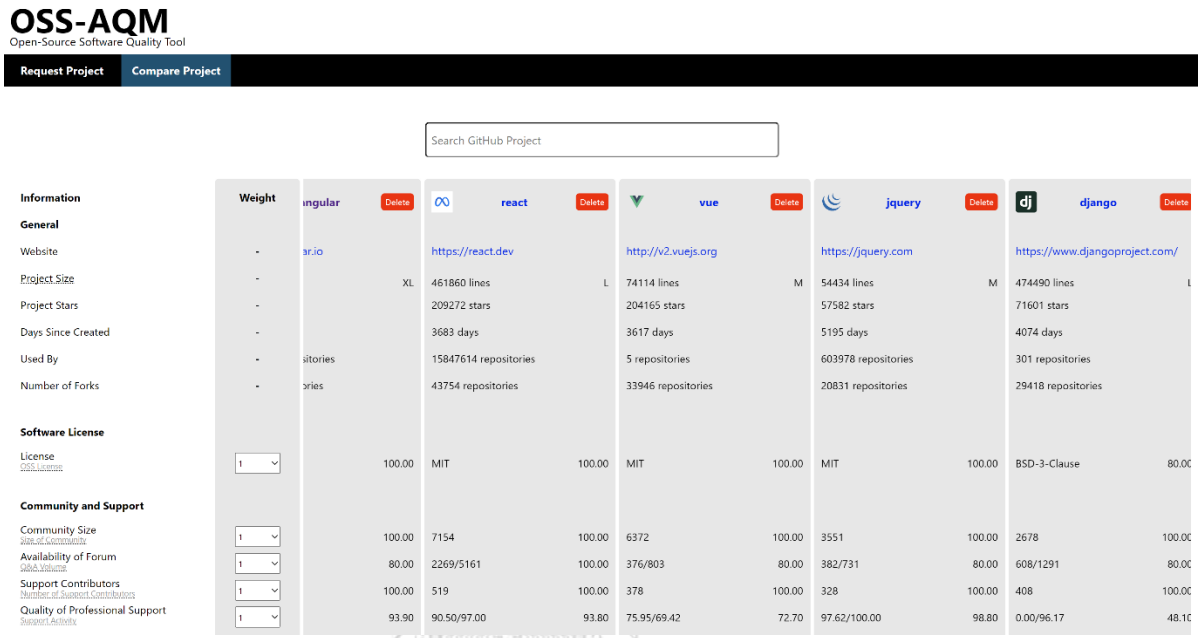


Economics					
Cost Cost of Ownership Reduction	1	100/94.7/100.0	98.23	100/93.8/100.0	97.93
Innovativeness New Features Focus	1	482/1473	32.72	42/700	6.00
Competitiveness Continuing Change	1	29/30	96.67	3/30	10.00
Product Quality					
Code Quality Code Quality Level		100.0/97.0	98.50	50.0/81.0	65.50
Reliability Reliability Level		E	20.00	E	20.00
Maintainability Maintainability Level		A	100.00	A	100.00
Security Security Level		D	40.00	D	40.00
Testability Testability Level		4.9	100.00	31.4	50.00
Compatibility Co-existence	1	2 platform(s)	50.00	2 platform(s)	50.00
Performance Lack of Performance Issues Level	1	280/727	61.49	67/400	83.25
Overall Quality					
Quality Score		**Score**	**74.18/100**	**Score**	**67.34/100**

DISABLED
1
2
3
4
5
6
7
8
9
10

รูปที่ 5.25 หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ เมื่อผู้ใช้เลือกเปลี่ยนค่าถ่วงน้ำหนักของปัจจัยคุณภาพย่อยที่สนใจ เครื่องมือจะทำการคำนวณคะแนนรวมใหม่อัตโนมัติ

เมื่อผู้ใช้ทำการเปรียบเทียบโปรเจคโอเพนซอร์ซมากกว่า 5 โปรเจค หน้าจอจะสามารถเลื่อนซ้าย หรือ ขวาได้ โดยที่คอลัมน์ข้อมูล “Information” จะยังคงอยู่ตำแหน่งเดิมทำให้ง่ายในการตรวจสอบค่าคะแนนแบบแถวในกรณีมีหลายโปรเจคดังรูปที่ 5.26 นอกจากนี้ผู้ใช้สามารถกดปุ่ม “Delete” ได้หากไม่ต้องการโปรเจคนั้น ๆ ในการเปรียบเทียบอีกต่อไป



รูปที่ 5.26 หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ เมื่อผู้ใช้งานเลือกหลายโปรเจคเพื่อเปรียบเทียบ โดยหน้าจอสามารถเลื่อนจากซ้ายไปขวาได้

5.5 การทดสอบเครื่องมือสนับสนุน

สำหรับการทดสอบเครื่องมือที่ได้พัฒนาเสร็จสิ้นนั้น ผู้วิจัยได้ดำเนินการทดสอบใน 2 มุมมอง โดยมุมมองแรกคือการตรวจสอบว่าเครื่องมือที่ได้พัฒนาตรงตามความต้องการหรือไม่ โดยใช้วิธีทวนสอบความครบถ้วนตามความต้องการของเครื่องมือที่ได้ระบุไว้ก่อนหน้า และมุมมองที่สองเป็นการทดสอบความสามารถของเครื่องมือโดยเปรียบเทียบกับวิธีอื่นในการประเมินซอฟต์แวร์โอเพนซอร์ซ โดยส่วนหลังจะกล่าวในบทที่ 6 ในส่วนของการทวนสอบความครบถ้วนตามความต้องการของเครื่องมือนั้นมีรายละเอียดดังตารางที่ 5.2

ตารางที่ 5.2 การทวนสอบความครบถ้วนตามความต้องการของเครื่องมือที่ได้ระบุ

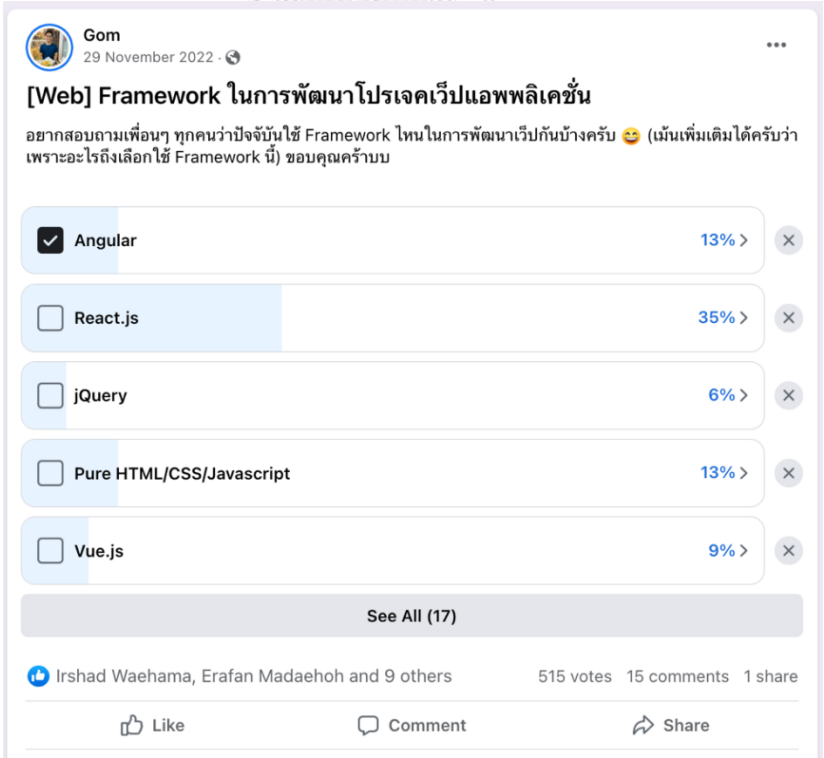
รหัส	รายละเอียดความต้องการ	การวิเคราะห์ผลการพัฒนาเครื่องมือ	พบความต้องการหรือไม่
RQ-1	ผู้ประเมินสามารถนำรายการโอเพนซอร์ซที่สนใจเข้าสู่ระบบได้	**หน้าจอ:** หน้าจอเพิ่มโปรเจคโอเพนซอร์ซใหม่ **ผลการวิเคราะห์:** ผู้ใช้ล็อกอินด้วยกิตฮับจากนั้นจะสามารถป้อนข้อมูลรายการโอเพนซอร์ซที่สนใจได้	ใช่
RQ-2	ผู้ประเมินสามารถเปรียบเทียบผลการประเมินของโอเพนซอร์ซได้	**หน้าจอ:** หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ **ผลการวิเคราะห์:** ผู้ใช้สามารถค้นหาโอเพนซอร์ซที่อยู่ในระบบและเลือกเพื่อนำมาเปรียบเทียบได้	ใช่
RQ-3	ผู้ประเมินสามารถให้ค่าถ่วงน้ำหนักในปัจจัยคุณภาพย่อยที่สนใจได้	**หน้าจอ:** หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ **ผลการวิเคราะห์:** เมื่อผู้ใช้เลือกโปรเจคที่ต้องการเปรียบเทียบแล้วจะสามารถปรับเปลี่ยนค่าถ่วงน้ำหนักของปัจจัยคุณภาพย่อยที่สนใจได้ โดยค่าเริ่มต้นอยู่ที่ 1	ใช่
RQ-4	ผู้ประเมินสามารถค้นหารายการโอเพนซอร์ซที่ได้รับการประเมินผลแล้วได้	**หน้าจอ:** หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ **ผลการวิเคราะห์:** ผู้ใช้สามารถค้นหาโอเพนซอร์ซที่สนใจได้ด้วยการพิมพ์ชื่อของโปรเจคให้ตรงกัน	ใช่
RQ-5	ผู้ประเมินสามารถลบโอเพนซอร์ซที่ไม่ต้องการเปรียบเทียบได้	**หน้าจอ:** หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ **ผลการวิเคราะห์:** ผู้ใช้สามารถลบโอเพนซอร์ซออกจากรายการเปรียบเทียบได้	ใช่
RQ-6	ผู้ประเมินสามารถลบปัจจัยคุณภาพย่อยที่ไม่สนใจออกได้	**หน้าจอ:** หน้าจอเปรียบเทียบซอฟต์แวร์โอเพนซอร์ซ **ผลการวิเคราะห์:** ผู้ใช้สามารถปิดปัจจัยคุณภาพย่อยที่ต้องการได้โดยการเลือกตัวเลือก “DISABLED” ในส่วนค่าถ่วงน้ำหนักในปัจจัยคุณภาพย่อยที่ต้องการปิด ทั้งนี้จะไม่ส่งผลต่อการคำนวณคะแนนรวมของโปรเจค	ใช่

บทที่ 6
การประเมินเครื่องมือ

บทนี้จะเป็นการนำเสนอวิธีการประเมินเครื่องมือสนับสนุนที่ได้พัฒนาเสร็จสิ้นแล้ว เพื่อตรวจสอบถึงความสามารถในการวัดคุณภาพของซอฟต์แวร์โอเพนซอร์ซโดยการนำไปเปรียบเทียบกับวิธีประเมินคุณภาพซอฟต์แวร์โอเพนซอร์ซวิธีอื่น

6.1 การเลือกซอฟต์แวร์โอเพนซอร์ซมาใช้ในการประเมิน

ผู้วิจัยเริ่มโดยการค้นหารายชื่อของโปรเจคซอฟต์แวร์โอเพนซอร์ซที่ได้รับความนิยมในปัจจุบัน ซึ่งผู้วิจัยได้ทำการสร้างแบบสอบถามแบบง่าย (Pool) ในกลุ่มเฟซบุ๊ก **สมาคมโปรแกรมเมอร์ไทย** ซึ่งมีสมาชิกจำนวนมากกว่า 160,000 คน ทั้งนี้เพื่อต้องการจำนวนคนโหวตที่มากพอ โดยผู้วิจัยตั้งโจทย์ว่า “**[Web] Framework ในการพัฒนาโปรเจคเว็บแอพพลิเคชั่น**” โดยเก็บข้อมูล 2 สัปดาห์ในช่วงวันที่ 29 พฤศจิกายน พ.ศ 2565 ถึง 11 ธันวาคม พ.ศ 2565 ตัวอย่างแสดงดังรูปที่ 6.1



รูปที่ 6.1 แบบสอบถามอย่างง่ายสำหรับโหวตโอเพนซอร์ซที่ใช้งาน

จากรูปที่ 6.1 แสดงให้เห็นผลการโหวตจากสมาชิกทั้งสิ้น 515 โหวต โดยผู้วิจัยได้เปิดให้ผู้ตอบแบบสอบถามเลือกโหวตได้อิสระ และสมาชิกสามารถเพิ่มชื่อโปรเจคโอเพนซอร์ซเพื่อเสนอขึ้นมาได้ เมื่อนำมาตัดรายชื่ออื่น ๆ ที่ไม่เกี่ยวข้องที่มีผู้ระบุเพิ่มเข้ามา เช่น ภาษา เครื่องมือ และเลือกเฉพาะโปรเจคที่อยู่ในกิตฮับจะทำให้ได้รายชื่อโปรเจคโอเพนซอร์ซดังตารางที่ 6.1 โดยผู้วิจัยได้ทำการเลือกรายการโอเพนซอร์ซ 8 รายการแรกที่สามารถใช้พัฒนาฟรอนต์เอนด์และมีผลโหวตมากที่สุดเพื่อนำไปประเมินต่อไป

ตารางที่ 6.1 ผลโหวตรายชื่อโปรเจคโอเพนซอร์ซที่ได้รับความนิยม (เรียงลำดับจากมากไปน้อย)

ลำดับ	ชื่อโปรเจค	เปอร์เซ็นผลโหวต	จำนวนคน
1	React.js	35%	180
2	Angular	13%	67
3	Vue	9%	46
4	jQuery	6%	31
5	Next.js	6%	31
6	Svelte	3%	10
7	Nuxt.js	2%	16
8	Flask	1%	5
9	LitElement	1%	5
10	Ember	1%	5
11	Laravel	1%	5
12	Django	1%	5
13	Flutter	1%	5
14	CI CodeIgniter	1%	5
ผลโหวตอื่น ๆ ที่ไม่เกี่ยวข้อง			**99**
รวมผลโหวต			**515**

6.2 การประเมินเครื่องมือด้วยการเปรียบเทียบกับการประเมินคุณภาพโอเพนซอร์ซด้วยวิธีอื่น

เมื่อได้รายชื่อโอเพนซอร์ซที่จะนำไปประเมินแล้วต่อไปเป็นการประเมินรายการโอเพนซอร์ซที่เลือกมา โดยผู้วิจัยได้ทำการศึกษางานวิจัยอื่น ๆ ที่ได้นำเสนอเครื่องมือตามตารางที่ 1.1 ทำให้ทราบว่าเครื่องมือจากงานวิจัยอื่น ๆ ไม่สามารถเข้าถึงได้ อีกทั้งในบทความวิจัยไม่ได้อธิบายรายละเอียดถึง

วิธีการคำนวณที่ชัดเจนทำให้ผู้วิจัยไม่สามารถคำนวณด้วยมือด้วยตนเอง ดังนั้นทำให้ไม่สามารถนำรายชื่อโอเพนซอร์ซที่ได้จากตารางที่ 6.1 มาเปรียบเทียบผลลัพธ์จากเครื่องมือก่อนหน้าได้ ผู้วิจัยจึงใช้วิธีการอื่น ๆ เพื่อเปรียบเทียบและหาความสัมพันธ์แทน จากการศึกษาผู้วิจัยได้เลือกใช้วิธีจากแหล่งอ้างอิงทั้งหมด 3 แหล่งด้วยกันโดยมีรายละเอียดดังต่อไปนี้

- **GitHub Stars** จำนวนสตาร์ที่สมาชิกของกิตฮับสามารถให้กับโปรเจค ทั้งนี้ขึ้นอยู่กับหลายปัจจัยที่มีผลต่อการให้สตาร์แก่โปรเจค เช่น ความชอบส่วนตัว ความนิยม การสนับสนุนจากองค์กร ความประทับใจในบริการของทีมพัฒนาซอฟต์แวร์ คุณภาพของโค้ด คุณภาพการแก้ไขข้อผิดพลาดที่รวดเร็ว และปัจจัยอื่น ๆ โดยจำนวนสตาร์สามารถนำมาจากกิตฮับโดยตรงหรือผ่านเอพีไอของกิตฮับเนื่องจากเป็นข้อมูลสาธารณะ โดยจำนวนสตาร์ของโปรเจคจะสะสมมาตั้งแต่โปรเจคได้ถูกสร้างขึ้นจนถึงปัจจุบัน ผู้วิจัยสนใจจำนวนสตาร์เนื่องจากเป็นแหล่งอ้างอิงแรกของเหล่านักพัฒนาที่ใช้เป็นเกณฑ์ในการเลือกใช้โอเพนซอร์ซเหนือปัจจัยอื่นที่สำคัญไม่แพ้กัน
- **deps.dev** เครื่องมือจากบริษัทกูเกิล (Google) ที่พัฒนาโครงการโอเพนซอร์ซอินไซต์ (Open-Source Insights) เพื่อต้องการตรวจสอบความมั่นคงของโปรเจคที่เป็นไลบรารีต่าง ๆ เช่น npm, Go, PyPl, NuGet เป็นต้น ทั้งนี้เพื่อป้องกันช่องโหว่ความมั่นคงและป้องกันความเสี่ยงจากการนำไลบรารีไปใช้งาน เมื่อไลบรารีมีความเสี่ยงแล้วถูกนำไปใช้งานในระบบอาจจะทำให้ระบบมีความเสี่ยงตามไปด้วยเช่นกัน โดยส่วนใหญ่แล้วรายการโปรเจคไลบรารีเหล่านี้เป็นโปรเจคโอเพนซอร์ซต่าง ๆ ที่อยู่ในกิตฮับนั่นเอง deps.dev ใช้วิธี OSSF Scorecard ในการตรวจสอบคุณภาพของโปรเจคโอเพนซอร์ซที่สนใจอย่างอัตโนมัติ โดยทุก ๆ เช็คลิสต์ที่สอดคล้องกับนโยบายความมั่นคงจะได้คะแนนตามที่เครื่องมือระบุไว้ ซึ่งมีค่าคะแนนอยู่ระหว่าง 0-10 คะแนนในทุกตัววัดจำนวน 12 ตัววัด [22] จากนั้นมาหาคะแนนรวมด้วยค่าเฉลี่ย (Aggregate Score) รายการตัววัดแสดงดังตารางที่ 6.2

ตารางที่ 6.2 OSSF Scorecard ของ deps.dev

ลำดับ	ชื่อตัววัด	ความหมาย
1	Binary-Artifacts	Is the project free of checked-in binaries?
2	Security-Policy	Does the project contain a security policy?
3	License	Does the project declare a license?
4	Pinned-Dependencies	Does the project declare and pin dependencies?

ตารางที่ 6.2 OSSF Scorecard ของ deps.dev (ต่อ)

ลำดับ	ชื่อตัววัด	ความหมาย
5	Maintained	Is the project at least 90 days old, and maintained?
6	CII-Best-Practices	Has the project earned an OpenSSF (formerly CII) Best Practices Badge at the passing, silver, or gold level?
7	SAST	Does the project use static code analysis tools, e.g. CodeQL, LGTM (deprecated), SonarCloud?
8	Fuzzing	Does the project use fuzzing tools, e.g. OSS-Fuzz, QuickCheck or fast-check?
9	Vulnerabilities	Does the project have unfixed vulnerabilities?
10	Signed-Release	Does the project cryptographically sign releases?
11	Token-Permission	Does the project declare GitHub workflow tokens as read only?
12	Dangerous-Workflow	Does the project avoid dangerous coding patterns in GitHub Action workflows?

- **Stack Overflow Developer Survey** เป็นแบบสำรวจที่จัดทำขึ้นทุกปีโดยทีมงาน StackOverflow.com ซึ่งทำการเก็บข้อมูลจากเหล่านักพัฒนาที่มาใช้บริการ แบบสำรวจล่าสุดที่ได้ออกมาเปิดเผยต่อสาธารณะคือแบบสำรวจประจำปี ค.ศ 2023 [23] มีผู้เข้าร่วมตอบแบบสอบถามมากกว่า 90,000 คนทั่วโลก โดยถามคำถามในประเด็นต่างๆ เช่น ภาษาคอมพิวเตอร์ที่นิยมใช้งาน ฐานข้อมูลที่ใช้ รูปแบบการทำงานของเหล่าพัฒนาทั่วโลก เครื่องมือเอไอ และอื่น ๆ แต่สิ่งที่ผู้วิจัยให้ความสนใจคือแบบสำรวจที่เกี่ยวข้องกับเทคโนโลยีของเว็บในปัจจุบัน โดยมีผู้ตอบแบบสอบถามจำนวน 71,802 คน แบ่งประเภทผู้ตอบแบบสอบถามออกเป็น 3 กลุ่มคือ นักพัฒนามืออาชีพ (Professional Developers) ผู้เริ่มต้นเรียนรู้การเขียนโค้ด (Learning to Code) และนักพัฒนาประเภทอื่น ๆ (Other Coders) ผลลัพธ์จากแบบสอบถามได้รายการเทคโนโลยีเว็บทั้งหมด 34 รายการที่มีผู้ให้ความสนใจมากที่สุดซึ่งในรายการดังกล่าวมีรายชื่อโปรเจคจากตารางที่ 6.1 รวมอยู่ด้วย ผู้วิจัยจึงใช้ผลโหวตนี้เป็นแหล่งอ้างอิงในการประเมินเครื่องมือสนับสนุนต่อไป

ผู้วิจัยใช้ลำดับของโอเพนซอร์ซ 8 โปรเจคที่ได้จากการคำนวณคะแนนโดยเครื่องมือสนับสนุนมาเปรียบเทียบกับลำดับที่ได้จากแหล่งอ้างอิง 3 แหล่งข้างต้นซึ่งเก็บข้อมูลในวันที่ 3 กรกฎาคม พ.ศ.

2566 จากนั้นจึงจะทำการวิเคราะห์สหสัมพันธ์ (Correlation) โดยใช้เทคนิค Spearman Correlation ในการวิเคราะห์ว่าผลลัพธ์จากเครื่องมือสนับสนุนฯ กับผลลัพธ์จากแหล่งข้อมูลดังกล่าวข้างต้นมีความสัมพันธ์กันแบบโมโนทอนิก (Monotonic) หรือไม่ และหากมีความสัมพันธ์อยู่ในระดับใด และเป็นไปในทิศทางเดียวกันหรือตรงข้าม ทั้งนี้จากการสังเกตผลจากการสร้าง Scatter Plot พบว่าผลลัพธ์จากเครื่องมือสนับสนุนฯ มีแนวโน้มที่จะมีความสัมพันธ์แบบวีคโมโนทอนิก (Weak-monotonic Relationship) กับผลลัพธ์จากแต่ละแหล่งอ้างอิง จึงสามารถใช้ Spearman Correlation เพื่อตรวจสอบระดับและทิศทางของความสัมพันธ์ โดยผลลัพธ์จากการเปรียบเทียบจะขออธิบายรายละเอียดใน 3 หัวข้อย่อยดังต่อไปนี้

6.2.1 การเปรียบเทียบผลลัพธ์โดยภาพรวม

ในขั้นตอนนี้เป็นการเปรียบเทียบผลจากเครื่องมือสนับสนุนและแหล่งอ้างอิงข้างต้นโดยคะแนนจากเครื่องมือนั้นพิจารณาในภาพรวมของทุกปัจจัยคุณภาพของแบบจำลองคุณภาพ และให้ค่าน้ำหนักเป็น 1 เท่ากันทุกปัจจัยคุณภาพย่อย ตารางที่ 6.3 แสดงผลการวิเคราะห์ Spearman Correlation ในภาพรวม

ตารางที่ 6.3 ผลลัพธ์การวิเคราะห์สหสัมพันธ์ เมื่อเครื่องมือพิจารณาทุกปัจจัยคุณภาพ

โปรเจค	OSS-AQM		GitHub Stars		deps.dev		Stack Overflow	
	คะแนน	ลำดับ	สตาร์	ลำดับ	คะแนน	ลำดับ	โหวต	ลำดับ
Flask	76.18	1	63462	6	7.2	2	8731	6
Svelte	75.27	2	68584	5	4.5	6	4753	7
jQuery	74.99	3	57582	7	7.9	1	15782	2
Nuxt.js	74.62	4	45950	8	6.7	3	2649	8
Angular	74.18	5	88678	4	6	4	12537	3
Next.js	72.79	6	107970	3	3.6	7	11969	4
Vue.js	69.18	7	204165	2	3.2	8	11761	5
React	67.34	8	209272	1	4.9	5	29137	1
Spearman Correlation			**-0.785**		**0.595**		**-0.523**	

จากการวิเคราะห์การจัดลำดับของเครื่องมือสนับสนุนกับแหล่งอ้างอิงสามารถสรุปผลได้ว่า GitHub Stars เป็นการให้คะแนนจากความนิยมหรือประสบการณ์ผู้ใช้ นอกจากนี้การจัดลำดับโดยวิธี Stack Overflow ก็เป็นผลโหวตจากผู้ใช้โดยตรงที่มาจากแบบสอบถาม ซึ่งเป็นข้อมูลจากความนิยมหรือประสบการณ์ผู้ใช้เช่นกัน ในขณะที่ OSS-AQM พิจารณาเพียงบางปัจจัยคุณภาพและตัววัดซึ่ง

คำนวณได้อัตโนมัติเท่านั้น จึงอาจจะไม่สามารถเปรียบเทียบผลการจัดลำดับกันได้โดยตรง อย่างไรก็ตาม การทดลองวิเคราะห์สหสัมพันธ์โดยเปรียบเทียบกับการแปลผลในตารางที่ 6.4 ได้ผลว่า คะแนนจากวิธี Stack Overflow มีความสัมพันธ์แบบโมโนทอนิกในทิศทางตรงกันข้ามในระดับปานกลางกับผลคะแนนที่ได้จาก OSS-AQM (ผลการวิเคราะห์สหสัมพันธ์เป็น -0.523) จึงสรุปได้ว่า OSS-AQM ไม่สามารถใช้ทดแทนวิธีการให้คะแนนที่อิงจากความนิยมหรือประสบการณ์ผู้ใช้จาก Stack Overflow ในขณะที่วิธี GitHub Stars มีความสัมพันธ์แบบโมโนทอนิกในทิศทางตรงกันข้ามในระดับสูงกับวิธีของ OSS-AQM (ผลการวิเคราะห์สหสัมพันธ์เป็น -0.785) (และเมื่อเปรียบเทียบเพิ่มเติมระหว่างวิธี deps.dev และวิธี GitHub Stars ผลลัพธ์คือมีความสัมพันธ์ในทิศทางตรงกันข้ามในระดับสูง (ผลการวิเคราะห์สหสัมพันธ์เป็น -0.738) อีกด้วย) การค้นพบครั้งนี้ทำให้ทราบว่าการเลือกใช้โอเพนซอร์ซด้วยกิตฮับสตาร์อย่างเดียวที่มาจากความนิยมนั้นอาจไม่เพียงพอต่อการตรวจสอบคุณภาพโอเพนซอร์ซทั้งหมด ผู้ใช้โอเพนซอร์ซควรต้องหาข้อมูลเชิงลึกเพิ่มเติมเพื่อนำมาประกอบการตัดสินใจให้เหมาะสมกับการใช้งานด้วย เช่น การป้องกันช่องโหว่ความมั่นคง คุณภาพของโค้ด และปัจจัยอื่น ๆ ประกอบ

ตารางที่ 6.4 ขนาดของความสัมพันธ์บาร์ทช์ [24]

ค่าสัมประสิทธิ์สหสัมพันธ์		
ค่าสหสัมพันธ์ทางบวก	**ค่าสหสัมพันธ์ทางลบ**	**ความหมาย**
0.81 ถึง 1.0	-0.81 ถึง -1.0	สูงมาก
0.61 ถึง 0.80	-0.61 ถึง -0.80	สูง
0.41 ถึง 0.60	-0.41 ถึง -0.60	ปานกลาง
0.21 ถึง 0.40	-0.21 ถึง -0.40	ต่ำ
0.0 ถึง 0.20	-0.0 ถึง -0.20	ต่ำมาก

นอกจากนี้วิธีการจัดลำดับของ deps.dev เป็นการให้คะแนนจากตัววัดที่วัดค่าได้อัตโนมัติโดยเน้นที่เรื่องความมั่นคง แต่มีปัจจัยด้านไลเซนส์และการบำรุงรักษาซึ่งสามารถใช้สะท้อนเรื่องความมั่นคงได้เช่นกัน ผลการวิเคราะห์สหสัมพันธ์ได้ว่า คะแนนจากวิธีนี้มีความสัมพันธ์แบบโมโนทอนิกในระดับปานกลางในเชิงบวกกับผลคะแนนที่ได้จาก OSS-AQM (ผลการวิเคราะห์สหสัมพันธ์เป็น 0.595) จึงสรุปได้ว่าผลการวัดค่อนข้างไปในทิศทางเดียวกัน แต่เนื่องจาก deps.dev เน้นเรื่องความมั่นคง ในขณะที่ OSS-AQM พิจารณาปัจจัยคุณภาพหลายด้าน ผลคะแนนจาก OSS-AQM จึงได้รับอิทธิพลจากปัจจัยอื่นที่ deps.dev ไม่พิจารณา จึงทำให้ความสัมพันธ์อยู่เพียงระดับปานกลางเท่านั้น จากการตรวจสอบตัววัดคุณภาพด้วยวิธี deps.dev นั้นมีตัววัดทั้งหมด 12 ตัววัด ถึงแม้เกณฑ์การให้คะแนน

และวิธีการคำนวณคะแนนจะแตกต่างกัน (deps.dev มีคะแนนรวม 0-10 เทียบกับ OSS-AQM ที่มีคะแนนรวมที่ (0-100] แต่ก็มีความคล้ายกันในระดับตัววัดคุณภาพ เช่น 1) ตัววัด Maintained คล้ายกับตัววัดคุณภาพ V5 Support Activity, V6 Maturity Level, V11 New Features Focus และ V12 Continuing Change 2) ตัววัด License คล้ายกับตัววัดคุณภาพ V1 OSS License 3) ตัววัด Vulnerability คล้ายกับตัววัดคุณภาพ V16 Security Level

6.2.2 การเปรียบเทียบเมื่อเครื่องมือพิจารณาเฉพาะปัจจัยคุณภาพที่สนใจ

ในขั้นตอนนี้เป็นการตรวจสอบความสัมพันธ์กับแหล่งข้อมูลข้างต้นโดยเครื่องมือพิจารณาเฉพาะบางปัจจัยคุณภาพของแบบจำลองคุณภาพ ผลลัพธ์จากตารางที่ 6.3 ทำให้ทราบว่าการเปรียบเทียบผลลัพธ์จากเครื่องมือสนับสนุนกับวิธี GitHub Stars และ Stack Overflow ยังมีความไม่สอดคล้องกันในภาพรวมทำให้ผู้วิจัยสนใจปัจจัยคุณภาพบางปัจจัยเพื่อต้องการตรวจสอบความสัมพันธ์เพิ่มเติมโดยเลือกปัจจัยที่มีส่วนคล้ายกับทั้ง 3 วิธีข้างต้นและทำการตัดปัจจัยคุณภาพอื่น ๆ ออกไป

- **ปัจจัยคุณภาพ Software License กับ Community and Support** เป็นปัจจัยคุณภาพที่มีความเกี่ยวข้องกับผู้ใช้ ความนิยม และแหล่งชุมชนพัฒนาโดยตรง โดยมีปัจจัยคุณภาพย่อยต่าง ๆ ได้แก่ License, Community Size, Availability of Forum, Support Contributors และ Quality of Professional Support ผลลัพธ์แสดงในตารางที่ 6.5

ตารางที่ 6.5 ผลลัพธ์การวิเคราะห์สหสัมพันธ์ เมื่อเครื่องมือพิจารณาเฉพาะปัจจัยคุณภาพ Software License และ Community and Support

โปรเจค	OSS-AQM		GitHub Stars		deps.dev		Stack Overflow	
	คะแนน	ลำดับ	สตาร์	ลำดับ	คะแนน	ลำดับ	โหวต	ลำดับ
Flask	87.92	6	63462	6	7.2	2	8731	6
Svelte	86.22	8	68584	5	4.5	6	4753	7
jQuery	95.76	2	57582	7	7.9	1	15782	2
Nuxt.js	87.48	7	45950	8	6.7	3	2649	8
Angular	94.94	3	88678	4	6	4	12537	3
Next.js	94.22	4	107970	3	3.6	7	11969	4
Vue.js	90.54	5	204165	2	3.2	8	11761	5
React	98.76	1	209272	1	4.9	5	29137	1
Spearman Correlation			**0.452**		**0.143**		**0.976**	

จากตารางที่ 6.5 ผลการวิเคราะห์สหสัมพันธ์ได้ว่า คะแนนจากวิธี OSS-AQM มีความสัมพันธ์แบบโมโนทอนิกในเชิงบวกกับทุกวิธีจากแหล่งข้อมูลดังกล่าวข้างต้น โดยเฉพาะวิธี Stack Overflow มีความสัมพันธ์ในระดับที่สูงมาก (ผลการวิเคราะห์สหสัมพันธ์เป็น 0.976) รองลงมาด้วยวิธี GitHub Stars มีความสัมพันธ์ในระดับปานกลาง (ผลการวิเคราะห์สหสัมพันธ์เป็น 0.452) จึงสรุปได้ว่าผลการวัดค่อนข้างไปในทิศทางเดียวกันเมื่อเลือกเฉพาะปัจจัยคุณภาพที่เกี่ยวข้องกับผู้ใช้และแหล่งชุมชนพัฒนา ยกเว้นวิธี deps.dev ที่มีความสัมพันธ์ในทิศทางเดียวกันในระดับต่ำมาก (ผลการวิเคราะห์สหสัมพันธ์เป็น 0.143) อันเนื่องมาจากวิธี deps.dev แม้จะพิจารณาด้านไลเซนส์และการบำรุงรักษาโดยชุมชนพัฒนาอยู่บ้าง แต่เน้นเรื่องความมั่นคงที่ตรวจสอบได้จากโค้ดมากกว่า

- **ปัจจัยคุณภาพ Operational Software Characteristics** เป็นปัจจัยคุณภาพที่เน้นคุณลักษณะของซอฟต์แวร์โอเพนซอร์ซโดยมีปัจจัยคุณภาพย่อย คือ Maturity, Development Language, Documentation และ Book/Online ผลลัพธ์แสดงดังตารางที่ 6.6

ตารางที่ 6.6 ผลลัพธ์การวิเคราะห์สหสัมพันธ์ เมื่อเครื่องมือพิจารณาเฉพาะปัจจัยคุณภาพ Operational Software Characteristics

โปรเจค	OSS-AQM		GitHub Stars		deps.dev		Stack Overflow	
	คะแนน	ลำดับ	สตาร์	ลำดับ	คะแนน	ลำดับ	โหวต	ลำดับ
Flask	69.62	2	63462	6	7.2	2	8731	6
Svelte	68.61	3	68584	5	4.5	6	4753	7
jQuery	72.11	1	57582	7	7.9	1	15782	2
Nuxt.js	61.33	7	45950	8	6.7	3	2649	8
Angular	59.27	8	88678	4	6	4	12537	3
Next.js	63.64	6	107970	3	3.6	7	11969	4
Vue.js	63.81	5	204165	2	3.2	8	11761	5
React	65.73	4	209272	1	4.9	5	29137	1
Spearman Correlation			**-0.190**		**0.380**		**0.143**	

ผลการวิเคราะห์สหสัมพันธ์พบว่า คะแนนจากวิธี OSS-AQM มีความสัมพันธ์แบบโมโนทอนิกในทิศทางเดียวกันกับวิธี deps.dev ที่ระดับต่ำ (ผลการวิเคราะห์สหสัมพันธ์เป็น 0.380) เนื่องจากทั้งสองวิธีนี้ใช้วิธีการประเมินแบบอัตโนมัติจากโค้ดที่คล้ายกัน โดยจากการวิเคราะห์

รายละเอียดเพิ่มเติมพบว่าปัจจัยคุณภาพย่อย Matrurity มีตัววัด Minor Releases โดยนับจำนวน Minor Release มาคำนวณคะแนนโดยจำนวนเวอร์ชันย่อยเกิดจากการที่ซอฟต์แวร์ทำการแก้ไขข้อผิดพลาด และอุดช่องโหว่ความมั่นคงตลอดระยะเวลา 1 ปี ทำให้คุณภาพของซอฟต์แวร์ดีขึ้นเรื่อย ๆ ซึ่งตรงตามความต้องการของวิธี deps.dev ที่เน้นเรื่องความมั่นคงและป้องกันช่องโหว่ อีกทั้งยังสอดคล้องกับตัววัดคุณภาพ Age ที่เมื่อระยะเวลาผ่านไปซอฟต์แวร์จะมีวุฒิภาวะที่ดีขึ้นตามกาลเวลา นอกจากนี้มีวิธีของ Stack Overflow ที่มีความสัมพันธ์แบบโมโนทอนิกในทิศทางเดียวกันในระดับต่ำมาก (ผลการวิเคราะห์สหสัมพันธ์เป็น 0.143) ซึ่งผู้วิจัยคิดว่ามีปัจจัยคุณภาพย่อย Development Language ที่อ้างอิงจากความนิยมภาษาคอมพิวเตอร์ในปัจจุบันมาเกี่ยวข้อง และปัจจัยคุณภาพย่อย Book/Online ที่ช่วยเพิ่มจำนวนผู้ใช้ให้สนใจในโอเพนซอร์ซจากหนังสือและคอร์สเรียนที่น่าสนใจ วิธีสุดท้ายคือ GitHub Stars ที่มีความสัมพันธ์แบบโมโนทอนิกในทิศทางตรงกันข้ามในระดับต่ำมาก (ผลการวิเคราะห์สหสัมพันธ์เป็น -0.190) จึงสรุปว่าวิธี OSS-AQM ตามปัจจัยคุณภาพ Operational Software Characteristics ไม่มีความเกี่ยวข้องกันกับวิธีการ Github Stars

- **ปัจจัยคุณภาพ Economics** เป็นปัจจัยคุณภาพที่เน้นที่ผลประโยชน์ขององค์กรที่นำโอเพนซอร์ซไปใช้งาน ประกอบด้วยปัจจัยคุณภาพย่อย คือ Cost, Innovativeness และ Competitiveness ผลลัพธ์แสดงดังตารางที่ 6.7

ตารางที่ 6.7 ผลลัพธ์การวิเคราะห์สหสัมพันธ์ เมื่อเครื่องมือพิจารณาเฉพาะปัจจัยคุณภาพ Economics

โปรเจค	OSS-AQM		GitHub Stars		deps.dev		Stack Overflow	
	คะแนน	ลำดับ	สตาร์	ลำดับ	คะแนน	ลำดับ	โหวต	ลำดับ
Flask	57.85	7	63462	6	7.2	2	8731	6
Svelte	76.44	2	68584	5	4.5	6	4753	7
jQuery	62.36	6	57582	7	7.9	1	15782	2
Nuxt.js	81.83	1	45950	8	6.7	3	2649	8
Angular	69.98	4	88678	4	6	4	12537	3
Next.js	72.81	3	107970	3	3.6	7	11969	4
Vue.js	55.43	8	204165	2	3.2	8	11761	5
React	62.56	5	209272	1	4.9	5	29137	1
Spearman Correlation			-0.333		-0.024		-0.405	

ผลการวิเคราะห์สหสัมพันธ์พบว่า คะแนนจากวิธี OSS-AQM มีความสัมพันธ์แบบโมโนทอนิกในทิศทางตรงกันข้ามกับทุกวิธีดังกล่าวข้างต้น โดยวิธี Stack Overflow และ Github Stars มีความสัมพันธ์ตรงกันข้ามในระดับต่ำ (ผลการวิเคราะห์สหสัมพันธ์เป็น -0.405, -0.333 ตามลำดับ) นอกจากนี้วิธี deps.dev ไม่มีความสัมพันธ์หรือมีความสัมพันธ์ในทิศทางตรงกันข้ามในระดับต่ำมากกับผลคะแนนที่ได้จาก OSS-AQM (ผลการวิเคราะห์สหสัมพันธ์เป็น -0.024) จึงสรุปได้ว่า OSS-AQM เฉพาะปัจจัยคุณภาพ Economics ไม่สามารถใช้แทนวิธีในทุกวิธีดังกล่าว ผู้วิจัยมองว่าเฉพาะปัจจัยคุณภาพ Economics นี้เป็นมุมมองใหม่ที่นำเสนอแค่เพียงในแบบจำลองคุณภาพเท่านั้นเพราะเป็นมุมมองระดับองค์กรที่ได้ประโยชน์ ไม่ใช่จากผู้ใช้งานโอเพนซอร์ซที่ตอบแบบสำรวจ หรือนักพัฒนาจากกิตฮับโดยตรง

- **ปัจจัยคุณภาพ Product Quality** เป็นปัจจัยคุณภาพที่เน้นวิเคราะห์คุณภาพของโค้ด หรือซอร์ซโค้ดของโปรเจคโอเพนซอร์ซซึ่งเป็นผลิตภัณฑ์จริง ๆ ของโปรเจคโอเพนซอร์ซ เป็นสิ่งที่ระบบอื่น ๆ นำไปใช้งานหรือเป็นส่วนหนึ่งของระบบอื่นในรูปแบบ Dependency โดยในปัจจัยคุณภาพนี้ประกอบไปด้วย 7 ปัจจัยคุณภาพย่อย คือ Code Quality, Reliability, Maintainability, Security, Testibility, Compatibility และ Performance ผลลัพธ์แสดงดังตารางที่ 6.8

ตารางที่ 6.8 ผลลัพธ์การวิเคราะห์สหสัมพันธ์ เมื่อเครื่องมือพิจารณาเฉพาะปัจจัยคุณภาพ Product Quality

โปรเจค	OSS-AQM		GitHub Stars		deps.dev		Stack Overflow	
	คะแนน	ลำดับ	สตาร์	ลำดับ	คะแนน	ลำดับ	โหวต	ลำดับ
Flask	82.74	1	63462	6	7.2	2	8731	6
Svelte	69.6	2	68584	5	4.5	6	4753	7
jQuery	68.66	3	57582	7	7.9	1	15782	2
Nuxt.js	66.88	6	45950	8	6.7	3	2649	8
Angular	67.14	5	88678	4	6	4	12537	3
Next.js	61.04	7	107970	3	3.6	7	11969	4
Vue.js	67.29	4	204165	2	3.2	8	11761	5
React	58.39	8	209272	1	4.9	5	29137	1
Spearman Correlation			**-0.476**		**0.333**		**-0.404**	

ผลการวิเคราะห์สหสัมพันธ์ได้ว่า วิธี deps.dev นั้นมีความสัมพันธ์แบบโมโนทอนิกในทิศทางเดียวกันกับวิธี OSS-AQM ในปัจจัยคุณภาพนี้ในระดับต่ำ (ผลการวิเคราะห์สหสัมพันธ์เป็น

0.333) จากการตรวจสอบพบว่ามีเพียงตัววัด Vulnerabilities จาก deps.dev ที่สอดคล้องกับวิธี OSS-AQM ในระดับปัจจัยคุณภาพย่อย Security ซึ่งตรวจสอบช่องโหว่ที่เกี่ยวข้องกับความมั่นคงในโค้ด (Vulnerabilities) แต่เนื่องจากวิธี deps.dev พิจารณาตัววัดด้านความมั่นคงอีกหลายตัววัด จึงทำให้ได้ความสัมพันธ์ในระดับต่ำ นอกจากนี้วิธี Github Stars และ Stack Overflow นั้นมีความสัมพันธ์แบบโมโนทอนิกในทิศทางตรงกันข้ามในระดับปานกลาง (ผลการวิเคราะห์สหสัมพันธ์เป็น -0.476) และต่ำ (ผลการวิเคราะห์สหสัมพันธ์เป็น -0.404) ตามลำดับ เนื่องจากเป็นวิธีที่อ้างอิงข้อมูลจากความนิยมของผู้ใช้ และแหล่งพัฒนาชุมชนซึ่งตรงกันข้ามกับปัจจัยคุณภาพ Product Quality ที่เน้นวัดคุณภาพของโค้ดของโอเพนซอร์ซ จึงทำให้ได้ข้อสังเกตว่าซอฟต์แวร์โอเพนซอร์ซที่นิยมใช้กันนั้นอาจจะมีข้อบกพร่องด้านคุณภาพเชิงเทคนิคซ่อนอยู่ภายในโค้ด โดยที่ผู้ใช้ไม่รับรู้ หรืออาจเป็นกรณีที่ผู้ใช้ไม่ให้ความสนใจก็ได้ เช่น กรณีที่ผู้ใช้ใช้งานโอเพนซอร์ซแต่เพียงอย่างเดียว โดยไม่ได้แก้ไขโค้ด จึงใช้โอเพนซอร์ซตามความนิยมโดยไม่ต้องสนใจว่าโค้ดมีความซ้ำซ้อน มีโค้ดซ้ำหรือมีร่องรอยที่ไม่ดี ซึ่งค่าคุณภาพเหล่านี้สะท้อนอยู่ในวิธี OSS-AQM

6.2.3 การเปรียบเทียบทีละรายการโอเพนซอร์ซ

ในขั้นตอนนี้เป็นการเปรียบเทียบผลจากตารางที่ 6.3 เพิ่มเติมโดยการลงรายละเอียดที่น่าสนใจในแต่ละรายการโอเพนซอร์ซซึ่งมีรายละเอียดดังต่อไปนี้

- **Flask** ได้ลำดับที่ 6 ด้วยวิธี GitHub Stars และ Stack Overflow เนื่องจากได้รับความนิยมลดลงเมื่อเทียบกับเทคโนโลยีเว็บสมัยใหม่ที่มาทดแทน แต่ได้ลำดับที่ 1 ในวิธี OSS-AQM และลำดับที่ 2 ด้วยวิธี deps.dev ทำให้ผลลัพธ์ที่ได้ขัดแย้งอย่างเห็นได้ชัด จากการตรวจสอบในเครื่องมือ OSS-AQM ได้คะแนนรวม 76.18 คะแนน มากที่สุดในรายการโอเพนซอร์ซ จากการตรวจสอบพบว่าได้คะแนนในระดับที่สูงจากเกือบทุกปัจจัยคุณภาพ โดยมีเพียงบางปัจจัยคุณภาพย่อยที่ได้น้อย เช่น Innovativeness (20.34/100) และ Competitiveness (30/100) ซึ่งคาดว่าได้รับผลกระทบจากการใช้งานและความนิยมที่ลดลงจึงทำให้เกิด Pull Request ใหม่ ๆ น้อยลง จากการตรวจสอบเพิ่มเติมในเครื่องมือ deps.dev ให้คะแนนรวม 7.2/10 คะแนน โดยตัววัดส่วนใหญ่ได้คะแนน 10/10 และมีเพียง 3 ตัววัดเท่านั้นที่ได้คะแนน 0/10 คือ CII-Best-Practices, Token-Permission, SAST จากการค้นพบนี้ทำให้ทราบว่า Flask ได้รับการพัฒนามาอย่างยาวนาน 13 ปีตั้งแต่ปี 2010 และ

มีการสนับสนุนที่ดีมากจากทีมพัฒนาโดยขณะนี้ไม่มี Issue ที่อยู่สถานะรอแก้ในกิตฮับ (3 กรกฎาคม พ.ศ. 2566)

- **React** ได้ลำดับที่ 1 ในวิธี GitHub Stars และ Stack Overflow เนื่องจากได้รับความนิยมอย่างมากและมีฐานผู้ใช้งานขนาดใหญ่ แต่ได้ลำดับที่ 8 สำหรับวิธี OSS-AQM และลำดับที่ 5 สำหรับวิธี deps.dev จากการตรวจสอบในเครื่องมือ OSS-AQM ได้คะแนนรวม 67.34/100 คะแนน เมื่อลงรายละเอียดเพิ่มเติมนั้นทำให้ทราบว่าในปัจจัยคุณภาพ Product Quality ได้คะแนนระดับต่ำกว่าที่ควรในปัจจัยคุณภาพย่อย Innovativeness (6.43/100), Reliability (20/100), Security (40/100) และ Testability (50/100) และได้ Maturity (57.67/100) จากปัจจัยคุณภาพ Operational Software Characteristics นอกจากนี้จากการตรวจสอบเพิ่มเติมในเครื่องมือ deps.dev พบว่าได้คะแนนรวมเพียง 4.9/10 คะแนน โดยตัววัดหลายตัวของเครื่องมือนี้ได้ 0/10 คะแนน ประกอบด้วย Signed-Releases (0/10), Token-Permission (0/10), SAST (0/10), Fuzzing (0/10), Vulnerabilities (0/10 - 171 vulnerabilities detected) จากการค้นพบนี้ทำให้ทราบว่าวิธี GitHub Stars และ Stack Overflow นั้นไม่เพียงพอเพื่อใช้สะท้อนคุณภาพของโอเพนซอร์ซทั้งหมดและตัวโอเพนซอร์ซมีความเสี่ยงในเรื่องความมั่นคงในการใช้งานอย่างมากในเวลานี้
- **Vue.js** ได้ลำดับที่ 2 รองจาก React ในวิธี GitHub Stars แต่ได้ลำดับที่ 7 ด้วยวิธี OSS-AQM และได้ลำดับที่ 8 ด้วยวิธี deps.dev จากการตรวจสอบด้วยเครื่องมือ OSS-AQM ให้คะแนนรวมเพียง 69.18 เมื่อตรวจสอบเพิ่มเติมพบว่าได้คะแนนต่ำในหลายปัจจัยคุณภาพย่อย เช่น Innovativeness (52.07/100), Competitiveness (3.33/100) ของปัจจัยคุณภาพ Economics และมีปัจจัยคุณภาพย่อย เช่น Reliability (60/100), Testability (50/100), Performance (31.65/100) ในปัจจัยคุณภาพ Product Quality และเมื่อตรวจสอบเพิ่มเติมในเครื่องมือ deps.dev ได้คะแนนรวม 3.2/10 คะแนนเนื่องจากได้คะแนนต่ำในหลายตัววัด ประกอบด้วย Maintained (3/10), CII-Best-Practices (2/10), Fuzzing (0/10), Token-Permission (0/10), SAST (0/10) และ Vulnerabilities (0/10 - 12 vulnerabilities detected) จากการค้นพบนี้ทำให้ทราบว่าวิธี GitHub Stars นั้นไม่เพียงพอที่จะใช้สะท้อนคุณภาพของโอเพนซอร์ซทั้งหมด และจากการตรวจสอบข่าวเกี่ยวกับ Vue.js นั้นทำให้ทราบว่าโปรเจคนี้ (Vue.js V2) ได้หยุดพัฒนาเพิ่มเติมแล้ว และได้ย้ายนักพัฒนาส่วนใหญ่ไปพัฒนา Vue.js V3 ที่อยู่อีกโปรเจคนั่นเอง (ปัจจุบัน Vue.js V3 มีจำนวนกิตฮับสตาร์ประมาณ 3 หมื่นคนในขณะที่ Vue.js V2 มีกิตฮับสตาร์มากถึง 2 แสนคน)

จากการประเมินด้วยการเปรียบเทียบเครื่องมือด้วยแหล่งข้อมูลต่าง ๆ ทำให้ทราบว่า OSS-AQM ได้ให้ผลลัพธ์ที่ขัดแย้งกับการประเมินด้วยรูปแบบวิธีปัจจุบัน ได้แก่ GitHub Stars และ Stack Overflow หากประเมินด้วยปัจจัยคุณภาพทั้งหมดของแบบจำลองคุณภาพ แต่กลับสอดคล้องกันมากขึ้นในระดับปัจจัยคุณภาพ Software License และ Community and Support เท่านั้นซึ่งเป็นปัจจัยด้านความนิยมผู้ใช้งานและแหล่งพัฒนาชุมชน ซึ่งจากการประเมินทำให้เห็นว่า การเลือกซอฟต์แวร์โอเพนซอร์ซที่มีคุณภาพนั้นจำเป็นต้องตรวจสอบด้วยหลายปัจจัยคุณภาพ ไม่ใช้เพียงความนิยมของโอเพนซอร์ซ อีกทั้งเครื่องมือ OSS-AQM ก็ให้ผลลัพธ์ที่สอดคล้องกับเครื่องมือ deps.dev ที่ตรวจสอบคุณภาพโค้ดและช่องโหว่ความมั่นคงในระดับปานกลาง สรุปคือเครื่องมือ OSS-AQM สามารถครอบคลุมปัจจัยต่าง ๆ ที่มาจากแหล่งข้อมูลที่หลากหลายได้ โดยผู้ใช้สามารถนำเครื่องมือ OSS-AQM มาใช้เพื่อตรวจสอบเพื่อประกอบการตัดสินใจเลือกใช้ซอฟต์แวร์โอเพนซอร์ซควบคู่กับแหล่งต่าง ๆ อย่าง GitHub Stars, deps.dev และ Stack Overflow ได้ นอกจากนี้จากการพิจารณาตารางที่ 6.3 ยังพบว่าค่าคะแนนของโอเพนซอร์ซที่ได้จากเครื่องมือ OSS-AQM ในแต่ละลำดับมีค่าไม่ต่างกันมากนัก เนื่องจากในตัววัดส่วนใหญ่ โอเพนซอร์ซจะได้คะแนนเสมอไม่มากก็น้อย (เมื่อเปรียบเทียบกับวิธี deps.dev ที่โอเพนซอร์ซได้ค่าเป็น 0 ในหลายตัววัด) ดังนั้นในการใช้เครื่องมือ OSS-AQM ควรพิจารณาทั้งคะแนนรวม และคะแนนรายปัจจัยคุณภาพย่อยประกอบกัน เพื่อช่วยให้เข้าใจเกี่ยวกับคุณภาพของโอเพนซอร์ซมากขึ้นและตัดสินใจเลือกใช้ได้ดีขึ้น

บทที่ 7
สรุปผลการวิจัย ข้อจำกัด และข้อเสนอแนะ

7.1 สรุปผลการวิจัย

วิทยานิพนธ์เล่มนี้ได้นำเสนอแบบจำลองคุณภาพโอเพนซอร์ซที่มีวัตถุประสงค์เพื่อใช้ตรวจสอบคุณภาพของโอเพนซอร์ซ และให้คะแนนคุณภาพของโอเพนซอร์ซที่สนใจ ทั้งนี้เมื่อสามารถแปลงคำว่าคุณภาพที่เป็นเชิงนามธรรมให้อยู่ในรูปแบบตัวเลขได้จะทำให้สามารถนำมาเปรียบเทียบกับรายการโอเพนซอร์ซอื่น ๆ ที่สนใจได้ โดยแบบจำลองคุณภาพฯ เป็นการออกแบบโครงสร้างตามลำดับชั้นซึ่งประกอบด้วย ชั้นที่ 1 ปัจจัยคุณภาพ 5 ปัจจัย ชั้นที่ 2 ปัจจัยคุณภาพย่อย 19 ปัจจัย ชั้นที่ 3 ตัววัดคุณภาพจำนวน 19 ตัววัด และชั้นที่ 4 องค์ประกอบของตัววัด 25 องค์ประกอบ ผลลัพธ์จากการนำแบบจำลองฯ ไปใช้งานจะสามารถประเมินโอเพนซอร์ซใด ๆ ในกิตฮับได้ด้วยคะแนนเต็ม 100 คะแนน

การประเมินแบบจำลองคุณภาพนั้นได้ถูกประเมินด้วยแบบสอบถามและการสัมภาษณ์จากผู้ประเมินทั้งหมด 26 ท่านโดยเป็นผู้ประเมินที่มีประสบการทำงานเกี่ยวกับซอฟต์แวร์โอเพนซอร์ซที่มาจากองค์กรต่าง ๆ เช่น บริษัทสตาร์ทอัพ บริษัทพัฒนาซอฟต์แวร์ขนาดกลาง/ใหญ่ องค์กรที่พัฒนาโอเพนซอร์ซโดยตรง และบุคคลทั่วไปที่มีประสบการณ์ทำงานเกี่ยวกับซอฟต์แวร์โอเพนซอร์ซมากกว่า 1 ปีขึ้นไป ผลการประเมินมีคะแนนโดยรวมที่น่าพอใจ (มากกว่าร้อยละ 70 ขึ้นไป) ทำให้แบบจำลองคุณภาพฯ ครอบคลุมพอที่จะนำไปใช้ตรวจสอบคุณภาพซอฟต์แวร์โอเพนซอร์ซในบริบทต่าง ๆ ที่หลากหลาย นอกจากนี้เครื่องมือสนับสนุนแบบจำลองคุณภาพได้ถูกพัฒนาเป็นโปรแกรมที่สามารถทำงานบนเว็บไซต์ และรองรับทุกระบบปฏิบัติการ เพื่อให้ผู้ใช้งานไม่ว่าจะเป็นหัวหน้าโครงการ ผู้พัฒนาซอฟต์แวร์อาวุโส และบุคคลทั่วไปสามารถเข้าถึงเครื่องมือได้ง่าย อีกทั้งผู้วิจัยได้เปิดเผยซอร์ซโค้ดให้นักพัฒนารุ่นต่อไปได้ทำการศึกษา ปรับปรุง และแสดงความคิดเห็นได้อย่างเปิดเผย

การประเมินเครื่องมือนั้นได้ถูกประเมินด้วยวิธีการเปรียบเทียบจากแหล่งอ้างอิงต่าง ๆ ที่ได้รับการยอมรับในปัจจุบัน ได้แก่ GitHub Stars, Stack Overflow และ deps.dev โดยใช้วิธีการนำข้อมูลที่ได้จากทุกวิธีมาจัดเรียงลำดับและนำลำดับมาเปรียบเทียบผลกับเครื่องมือ OSS-AQM ที่เป็นเครื่องมือสนับสนุนของแบบจำลองคุณภาพ จากนั้นนำมาหาค่าความสัมพันธ์ด้วยวิธี Spearman Correlation จากการประเมินด้วยการเปรียบเทียบเครื่องมือด้วยแหล่งข้อมูลต่าง ๆ

ทำให้ทราบว่า OSS-AQM ได้ให้ผลลัพธ์ที่ขัดแย้งกับการประเมินด้วยรูปแบบวิธีปัจจุบัน ได้แก่ GitHub Stars และ Stack Overflow หากประเมินด้วยปัจจัยคุณภาพทั้งหมดของแบบจำลองคุณภาพ แต่กลับสอดคล้องกันมากขึ้นในระดับปัจจัย Software License และ Community and Support เท่านั้นซึ่งเป็นปัจจัยด้านความนิยมของผู้ใช้และแหล่งพัฒนาชุมชนเพียงด้านเดียว และไม่ได้ครอบคลุมปัจจัยคุณภาพอื่นที่จำเป็น สรุปคือเครื่องมือ OSS-AQM สามารถครอบคลุมปัจจัยต่าง ๆ ที่มาจากแหล่งข้อมูลที่หลากหลายได้ โดยผู้ใช้สามารถนำเครื่องมือ OSS-AQM มาใช้เพื่อตรวจสอบเพื่อประกอบการตัดสินใจเลือกใช้ซอฟต์แวร์โอเพนซอร์ซควบคู่กับแหล่งต่าง ๆ อย่าง GitHub Stars, deps.dev และ Stack Overflow ได้

7.2 อุปสรรคและข้อจำกัด

1. อุปสรรคในการค้นหาปัจจัยคุณภาพหลัก/ปัจจัยคุณภาพย่อย : เนื่องจากแบบจำลองคุณภาพที่ให้ความสำคัญกับการวัดคุณภาพของซอฟต์แวร์โอเพนซอร์ซมีการตีพิมพ์จำนวนมากจาก ค.ศ. 1998-2019 ทำให้ต้องใช้เวลาจำนวนมากในการสรุปผลแบบจำลองฯ และพิจารณาว่าปัจจัยใดบ้างที่น่าจะสามารถประเมินคุณภาพได้อย่างอัตโนมัติ
2. อุปสรรคในการนิยามตัววัดคุณภาพที่จะสามารถทำงานได้อัตโนมัติ : จากงานวิจัยที่ตีพิมพ์ก่อนหน้าทำให้ผู้วิจัยทราบว่ามีตัววัดคุณภาพจำนวนมากที่ใช้เพื่อตรวจสอบคุณภาพโอเพนซอร์ซ แต่ไม่มีคำอธิบายในรายละเอียดที่มากพอถึงวิธีการคำนวณที่แน่ชัด อีกทั้งเครื่องมือที่งานวิจัยดังกล่าวข้างต้นได้เผยแพร่ไว้นั้นไม่สามารถเข้าถึงได้อีกต่อไปในปัจจุบันทำให้ยากที่จะรวบรวมตัววัดคุณภาพที่ใช้งานได้จริง ตามวัตถุประสงค์ของงานวิจัย
3. อุปสรรคในการพัฒนาเครื่องมือ : การพัฒนาเครื่องมือมีข้อจำกัดว่าจะต้องเป็นโปรเจคโอเพนซอร์ซที่สามารถเข้าถึงซอร์ซโคดบนกิตฮับได้เท่านั้น ทำให้ไม่สามารถนำแบบจำลองที่พัฒนาไปประยุกต์ใช้ในแพลตฟอร์มอื่น ๆ เช่น กิตแลบ (Gitlab), บิตบักเก็ต (Bitbucket) อะซัวร์โปรเจค (Azure Project) และ แอมาซอนโค้ดคอมมิต (Amazon CodeCommit) เป็นต้น
4. ข้อจำกัดของโซนาร์คิวบ์เซิร์ฟเวอร์ที่ใช้งานร่วมกับเครื่องมือรองรับรายการภาษาที่จำกัด (อ้างอิงรูปที่ 2.1 ภาษาที่รองรับตามเวอร์ชันของโซนาร์คิวบ์) ผู้วิจัยได้ใช้ Community Edition ทำให้เมื่อวิเคราะห์ภาษาที่ไม่ได้รองรับจะไม่มีค่าส่งกลับมาเพื่อประมวลผลข้อมูล
5. ข้อจำกัดของกิตฮับเอพีไอที่สามารถส่งคำร้องได้ 5,000 ครั้งต่อชั่วโมงต่อผู้ใช้ ผู้ใช้จึงสามารถส่งคำร้องขอนำเข้าข้อมูลโปรเจคกิตฮับในปริมาณที่จำกัดต่อชั่วโมง

7.3 ข้อเสนอแนะ

ข้อเสนอแนะดังต่อไปนี้เป็นข้อเสนอแนะจากผู้วิจัยที่ได้เล็งเห็นถึงช่องว่างที่จะสามารถต่อยอดได้ในอนาคต รวมถึงการปรับปรุงแบบจำลองคุณภาพฯ และเครื่องมือสนับสนุนให้มีความสามารถเพิ่มขึ้น โดยมีดังต่อไปนี้

1. สามารถต่อยอดปัจจัยคุณภาพอื่น ๆ เพิ่มเติม เช่น การใช้งานง่าย (Usability) การอ่านง่าย (Clean Code) รวมเข้ากับแบบจำลองปัจจุบันได้ ทั้งนี้ทำให้การวิเคราะห์ปัจจัยคุณภาพที่จำเป็นสำหรับการตรวจสอบซอฟต์แวร์โอเพนซอร์ซครอบคลุมมากยิ่งขึ้น
2. สามารถต่อยอดแบบจำลองคุณภาพเพื่อตรวจสอบเฉพาะโปรเจคที่เป็นลักษณะไลบรารี เช่น การนำวิธีการวัดระดับความปลอดภัยจากเว็บไซต์ deps.dev มาต่อยอดตัววัดคุณภาพที่ 16 Security Level ให้ดียิ่งขึ้น
3. พัฒนาเครื่องมือให้สามารถแนะนำโอเพนซอร์ซที่คล้ายกันจะทำให้สามารถเสนอแนะซอฟต์แวร์โอเพนซอร์ซที่ใกล้เคียงกัน ทั้งนี้มีประโยชน์ต่อผู้ใช้งานที่กำลังหาซอฟต์แวร์โอเพนซอร์ซทั้งที่เคยศึกษาและยังไม่เคยศึกษานำไปเปรียบเทียบเพิ่มเติม
4. พัฒนาเครื่องมือให้มีระบบจัดลำดับซอฟต์แวร์โอเพนซอร์ซจากผลคะแนนรวมที่แต่ละโอเพนซอร์ซทำได้ ทั้งนี้มีประโยชน์อย่างยิ่งต่อการจัดลำดับตามคุณภาพของซอฟต์แวร์โอเพนซอร์ซ
5. พัฒนาเครื่องมือให้สามารถนำข้อมูลออกในรูปแบบไฟล์ต่าง ๆ ได้เช่น CSV, Excel, PDF เป็นต้น

บรรณานุกรม

1. Lenarduzzi, V., et al. *Open source software evaluation, selection, and adoption: a systematic literature review*. in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2020. IEEE.
2. Sbai, N., et al., *Exploring information from OSS repositories and platforms to support OSS selection decisions.* Information and Software Technology, 2018. **104**: p. 104-108.
3. Rating, B.R. *Business readiness rating for open source*. 2005.
4. Li, X. and S. Moreschini, *OSS PESTO: An Open Source Software Project Evaluation and Selection TOol.* arXiv preprint arXiv:2102.12267, 2021.
5. Wasserman, A.I., et al. *OSSpal: finding and evaluating open source software*. in *IFIP International Conference on Open Source Systems*. 2017. Springer, Cham.
6. Duijnhouwer, F. and C. Widdows, *Open Source Maturity Model. Cap Gemini Expert Letter (August 2003)*. 2003.
7. Origin, A. *Method for Qualification and Selection of Open Source Software (QSOS)*. 2004; Available from: http://www.qsos.org
8. Zahoor, A., K. Mehboob, and S. Natha, *Comparison of open source maturity models.* Procedia computer science, 2017. **111**: p. 348-354.
9. Inc., N. *The Open Source Maturity Model is a vital tool for planning open source success*. 2004 [cited 2016 August]; Available from: http://navicasoft.com/pages/osmm.htm
10. Taibi, D., L. Lavazza, and S. Morasca. *OpenBQR: a framework for the assessment of OSS*. in *IFIP International Conference on Open Source Systems*. 2007. Springer.
11. Samoladas, I., et al. *The SQO-OSS quality model: measurement based open source software evaluation*. in *IFIP international conference on open source systems*. 2008. Springer.
12. Soto, M. and M. Ciolkowski. *The QualOSS open source assessment model measuring the performance of open source communities*. in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*.

2009. IEEE.

13. Del Bianco, V., et al. *Quality of open source software: the QualiPSo Trustworthiness Model*. in *IFIP International Conference on Open Source Systems*. 2009. Springer.
14. STANDARDIZATION, I.O.F., *ISO/IEC 25010: Systems and software engineering-systems and Software Quality Requirements and Evaluation (SQuaRE)*. 2011, System and software quality models Geneva.
15. Google. *Open Source Insights*. 2022; Available from: https://www.deps.dev.
16. Inc, R.H. *What's open source?* 2019 [cited 2021 November]; Available from: https://opensource.com/resources/what-open-source.
17. Brown, K. *What Is GitHub, and What Is It Used For?* 2019 [cited 2021; Available from: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/.
18. Github. *Saving repositories with stars*. Available from: https://docs.github.com/en/get-started/exploring-projects-on-github/saving-repositories-with-stars.
19. S.A, S. *About SonarQube*. [cited 2021 Nov 29]; Available from: https://www.sonarqube.org/about/.
20. Inc, S.E. *About Us*. [cited 2021 Nov 29]; Available from: https://stackexchange.com/about.
21. Del Bianco, V., et al., *A survey on open source software trustworthiness.* IEEE software, 2011. **28**(5): p. 67-75.
22. *OpenSSF Scorecard*. 2023/7/9; Available from: https://github.com/ossf/scorecard.
23. *Stack Overflow Developer Survey 2023*. 2023; Available from: https://survey.stackoverflow.co/2023/.
24. Bartz, A.E., *Basic statistical concepts*, ed. 4. 1999: Merrill.

ภาคผนวก

ภาคผนวก ก
ตัววัดคุณภาพทั้งหมดในแบบจำลองคุณภาพที่นำเสนอ

ตารางที่ ก.1 ตัววัดคุณภาพ V1 - OSS License

Metric name	V1 – OSS License
Hierarchy structure	Software License > License > OSS License
Description	OSS License refers to the open-source software license and permission to distribute software or source code. It consists of one quality metric element: • License Type refers to different types of open-source licenses that define how the open source can be used, modified, and shared.
Equation	$$V_1 = \frac{M}{5} \text{x} 100$$ where V1 OSS License M License Type M = 1 Undefined/Unclear M = 2 LGPL-2.1, GPL-2.0 M = 3 AGPL-3.0, EPL-2.0, GPL-3.0, MPL-2.0 M = 4 BSD-2-Clause, BSD-3-Clause, BSL-1.0, CC0-1.0 M = 5 Apache 2.0, MIT
Reference model	Metric element M adapted from OSMM [6]
Information source	GH – License API (https://docs.github.com/en/rest/reference/licenses) License types reference (https://choosealicense.com/appendix/)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.2 ตัววัดคุณภาพ V2 - Size of Community

Metric name	V2 – Size of Community
Hierarchy structure	Community and Support > Community Size > Size of Community
Description	Size of Community refers to the number of members in the open-source community. It consists of one quality metric element: • Number of Core Team Members, Contributors, and Watchers where ○ Number of Core Team Members refers to the number of core members of the open-source project, including owners and dedicated members, who determine the direction and evolution of the project. ○ Number of Contributors refers to the number of open-source project members who have contributed to the commit history of the project. ○ Number of Watchers refers to the number of open-source project members who have will be notified of activity in the project, but have not contributed to the project.
Equation	$$V_2 = \frac{M}{5} x100$$ where V2 Size of Community M Number of Core Team Members, Contributors, and Watchers M = 1 Small (< 50 people) M = 2 Relatively Small (50 – 115 people) M = 3 Medium (116 - 183 people) M = 4 Relatively Large (184 – 249 people) M = 5 Large (≥ 250 people)
Reference model	Metric element M adapted from Eurostat Glossary: Enterprise Size Website
Information source	GH – List of contributors and watchers (https://docs.github.com/en/rest/reference/activity#watching)

	GH – List of core team members (https://github.com/orgs/project_name/people) Community size based on Eurostat Glossary: Enterprise Size Website (https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:Enterprise_size)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.3 ตัววัดคุณภาพ V3 – Q&A Volume

Metric name	V3 – Q&A Volume
Hierarchy structure	Community and Support > Availability of Forum > Q&A Volume
Description	Q&A Volume refers to the number of questions and answers in the discussion forum of the open-source software which indicates the activeness of the forum. It consists of one quality metric element: • Average Q&A refers to an average number of questions and answers in the forum per month in the past 6 months.
Equation	$$V_3 = \frac{M}{5} \text{x} 100$$ where V_3 Q&A Volume M Average Q&A per month in the past 6 months M = 1 $\leq$ 30 messages M = 2 > 30 - 150 messages M = 3 > 150 - 300 messages M = 4 > 300 - 720 messages M = 5 > 720 messages
Reference model	Metric element M adapted from OpenBRR [3]
Information source	SE – Questions and answers related to the project name (https://api.stackexchange.com/docs/questions)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.4 ตัววัดคุณภาพ V4 – Number of Support Contributors

Metric name	V4 – Number of Support Contributors
Hierarchy structure	Community and Support > Support Contributors > Number of Support Contributors
Description	Number of Support Contributors refers to the number of open-source project members who provide support for the use of software. It consists of two quality metric elements: • Number of Contributors refers to the number of open-source project members who have contributed to the commit history of the project. • Number of Core Team Members refers to the number of core members of the open-source project, including owners and dedicated members, who determine the direction and evolution of the project.
Equation	$$V_4 = \frac{M}{5} x100$$ where V_4 Number of Support Contributors M Number of Contributors and Core Team member M=1 < 5 persons M=2 5-10 persons M=3 > 10-20 persons M=4 > 20-50 persons M=5 > 50 persons
Reference model	Metric element M adapted from OpenBRR [3]
Information source	GH – List of contributors and core team members (https://docs.github.com/en/rest/reference/teams)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.5 ตัววัดคุณภาพ V5 – Support Activity

Metric name	V5 – Support Activity
Hierarchy structure	Community and Support > Quality of Professional Support > Support Activity
Description	Support Activity refers to the degree of support activity for issues and pull requests. It consists of two quality metric elements: • Issue Support Activity is the proportion of issues that have been responded in the past 6 months. If there are no reported issues, there will be no evidence to show whether the support is active recently. • Pull Request Support Activity is the proportion of pull requests that have been responded in the past 6 months. If there are no pull requests, there will be no evidence to show whether the support is active recently.
Equation	(1) $$V_5 = \left(\frac{M_1 + M_2}{2}\right) x100$$ where V_5 Support Activity M_1 Issue Support Activity M_2 Pull Request Support Activity (2) $$M_1 = \begin{cases} \left(\frac{Number\ of\ issues\ that\ have\ been\ responded\ in\ the\ past\ 6\ months}{N}\right), & N > 0 \\ 0, & N = 0 \end{cases}$$ where M_1 Issue Support Activity N Total issues in the past 6 months (3)

	$$M_2 = \begin{cases} \left(\dfrac{Number\ of\ pull\ requests\ that\ have\ been\ responded\ in\ the\ past\ 6\ months}{N} \right), & N > 0 \\ 0, & N = 0 \end{cases}$$ where M_2 Pull Request Support Activity N Total pull requests in the past 6 months
Reference model	Metric element M1 newly introduced Metric element M2 newly introduced
Information source	GH – Number of issues and issues's comments (https://docs.github.com/en/rest/reference/issues#get-an-issue-comment) GH – Number of pull requests and pull request's comments (https://docs.github.com/en/rest/pulls)
Output	The result of the equation is a decimal number in the range [0,100].

ตารางที่ ก.6 ตัววัดคุณภาพ V6 – Maturity Level

Metric name	V6 – Maturity Level
Hierarchy structure	Operational Software Characteristics > Maturity > Maturity Level
Description	Maturity Level refers to the degree to which the open-source software meets the needs for reliability under normal operation. It consists of three quality metric elements: • Age refers to the age of project repository from created date to last update. • Issueless Code refers to a characteristic of the code with small number of issues in the past 6 months. It indicates that the open-source software is likely to be mature and work correctly under normal operation. • Minor Releases refers to the number of minor releases in the past 12 months. It typically indicates that the project has planned updates and bug fixes.

Equation	(1) $$V_6 = \left(\frac{M_1 + M_2 + M_3}{3}\right) x100$$

where V_6 Maturity Level
M_1 Age
M_2 Issueless Code
M_3 Minor Releases

(2) $$M_1 = \frac{S_{age}}{5}$$

where M_1 Age
$S_{age} = 1$ Age < 2 months
$S_{age} = 2$ Age 2-12 months
$S_{age} = 3$ Age > 1 - 2 years
$S_{age} = 4$ Age > 2 - 3 years
$S_{age} = 5$ Age > 3 years

(3) $$M_2 = \frac{S_{issue}}{5}$$

where M_2 Issueless Code
$S_{issue} = 1$ Number of issues in the past 6 months > 1000
$S_{issue} = 2$ Number of issues in the past 6 months > 500 - 1000
$S_{issue} = 3$ Number of issues in the past 6 months > 100 - 500
$S_{issue} = 4$ Number of issues in the past 6 months > 50 - 100
$S_{issue} = 5$ Number of issues in the past 6 months ≤ 50

(4) $$M_3 = \frac{S_{release}}{3}$$

where M_3 Minor Releases
$S_{release} = 1$ Number of minor releases in the past 12 months 0 or > 3 versions
$S_{release} = 2$ Number of minor releases in the past 12 months 1 or 3 versions

	$S_{release}$ = 3 Number of minor releases in the past 12 months 2 versions
Reference model	Metric element M_1 adapted from OSMM [6] Metric element M_2 adapted from SQO-OSS [11] and OpenBRR [3] Metric element M_3 referenced directly from OpenBRR [3]
Information source	GH – Repository info – created_at, updated_at fields (Age) (https://docs.github.com/en/rest/reference/repos#get-a-repository) GH – Issues List (https://docs.github.com/en/rest/reference/issues) GH – Releases (https://docs.github.com/en/rest/releases)
Output	The result of the equation will be a decimal number in the range [20,100].

ตารางที่ ก.7 ตัววัดคุณภาพ V7 – Development Language Popularity

Metric name	V7 – Development Language Popularity
Hierarchy structure	Operational Software Characteristics > Development Language > Development Language Popularity
Description	Development Language Popularity refers to the popularity of programming, scripting, and markup languages, and the use of popular development languages makes it easier to manage project operation and maintenance. It consists of one quality metric element: • Computer Language refers to how commonly the languages are used by professional developers.
Equation	$$V_7 = \frac{M}{5} x100$$ where V_7 Development Language Popularity M Computer Language M = 1 Julia, COBOL, Pascal, Fortran

	M = 2 Rust, Objective-C, Dart, Scala, Perl, Haskell M = 3 Kotlin, Ruby, Assembly, VBA, Swift, R M = 4 C#, PHP, TypeScript, C++, C, Go M = 5 JavaScript, HTML/CSS, SQL, Python, Java, Bash/Shell/PowerShell
Reference model	Metric element M newly introduced
Information source	GH – List repository languages (https://docs.github.com/en/rest/reference/repos#list-repository-languages) Computer Language is based on popular computer language surveys: (https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.8 ตัววัดคุณภาพ V8 – Code Documentation

Metric name	V8 – Code Documentation
Hierarchy structure	Operational Software Characteristics > Documentation > Code Documentation
Description	Code Documentation refers to the degree to which the code documentation is made available for effective use of the open-source software. It consists of two quality metric elements: • Code Comments refers to the amount of code comments, compared with the total amount of code. Some of the comments may be used to generate API documents. • Markdown Files refers to the proportion of markdown files in the project.
Equation	(1) $$V_8 = \left(\frac{M_1 + M_2}{2}\right) x100$$ where V_8 Code Documentation M_1 Code Comments

	M_2 Markdown Files (2) $$M_1 = \left(\frac{\text{Number of comment lines}}{\text{Number of lines of code} + \text{Number of comment lines}}\right)$$ where M_1 Code Comments (3) $$M_2 = \left(\frac{\text{Number of markdown files in the project repository}}{\text{Total files in the project repository}}\right)$$ where M_2 Markdown Files
Reference model	Metric element M1 referenced directly from SonarQube Metric element M2 newly introduced
Information source	GH – project files for markdown files (https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository) SQ – comment_lines/nloc (https://docs.sonarqube.org/latest/user-guide/metric-definitions/)
Output	The result of the equation is a decimal number in the range [0,100].

ตารางที่ ก.9 Table 9 ตัววัดคุณภาพ V9 – Learning Materials

Metric name	V9 – Learning Materials
Hierarchy structure	Operational Software Characteristics > Book/Online > Learning Materials
Description	Learning Materials refers to the number of published learning materials for the open-source software. If there are none of these materials, learning about the software will have to rely on the published source code and code documentation. It consists of two quality metric elements: • Books refers to the number of book titles about the software. • Courses refers to the number of videos about the software.
Equation	$$V_9 = \frac{M}{5} x100$$ where V_9 Learning Materials M Books and Courses M = 1 None M = 2 1-3 materials M = 3 > 3 - 6 materials M = 4 > 6 - 15 materials M = 5 > 15 materials
Reference model	Metric element M adapted from OpenBRR [3]
Information source	Goog – Find books related to the project (https://developers.google.com/books/docs/v1/using) YT – Find videos related to the project (https://developers.google.com/youtube/v3/docs/videos)
Output	The result of the equation is an integer number in the range [20,100].

ตารางที่ ก.10 ตัววัดคุณภาพ V10 – Cost of Ownership Reduction

Metric name	V10 – Cost of Ownership Reduction
Hierarchy structure	Economics > Cost > Cost of Ownership Reduction
Description	Cost of Ownership Reduction refers to the degree to which the cost of owning a piece of software is reduced when the software is open source. It consists of three quality metric elements: • Maintainability Level (V15) refers to the degree of ease with which the open-source software can be maintained and is defined in terms of the Technical Debt Ratio of the software, i.e. the ratio between the cost to fix all code smells in the software and the cost to develop the software. When maintainability is high, maintenance costs can be reduced. • Support Activity (V5) refers to the degree of support activity for issues and pull requests in the past 6 months. If there are no reported issues and pull requests, there will be no evidence to show whether the support is active recently. With active support from the contributors and core team members, development costs and maintenance costs can be reduced. • Learning Materials (V9) refers to the number of published learning materials for the open-source software, i.e. books and courses. If there are none of these materials, learning about the software will have to rely on the published source code and code documentation. The materials can reduce training costs.
Equation	$$V_{10} = \left(\frac{M_1 + M_2 + M_3}{3}\right)$$ where V_{10} Cost of Ownership Reduction M_1 Maintainability Level (V15) M_2 Support Activity (V5) M_3 Learning Materials (V9)
Reference model	Metric element M1 newly introduced Metric element M2 newly introduced Metric element M3 newly introduced
Information source	Refer to information source of V15, V5, and V9
Output	The result of the equation is a decimal number in the range (13,100].

ตารางที่ ก.11 ตัววัดคุณภาพ V11 – New Features Focus

Metric name	V11 – New Features Focus
Hierarchy structure	Economics > Innovativeness > New Features Focus
Description	New Features Focus refers to the degree to which the change as new feature enhancement has been introduced to the open-source software in relation to the effort to effect change to the open source. This indicates the degree of the contributors' focus on innovativeness of the open source. The focus on enhancing the open source with new features helps keep the software that utilizes the open source upgraded and modernized, making it beneficial for economic reasons. The metric consists of one quality metric element: • New Feature Pull Requests refers to the proportion of pull requests labeled with new features in the past 6 months.
Equation	(1) $$V_{11} = M \times 100$$ where V_{11} New Features Focus M New Feature Pull Requests (2) $$M = \begin{cases} \left(\frac{Number\ of\ new\ feature\ pull\ requests\ in\ the\ past\ 6\ months}{N} \right), & N > 0 \\ 0, & N = 0 \end{cases}$$ where M New Feature Pull Requests N Total pull requests in the past 6 months
Reference model	Metric element M newly introduced
Information source	GH – Pull requests with features label (https://docs.github.com/en/rest/reference/pulls#list-pull-requests) Search keywords: feature, new feature, user request, redesign, new function, new item, new component.
Output	The result of the equation is a decimal number in the range [0,100].

ตารางที่ ก.12 ตัววัดคุณภาพ V12 – Continuing Change

Metric name	V12 – Continuing Change
Hierarchy structure	Economics > Competitiveness > Continuing Change
Description	Continuing Change refers to the degree to which the change to correct, improve, and enhance has been introduced to the open-source software. For the open source to stay competitive with other open source of a similar kind, such change needs to be incorporated regularly and in a timely manner, or else the open source becomes less satisfactory. Such change also helps with keeping the software that utilizes the open source upgraded, modernized, and competitive, making it beneficial for economic reasons. The metric consists of one quality metric element: • Pull Request Frequency refers to the frequency of pull requests that have proposed change to the project in the past 30 days.
Equation	(1) $$V_{12} = M \times 100$$ where V_{12} Continuing Change M Pull Request Frequency (2) $$M = \frac{Number\ of\ days\ in\ the\ past\ 30\ days\ that\ pull\ requests\ have\ been\ created}{30}$$ where M Pull Request Frequency
Reference model	Metric element M newly introduced
Information source	GH – Pull requests API (https://docs.github.com/en/rest/reference/pulls#list-pull-requests)
Output	The result of the equation is a decimal number in the range [0,100].

ตารางที่ ก.13 ตัววัดคุณภาพ V13 – Code Quality Level

Metric name	V13 – Code Quality Level
Hierarchy structure	Product Quality > Code Quality > Code Quality Level
Description	Code Quality Level refers to the degree to which the code of the open-source software exhibits good quality for long-term maintenance and evolution. It consists of two quality metric elements: • Uncomplex Code refers to a characteristic of the code with less degree of cyclomatic complexity which indicates that the code has better quality. • Unduplicated Code refers to a characteristic of the code with less degree of duplications which indicates that the code has better quality.
Equation	(1) $$V_{13} = \left(\frac{M_1 + M_2}{2}\right) x100$$ where V_{13} Code Quality Level M_1 Uncomplex Code M_2 Unduplicated Code (2) $$M_1 = \frac{S}{4}$$ where M_1 Uncomplex Code S = 1 Cyclomatic Complexity > 50 S = 2 Cyclomatic Complexity between 21 and 50 S = 3 Cyclomatic Complexity between 11 and 20 S = 4 Cyclomatic Complexity between 1 and 10

	(3) $$M_2 = 1 - (\frac{\text{Number of duplicated lines}}{\text{Number of lines of code}})$$ where M_2 Unduplicated Code
Reference model	Metric element M_1 referenced directly from Code Metrics Cyclomatic Complexity Website Metric element M_2 newly introduced
Information source	(For Metric M_1) SQ – complexity/functions (https://docs.sonarqube.org/latest/user-guide/metric-definitions/) Cyclomatic Complexity refer to: Code Metrics Cyclomatic Complexity (https://www.c-sharpcorner.com/article/code-metrics-cyclomatic-complexity) (For Metric M_2) SQ – duplicated_lines_density/line Complexity and Size section (https://docs.sonarqube.org/latest/user-guide/metric-definitions/)
Output	The result of the equation is a decimal number in the range (12,100].

ตารางที่ ก.14 ตัววัดคุณภาพ V14 – Reliability Level

Metric name	V14 – Reliability Level
Hierarchy structure	Product Quality > Reliability > Reliability Level
Description	Reliability Level refers to the degree to which the open-source software works well as expected without bugs, or any present bugs are not severe. It consists of one quality metric element: • Reliability Rating refers to the degree of quality of the code that is bug-free or has less severe bugs.
Equation	$$V_{14} = \frac{M}{5} x100$$ where V_{14} Reliability Level M Reliability Rating M = 1 E - at least 1 blocker bug M = 2 D - at least 1 critical bug M = 3 C - at least 1 major bug M = 4 B - at least 1 minor bug M = 5 A - 0 bug
Reference model	Metric element M referenced directly from SonarQube
Information source	SQ – reliability_rating (Reliability Section) (https://docs.sonarqube.org/latest/user-guide/metric-definitions/)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.15 ตัววัดคุณภาพ V15 – Maintainability Level

Metric name	V15 – Maintainability Level
Hierarchy structure	Product Quality > Maintainability > Maintainability Level
Description	(Formerly the SQALE rating) Maintainability Level refers to the degree of ease with which the open-source software can be maintained and is defined in terms of the Technical Debt Ratio of the software, i.e. the ratio between the cost to fix all code smells in the software and the cost to develop the software. The metric consists of one quality metric element: • Maintainability Rating refers to the degree of quality of the code having less technical debt ratio.
Equation	$$V_{15} = \frac{M}{5} x100$$ where V_{15} Maintainability Level M Maintainability Rating M = 1 E – Technical debt ratio > 50% M = 2 D – Technical debt ratio between 21 and 50% M = 3 C – Technical debt ratio between 11 and 20% M = 4 B – Technical debt ratio between 6 and 10% M = 5 A – Technical debt ratio ≤ 5%
Reference model	Metric element M referenced directly from SonarQube
Information source	SQ – maintainability rating (sqale_rating) (Maintainability Section) (https://docs.sonarqube.org/latest/user-guide/metric-definitions/)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.16 ตัววัดคุณภาพ V16 – Security Level

Metric name	V16 – Security Level
Hierarchy structure	Product Quality > Security > Security Level
Description	Security Level refers to the degree of security of the open-source software in terms of security vulnerabilities that are present in the code and how severe they are. It consists of one quality metric element: • Security Rating refers to the degree of quality of the code that is vulnerability-free or has less severe vulnerabilities.
Equation	$$V_{16} = \frac{M}{5} x100$$ where V_{16} Security Level M Security Rating M = 1 E – at least 1 blocker vulnerability M = 2 D – at least 1 critical vulnerability M = 3 C – at least 1 major vulnerability M = 4 B – at least 1 minor vulnerability M = 5 A – no vulnerabilities
Reference model	Metric element M referenced directly from SonarQube
Information source	SQ – security_rating (Security Section) (https://docs.sonarqube.org/latest/user-guide/metric-definitions/)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.17 ตัววัดคุณภาพ V17 – Testability Level

Metric name	V17 – Testability Level
Hierarchy structure	Product Quality > Testability > Testability Level
Description	Testability Level refers to the degree of ease with which the open-source software can be tested. It consists of one quality metric element: • Uncomplex Code refers to a characteristic of the code with less degree of cyclomatic complexity which indicates that there is a smaller number of paths through the code, making it less complex to test.
Equation	$$V_{17} = \frac{M}{4} x100$$ where V_{17} Testability Level M Uncomplex code (Metric element M1 of V13) M=1 Cyclomatic Complexity > 50 M=2 Cyclomatic Complexity between 21 and 50 M=3 Cyclomatic Complexity between 11 and 20 M=4 Cyclomatic Complexity between 1 and 10
Reference model	Metric element M referenced directly from Code Metrics Cyclomatic Complexity Website
Information source	SQ – complexity/functions (https://docs.sonarqube.org/latest/user-guide/metric-definitions/) Cyclomatic Complexity refer to: Code Metrics Cyclomatic Complexity (https://www.c-sharpcorner.com/article/code-metrics-cyclomatic-complexity)
Output	The result of the equation will be an integer number in the range [25,100].

ตารางที่ ก.18 ตัววัดคุณภาพ V18 – Co-existence

Metric name	V18 – Co-existence
Hierarchy structure	Product Quality > Compatibility > Co-existence
Description	Co-existence refers to the degree to which the open-source software can co-exist or work with other software in various development environment, and is dependent on the various development environment that the programming language of the project is used for. It consists of one quality metric element: • Development Environment refers to the four types of development environment, i.e. web, mobile, desktop, or embedded development environment, which the primary programming language of the project can support.
Equation	$$V_{18} = \frac{M}{4} x100$$ where V_{18} Co-existence M Development Environment, i.e. web, mobile, desktop, embedded M=1 1 Supported Development Environment Type M=2 2 Supported Development Environment Types M=3 3 Supported Development Environment Types M=4 4 Supported Development Environment Types
Reference model	Metric element M newly introduced
Information source	GH – List repository languages (https://docs.github.com/en/rest/reference/repos#list-repository-languages) Development environment supported by programming languages based on IEEE research (https://spectrum.ieee.org/top-programming-languages-2021)
Output	The result of the equation will be an integer number in the range [20,100].

ตารางที่ ก.19 ตัววัดคุณภาพ V19 – Lack of Performance Issues Level

Metric name	V19 – Lack of Performance Issues Level
Hierarchy structure	Product Quality > Performance > Lack of Performance Issues Level
Description	Lack of Performance Issues Level refers to the degree to which the open-source software works without performance issues. It consists of one quality metric element: • Lack of Performance Issues refers to the proportion of non-performance issues in relation to other reported issues in the past 6 months.
Equation	(1) $$V_{19} = M \times 100$$ where V19 Lack of Performance Issues Level M Lack of Performance Issues (2) $$M = \begin{cases} \left(1 - \left(\frac{Number\ of\ performance\ issues\ in\ the\ past\ 6\ months}{N}\right)\right), & N > 0 \\ 1, & N = 0 \end{cases}$$ where M Lack of Performance Issues N Total opened issues in the past 6 months
Reference model	Metric element M newly introduced
Information source	GH – Issues API (https://docs.github.com/en/rest/reference/issues#list-repository-issues) Search keywords: performance, space, response time, latency, transit delay, miss, rate, loss, workload, capacity, computation, speed, throughput, memory usage, accuracy, efficiency.
Output	The result of the equation is a decimal number in the range [0,100].

ภาคผนวก ข

ตัวอย่างแบบสอบถามเพื่อประเมินแบบจำลองคุณภาพ

<table>
<tr><td colspan="4"><u>แบบสอบถามความคิดเห็นต่อแบบจำลองคุณภาพที่นำเสนอ</u></td></tr>
<tr><td colspan="4">คำชี้แจง แบบสอบถามนี้ถูกจัดทำเพื่อใช้ประเมินแบบจำลองคุณภาพอันเป็นส่วนหนึ่งของวิทยานิพนธ์เรื่อง “แบบจำลองคุณภาพซอฟต์แวร์โอเพนซอร์ซเพื่อการวัดคุณภาพอย่างอัตโนมัติ” ซึ่งแบบจำลองนี้ประกอบด้วยปัจจัยคุณภาพและปัจจัยคุณภาพย่อย ที่ใช้ในการประเมินคุณภาพ และตัววัดคุณภาพซึ่งสะท้อนปัจจัยเหล่านั้น โดยมุ่งเน้นที่ตัววัดที่สามารถดึงข้อมูลมาคำนวณค่าได้อย่างอัตโนมัติเท่านั้น (ทั้งนี้เพื่อนำมาใช้ในการพัฒนาเครื่องมือสนับสนุนต่อไป) แบบสอบถามนี้มุ่งหวังจะสำรวจความคิดเห็นของผู้เชี่ยวชาญ เพื่อสรุปภาพรวมคุณภาพของแบบจำลองคุณภาพที่นำเสนอว่ามีความครอบคลุม และสมเหตุสมผลหรือไม่ สำหรับนำไปใช้ในการวัดคุณภาพและเปรียบเทียบโอเพนซอร์ซ ทั้งนี้ผู้วิจัยจะนำข้อเสนอแนะไปพิจารณาแก้ไขแบบจำลองคุณภาพต่อไป ผู้ตอบแบบสอบถามสามารถแสดงความคิดเห็นตามรายการข้อคำถามตามเกณฑ์การให้คะแนนที่แบ่งออกเป็น 5 ระดับ ดังนี้</td></tr>
<tr><td colspan="2">ระดับ 4 (เห็นด้วยอย่างยิ่ง)</td><td colspan="2">ผู้ประเมินเห็นด้วยกับข้อคำถามนั้น ๆ อย่างยิ่ง ว่าแบบจำลองคุณภาพมีคุณภาพตามข้อคำถามนั้น ๆ และไม่จำเป็นต้องแก้ไขใด ๆ</td></tr>
<tr><td colspan="2">ระดับ 3 (ค่อนข้างเห็นด้วย)</td><td colspan="2">ผู้ประเมินค่อนข้างเห็นด้วยกับข้อคำถามนั้น ๆ ว่าแบบจำลองมีคุณภาพเพียงพอ แต่ยังไม่ชัดเจน หรืออาจจะพิจารณาแก้ไขปรับปรุงเพิ่มเติม (โปรดแสดงความคิดเห็นเพิ่มเติม)</td></tr>
<tr><td colspan="2">ระดับ 2 (ค่อนข้างไม่เห็นด้วย)</td><td colspan="2">ผู้ประเมินค่อนข้างไม่เห็นด้วยกับข้อคำถามนั้น ๆ ว่าแบบจำลองคุณภาพมีคุณภาพตามข้อคำถาม ผู้วิจัยควรตรวจสอบและแก้ไขตามข้อคำถามนั้น ๆ (โปรดแสดงความคิดเห็นเพิ่มเติม)</td></tr>
<tr><td colspan="2">ระดับ 1 (ไม่เห็นด้วยอย่างยิ่ง)</td><td colspan="2">ผู้ประเมินค่อนข้างไม่เห็นด้วยกับข้อคำถามนั้น ๆ อย่างยิ่ง ผู้วิจัยจำเป็นต้องแก้ไขตามข้อนั้น ๆ ตามคำแนะนำ (โปรดแสดงความคิดเห็นเพิ่มเติม)</td></tr>
<tr><td colspan="2">X (ไม่ออกความคิดเห็น)</td><td colspan="2">ผู้ประเมินไม่ทราบ หรือไม่ขอออกความคิดเห็นตามข้อคำถามนั้น ๆ</td></tr>
<tr><td colspan="4">0. ข้อมูลทั่วไป</td></tr>
<tr><td colspan="4">คำชี้แจง โปรดกรอกเครื่องหมายถูก (✔) ลงในช่องว่าง [ ] ที่ตรงตามข้อมูลของคุณตามความเป็นจริง</td></tr>
<tr><td>อายุ</td><td>[ ] ต่ำกว่า 20 ปี [ ] 21 – 25 ปี
[ ] 26 – 30 ปี [ ] มากกว่า 30 ปี</td><td>ประสบการณ์เกี่ยวกับโครงการโอเพนซอร์ซ</td><td>[ ] 1 - 2 ปี
[ ] มากกว่า 2 ปี</td></tr>
<tr><td>มีประสบการณ์เกี่ยวกับโครงการโอเพนซอร์ซจาก</td><td colspan="3">[ ] การศึกษาในโรงเรียน/มหาวิทยาลัย/อื่น ๆ
[ ] ทำงานวิจัยเกี่ยวกับโครงการโอเพนซอร์ซ
[ ] ทำงานในองค์การที่มีการนำโครงการโอเพนซอร์ซไปใช้
[ ] อื่น ๆ (โปรดระบุ) .....</td></tr>
</table>

1. รายการข้อคำถามภาพรวม

วัตถุประสงค์ รายการคำถามส่วนนี้มีเพื่อใช้ประเมินความพร้อมของแบบจำลองคุณภาพโดยรวม รวมถึงตรวจสอบความสัมพันธ์ระหว่างปัจจัยคุณภาพในแต่ละระดับว่ามีความครบถ้วนและเหมาะสมต่อการนำไปใช้งานในระดับใด

วิธีทำ โปรดศึกษาคำอธิบาย และรายละเอียดของข้อคำถามแต่ละข้อ จากนั้นจึงกรอกเครื่องหมายถูก (✔) ลงในช่องว่างด้านขวา ตามระดับที่ท่านคิดเห็น หากมีความคิดเห็นเพิ่มเติมกรุณากรอกในที่ว่างที่เว้นไว้ หรือกรอกในช่องความคิดเห็นเพิ่มเติมท้ายรายการข้อคำถาม (ดูรายละเอียดในเอกสารประกอบการประเมิน แผนภาพแบบจำลองคุณภาพโดยรวม)

คำอธิบายข้อคำถาม	ความคิดเห็น				
	4	3	2	1	X
1. คุณคิดว่าระดับปัจจัยคุณภาพ (Quality Factors) ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสม และทำให้สามารถสะท้อนคุณภาพของซอฟต์แวร์โอเพนซอร์ซได้ ความคิดเห็น					
2. คุณคิดว่าระดับปัจจัยคุณภาพ (Quality Factors) ที่ระบุในแบบจำลองคุณภาพ ครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้ ความคิดเห็น					
3. คุณคิดว่าระดับปัจจัยคุณภาพย่อย (Sub-quality factors) ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสมทำให้สามารถสะท้อนคุณภาพของระดับก่อนหน้าได้ ความคิดเห็น					
คำอธิบายข้อคำถาม	ความคิดเห็น				

	4	3	2	1	X
4. คุณคิดว่าระดับปัจจัยคุณภาพย่อย (Sub-quality factors) ที่ระบุในแบบจำลองคุณภาพ ครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติม จากนี้ ความคิดเห็น					
5. คุณคิดว่าค่าคะแนนที่เป็นผลลัพธ์จากแบบจำลองคุณภาพ จากวิธีการคำนวณที่ระบุไว้สามารถสะท้อนคุณภาพซอฟต์แวร์โอเพนซอร์ซที่กำลังสนใจได้จริง และค่าคะแนนสามารถนำไปเปรียบเทียบกับซอฟต์แวร์โอเพนซอร์ซอื่น ๆ ได้ ความคิดเห็น					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม					

2. รายการข้อคำถามรายละเอียด

วัตถุประสงค์ รายการคำถามส่วนนี้มีเพื่อต้องการประเมินตัววัดคุณภาพ (Quality Metrics) ทั้ง 20 รายการ ว่าแต่ละรายการมีความถูกต้อง เหมาะสม และเพียงพอต่อการนำไปใช้งานได้จริงในระดับใด

วิธีทำ โปรดศึกษาคำอธิบาย และรายละเอียดของข้อคำถามแต่ละข้อ จากนั้นจึงกรอกเครื่องหมายถูก (✔) ลงในช่องว่างด้านขวา ตามระดับที่ท่านคิดเห็น หากมีความคิดเห็นเพิ่มเติมกรุณากรอกในที่ว่างที่เว้นไว้ หรือกรอกในช่องความคิดเห็นเพิ่มเติมท้ายรายการข้อคำถาม (ดูรายละเอียดในเอกสารประกอบการประเมินรายละเอียดตัววัดคุณภาพในแบบจำลองคุณภาพที่นำเสนอ)

ตัววัด	คำอธิบายข้อคำถาม	ความคิดเห็น				
		4	3	2	1	X
V1	*การลดต้นทุนการเป็นเจ้าของ (Cost of Ownership Reduction)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V2	*การเพิ่มฟังก์ชันใหม่ (New Features Focus)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					

ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V3	*ความถี่ของคำร้องขอรวมโค้ด (Pull Requests Frequency)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V4	*ประเภทไลเซนส์ของโอเพนซอร์ซ (OSS License)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V5	*ขนาดของชุมชนพัฒนา (Size of Community)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						

.....						
.....						
V6	*ปริมาณการถามตอบปัญหา (Q&A Volume)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V7	*จำนวนผู้พัฒนาโครงการ (Number of Contributors)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V8	*อัตราการสนับสนุนช่วยเหลือ (Support Rate)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						

.....						
V9	*ตัววัดคุณภาพรหัส V9 – ระดับวุฒิภาวะ (Maturity Rating)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V10	*ความนิยมของภาษาที่ใช้ในการพัฒนา (Development Language Popularity)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V11	*เอกสารสำหรับโค้ด (Code Documentation)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						

V12	*สื่อการเรียนรู้ (Learning Materials)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V13	*ระดับคุณภาพของโค้ด (Code Quality Rating)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V14	*ระดับความเชื่อถือได้ของโอเพนซอร์ซ (Reliability Rating)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						

V15	*ภาษาหลักที่รันได้หลายแพลตฟอร์ม (Multiplatform Language)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V16	*ระดับความสามารถในการบำรุงรักษา (Maintainability Rating)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V17	*ระดับความมั่นคง (Security Rating)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						

V18	*การขาดปัญหาด้านสมรรถนะ (Lack of Performance Issues)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						
V20	*โค้ดที่ไม่ซับซ้อน (Uncomplex Code)*					
1	คุณคิดว่านิยามของตัววัดคุณภาพนี้มีความสมบูรณ์ ครบถ้วน และถูกต้องแล้ว					
2	คุณคิดว่าสมการการวัดที่นำมาใช้ในตัววัดคุณภาพนี้มีความเหมาะสมแล้ว					
3	คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้ว					
4	คุณคิดว่าตัววัดคุณภาพนี้มีความสำคัญต่อการตัดสินใจเลือกโอเพนซอร์ซ					
ความคิดเห็น/ข้อเสนอแนะเพิ่มเติม						

<u>หมายเหตุ:</u>

กำหนดให้ Vn คือ ตัววัดใด ๆ

คำถามย่อยที่ 1 ชื่อตัววัด และคำอธิบายในตารางที่ Vn นั้นมีความสมบูรณ์ ครบถ้วน และถูกต้องแล้วในระดับใด

คำถามย่อยที่ 2 สมการการวัดในตารางที่ Vn นั้นเหมาะสมแล้วในระดับใด หรือควรปรับปรุงเพิ่มเติมหรือไม่

คำถามย่อยที่ 3 คุณคิดว่าแหล่งข้อมูลที่นำมาคำนวณตัววัดนี้เหมาะสมและเพียงพอแล้วในระดับใด หรือมีแหล่งข้อมูลที่ดีกว่าหรือไม่

คำถามย่อยที่ 4 หากคุณเป็นผู้เลือกโอเพนซอร์ซด้วยตัวเอง คุณคิดว่าตัววัดดังกล่าวมีความสำคัญต่อการพิจารณาในระดับใด

ภาคผนวก ค ตารางเปรียบเทียบการจัดลำดับประเภทไลเซนส์ของโอเพนซอร์ซ (OSS License) ตามตัววัดคุณภาพที่ V1

(อ้างอิงจาก: https://choosealicense.com/appendix/)

License Type	Description	GitHub Reference	Commercial use (1)	Distribution (1)	Modification (1)	Patent use (1)	Private use (2)	Closed source project use (3)	Total
GNU Affero General Public License v3.0 (AGPL-3.0)	Permissions of this strongest copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights. When a modified version is used to provide a service over a network, the complete source code of the modified version must be made available.	https://api.github.com/licenses/agpl-3.0	/	/	/	/	/	X	6
Apache License 2.0	A permissive license whose main conditions require preservation of	https://api.github.co	/	/	/	/	/	/	9

License Type	Description	GitHub Reference	Commercial use (1)	Distribution (1)	Modification (1)	Patent use (1)	Private use (2)	Closed source project use (3)	Total
(Apache-2.0)	copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code	m/licenses /apache-2.0							
BSD 2-Clause "Simplified" License (BSD-2-Clause)	A permissive license that comes in two variants, the BSD 2-Clause and BSD 3-Clause. Both have very minute differences to the MIT license.	https://api .github.com/licenses /bsd-2-clause	/	/	/	X	/	/	8
BSD 3-Clause "New" or "Revised " License (BSD-3-Clause)	A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the project or its contributors to promote derived products without written consent.	https://api .github.com/licenses /bsd-3-clause	/	/	/	X	/	/	8
Boost Software	A simple permissive license only	https://api	/	/	/	X	/	/	8

License Type	Description	GitHub Reference	Commercial use (1)	Distribution (1)	Modification (1)	Patent use (1)	Private use (2)	Closed source project use (3)	Total
License 1.0 (BSL-1.0)	requiring preservation of copyright and license notices for source (and not binary) distribution. Licensed works, modifications, and larger works may be distributed under different terms and without source code.	.github.com/licenses/bsl-1.0							
Creative Commons Zero v1.0 Universal (CC0-1.0)	The Creative Commons CC0 Public Domain Dedication waives copyright interest in a work you've created and dedicates it to the world-wide public domain. Use CC0 to opt out of copyright entirely and ensure your work has the widest reach. As with the Unlicense and typical software licenses, CC0 disclaims warranties. CC0 is very similar to the Unlicense	https://api.github.com/licenses/cc0-1.0	/	/	/	X	/	/	8
Eclipse Public License 2.0	This commercially-friendly copyleft license provides the ability to	https://api.github.co	/	/	/	/	/	X	6

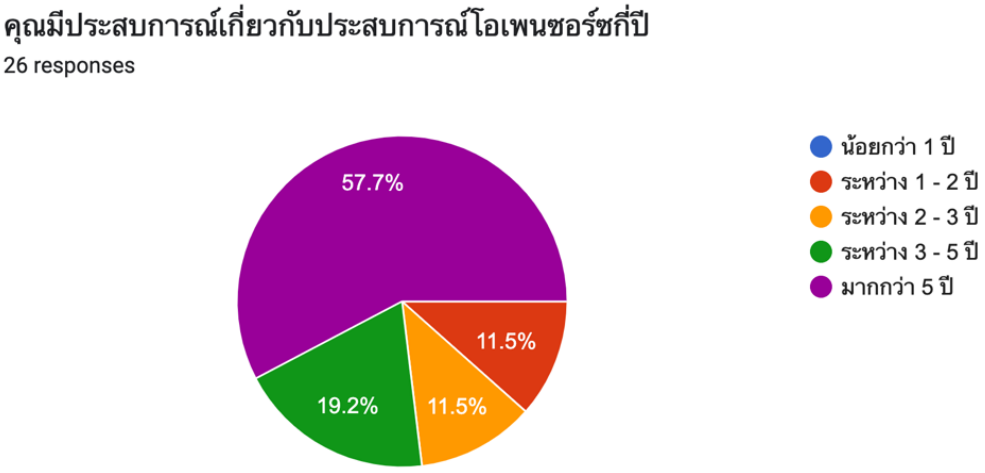
License Type	Description	GitHub Reference	Commercial use (1)	Distribution (1)	Modification (1)	Patent use (1)	Private use (2)	Closed source project use (3)	Total
(EPL-2.0)	commercially license binaries; a modern royalty-free patent license grant; and the ability for linked works to use other licenses, including commercial ones.	m/licenses /epl-2.0							
GNU General Public License v2.0 (GPL-2.0)	The GNU GPL is the most widely used free software license and has a strong copyleft requirement. When distributing derived works, the source code of the work must be made available under the same license. There are multiple variants of the GNU GPL, each with different requirements.	https://api .github.co m/licenses /gpl-2.0	/	/	/	X	/	X	5
GNU General Public License v3.0 (GPL-3.0)	Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright	https://api .github.co m/licenses /gpl-3.0	/	/	/	/	/	X	6

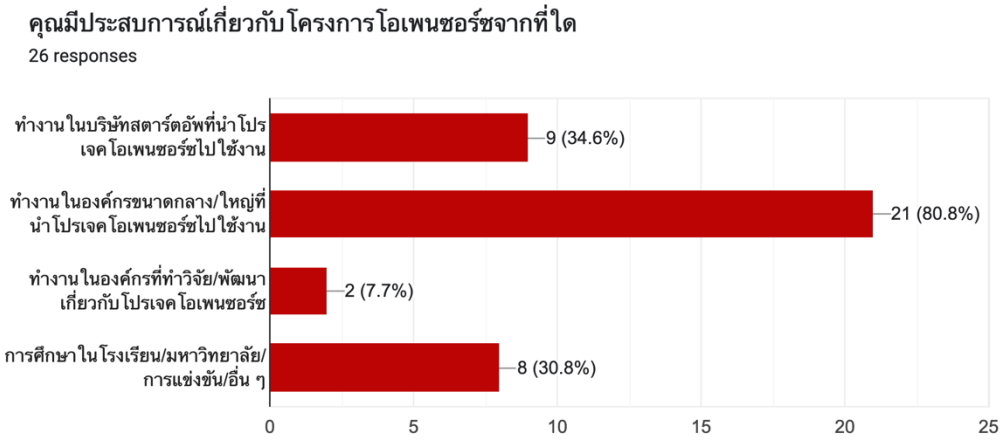
License Type	Description	GitHub Reference	Commercial use (1)	Distribution (1)	Modification (1)	Patent use (1)	Private use (2)	Closed source project use (3)	Total
	and license notices must be preserved. Contributors provide an express grant of patent rights.								
GNU Lesser General Public License v2.1 (LGPL-2.1)	Primarily used for software libraries, the GNU LGPL requires that derived works be licensed under the same license, but works that only link to it do not fall under this restriction. There are two commonly used versions of the GNU LGPL.	https://api.github.com/licenses/lgpl-2.1	/	/	/	X	/	X	5
MIT License (MIT)	A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.	https://api.github.com/licenses/mit	/	/	/	X	/	/	8
Mozilla Public	Permissions of this weak copyleft	https://api	/	/	/	/	/	X	6

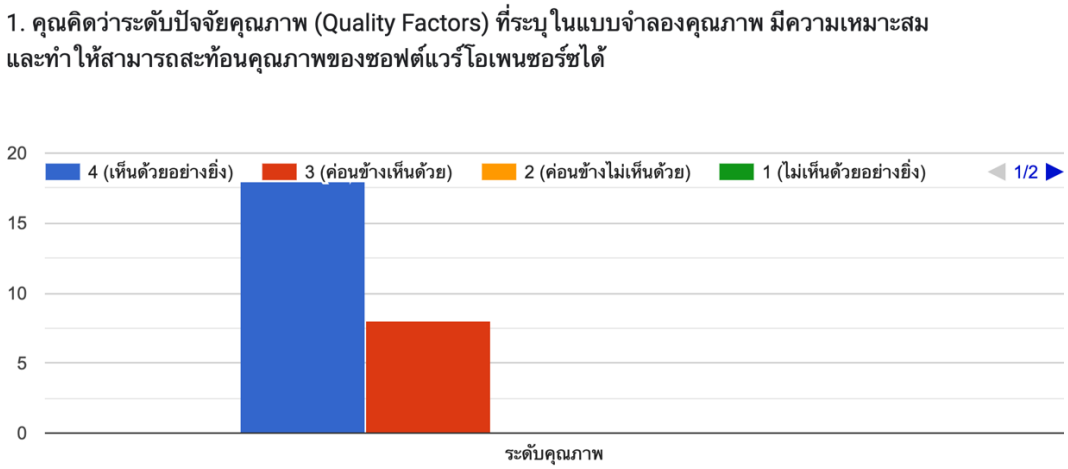
License Type	Description	GitHub Reference	Commercial use (1)	Distribution (1)	Modification (1)	Patent use (1)	Private use (2)	Closed source project use (3)	Total
License 2.0 (MPL-2.0)	license are conditioned on making available source code of licensed files and modifications of those files under the same license (or in certain cases, one of the GNU licenses). Copyright and license notices must be preserved. Contributors provide an express grant of patent rights. However, a larger work using the licensed work may be distributed under different terms and without source code for files added in the larger work	.github.com/licenses/mpl-2.0							
Unlicense	A license with no conditions whatsoever which dedicates works to the public domain. Unlicensed works, modifications, and larger works may be distributed under different terms and without source code.	https://api.github.com/licenses/unlicense	/	/	/	X	/	/	8

ภาคผนวก ง

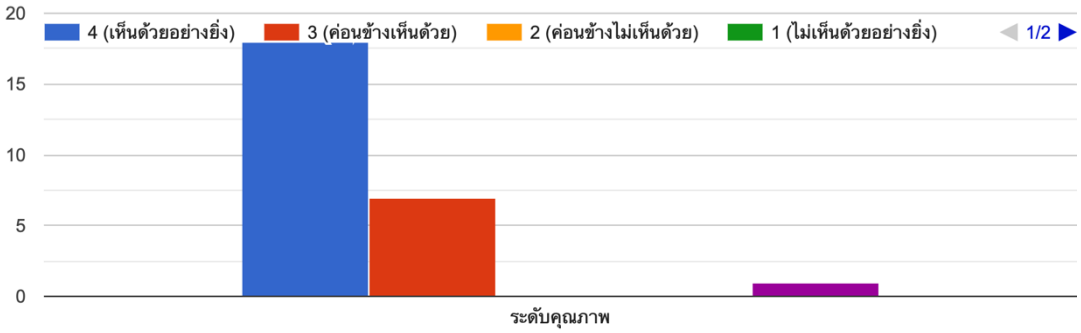
ผลลัพธ์แบบสอบถามความคิดเห็นต่อแบบจำลองคุณภาพฯ



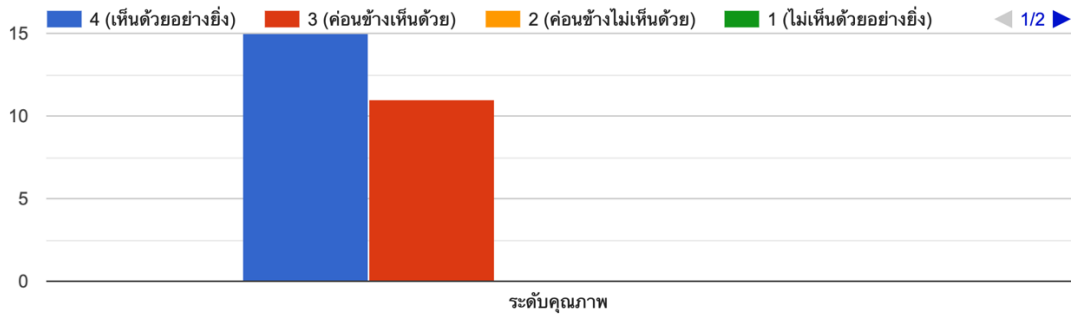




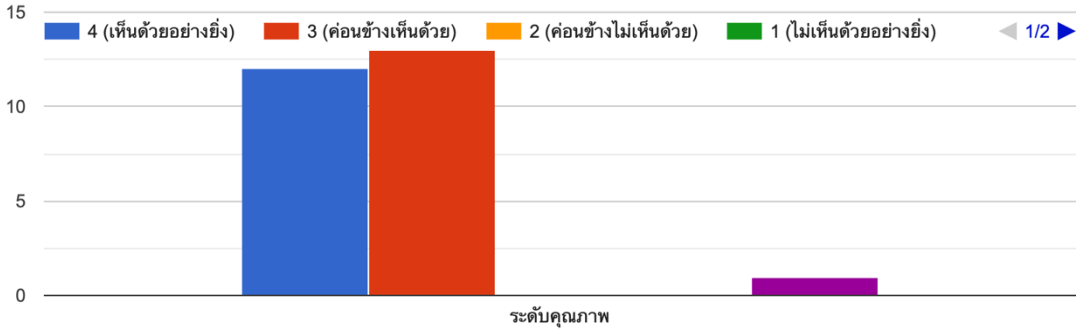
2. คุณคิดว่าระดับปัจจัยคุณภาพ (Quality Factors) ที่ระบุในแบบจำลองคุณภาพ ครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้



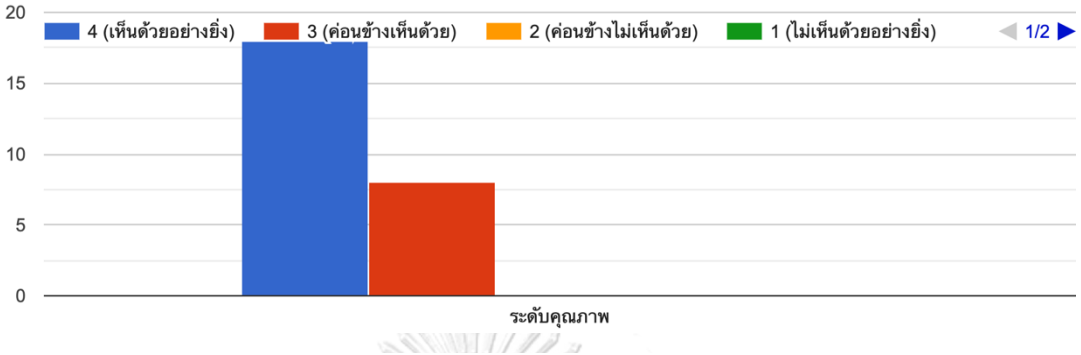
3. คุณคิดว่าระดับปัจจัยคุณภาพย่อย (Sub-quality factors) ที่ระบุในแบบจำลองคุณภาพ มีความเหมาะสมทำให้สามารถสะท้อนคุณภาพของระดับก่อนหน้าได้



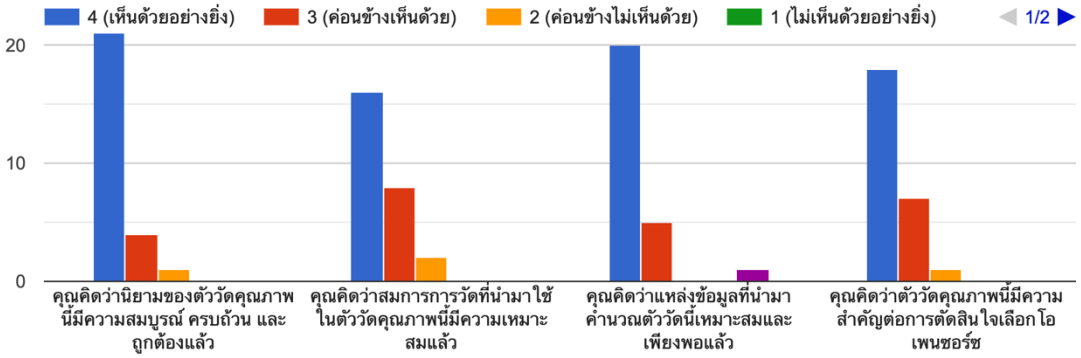
4. คุณคิดว่าระดับปัจจัยคุณภาพย่อย (Sub-quality factors) ที่ระบุในแบบจำลองคุณภาพ ครบถ้วนและเพียงพอแล้ว ไม่จำเป็นต้องเพิ่มเติมจากนี้



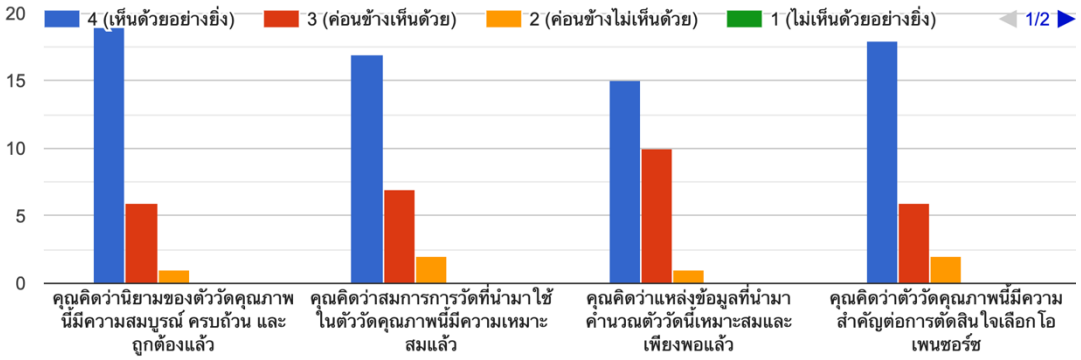
5. คุณคิดว่าค่าคะแนนที่เป็นผลลัพธ์จากแบบจำลองคุณภาพ (Open Source Quality) จากวิธีการคำนวณที่ระบุไว้สามารถสะท้อนคุณภ...รถนำไปเปรียบเทียบกับซอฟต์แวร์โอเพนซอร์ซอื่น ๆ ได้



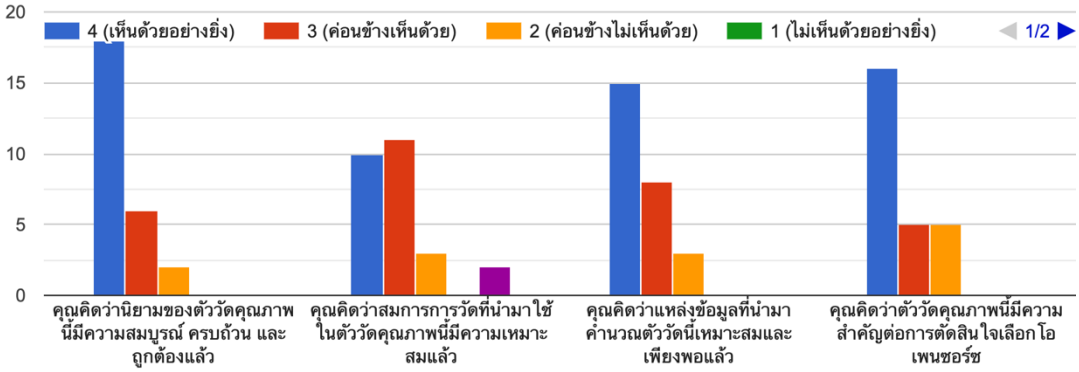
V1 - OSS License



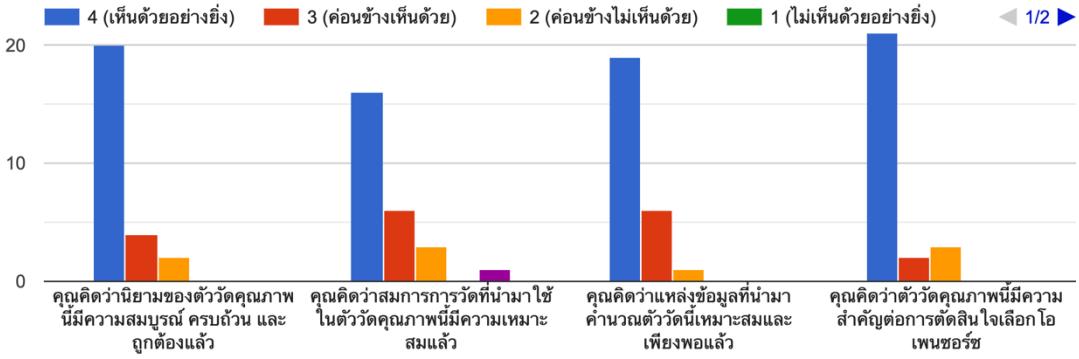
V2 - Size of Community



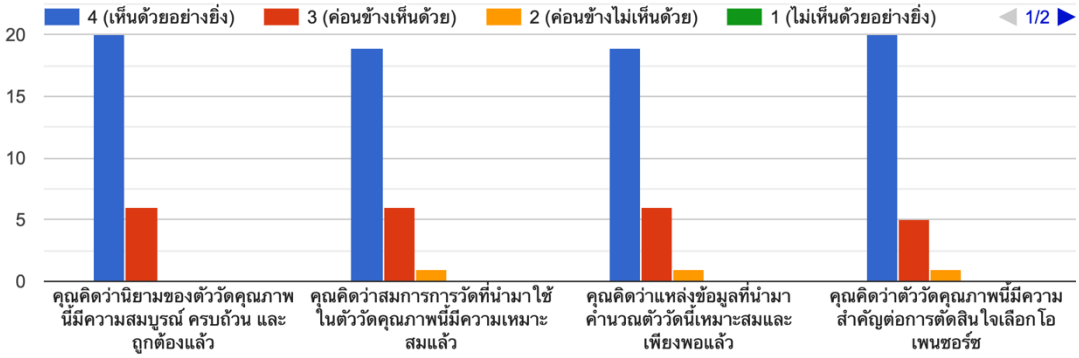
V3 - Q&A Volume



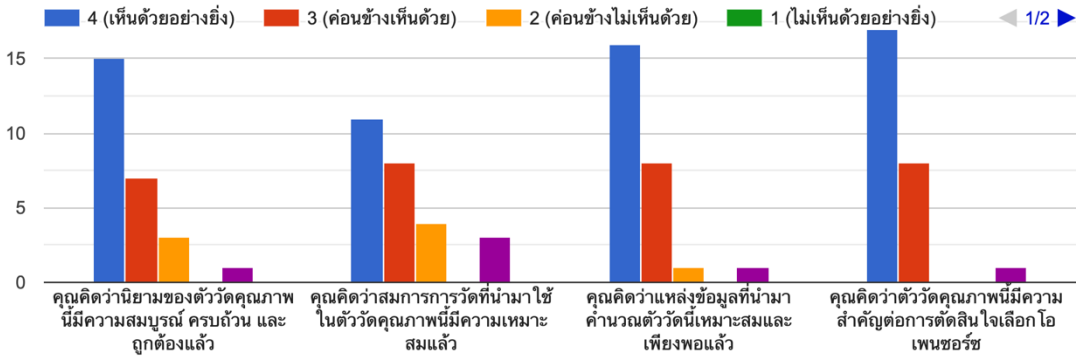
V4 - Number of Support Contributors



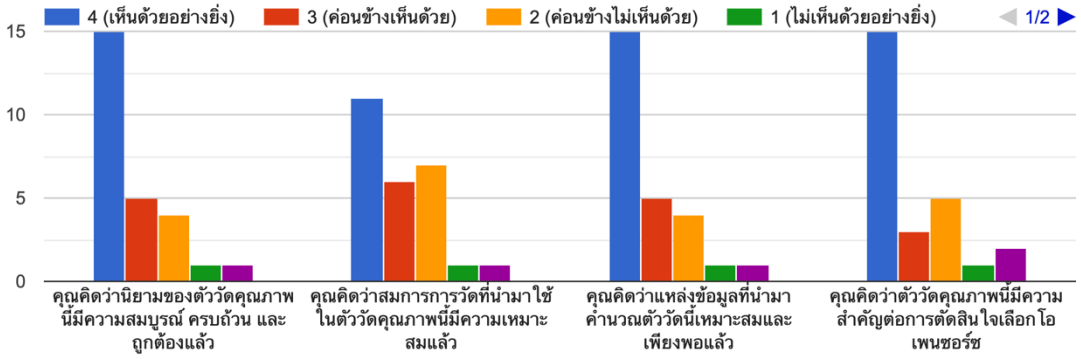
V5 - Support Activity



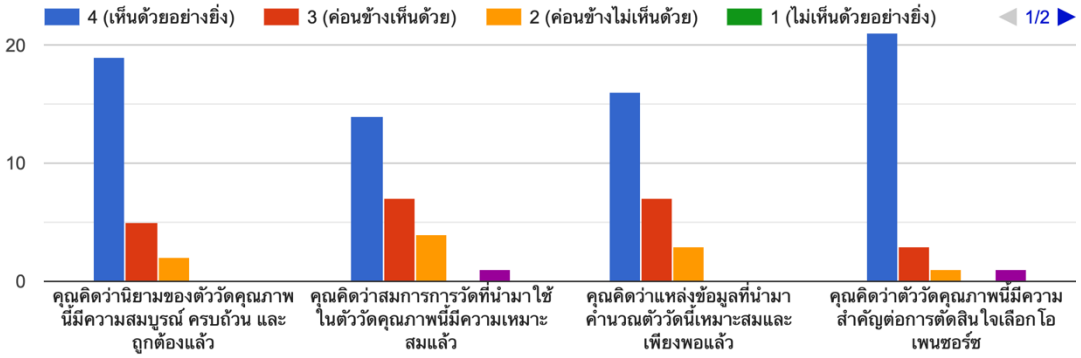
V6 - Maturity Level



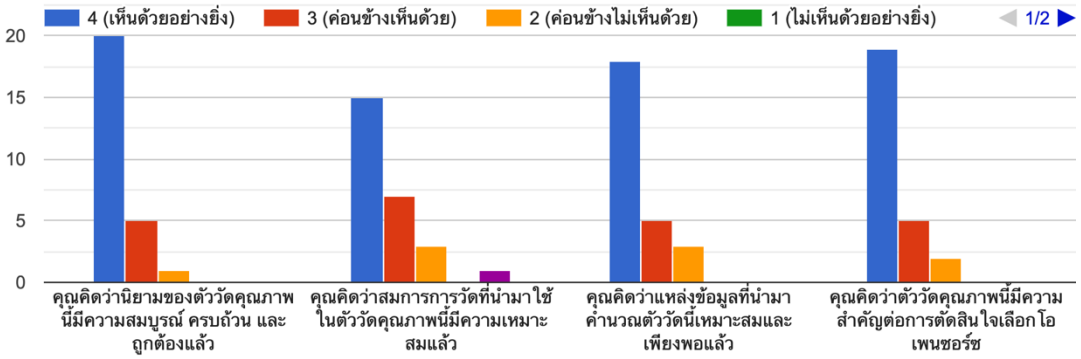
V7 - Development Language Popularity



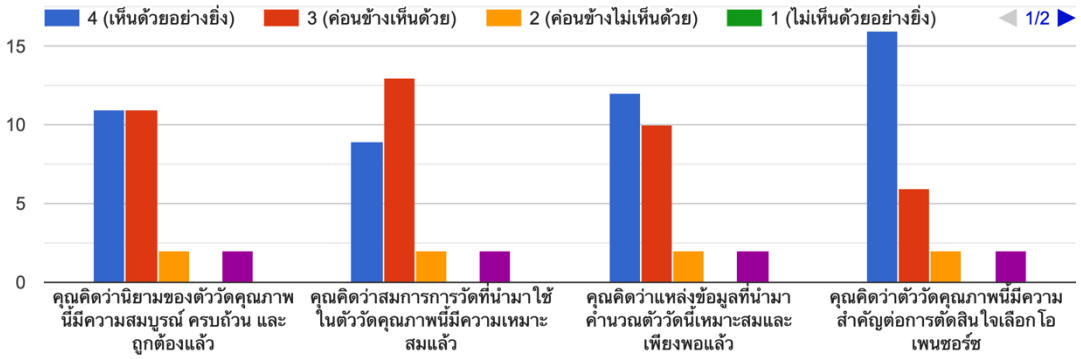
V8 - Code Documentation



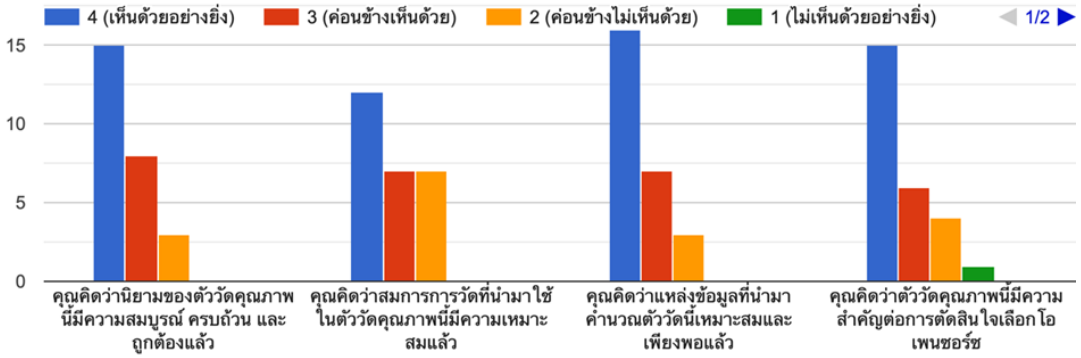
V9 - Learning Materials



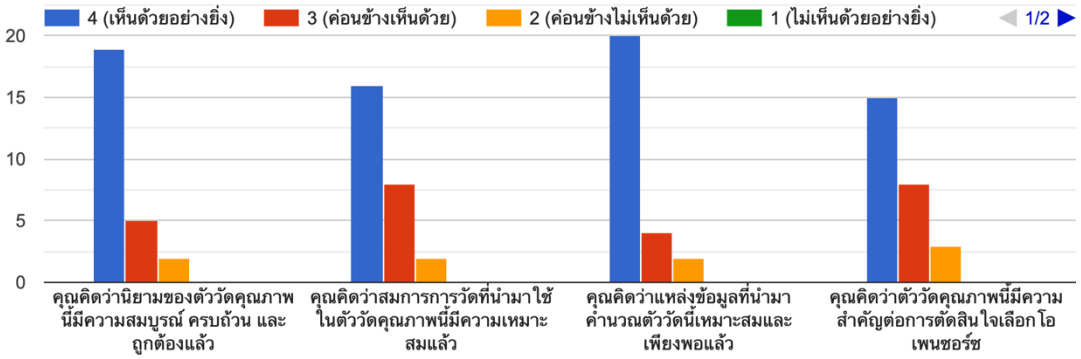
V10 - Cost of Ownership Reduction



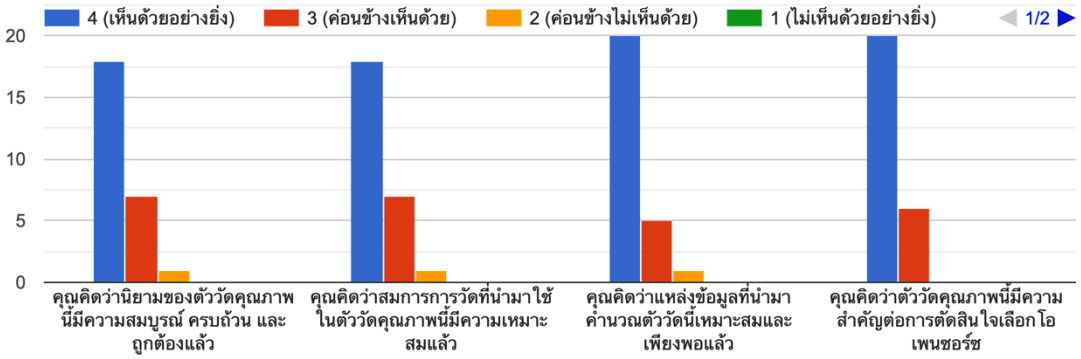
V11 - New Features Focus



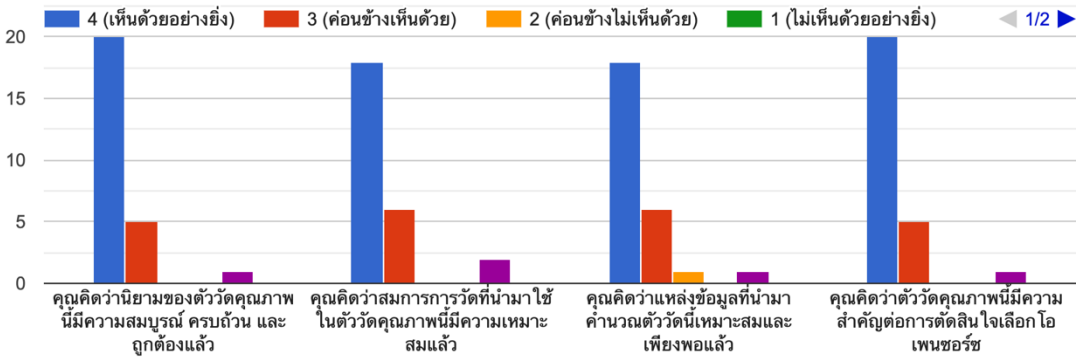
V12 - Continuing Change



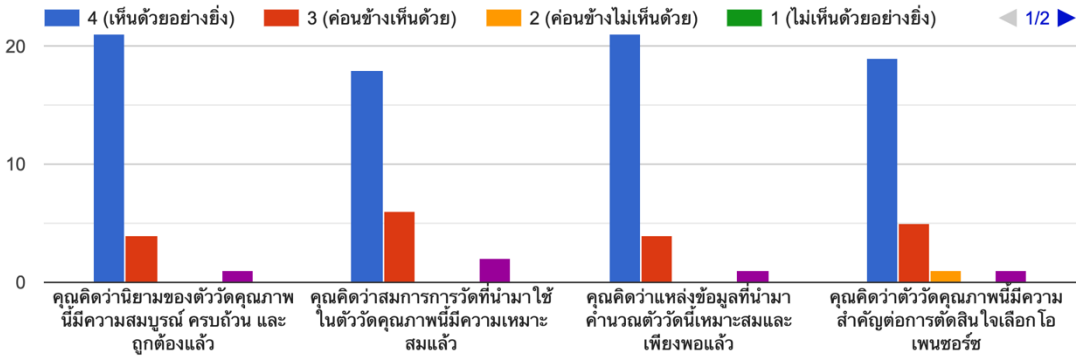
V13 - Code Quality Level



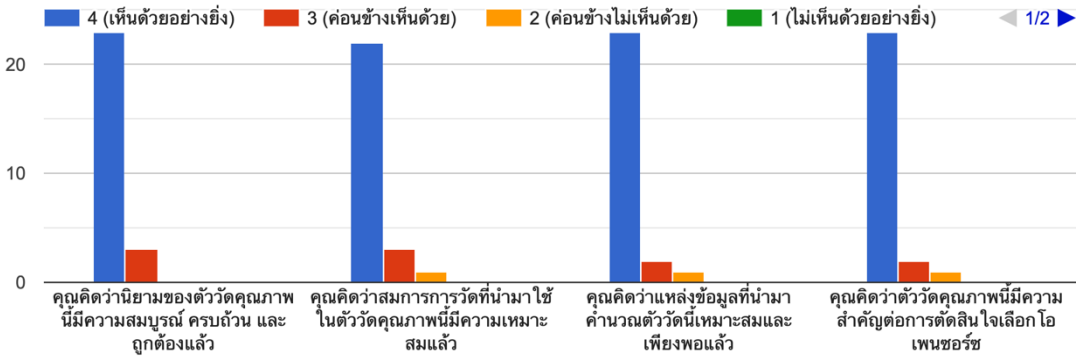
V14 - Reliability Level



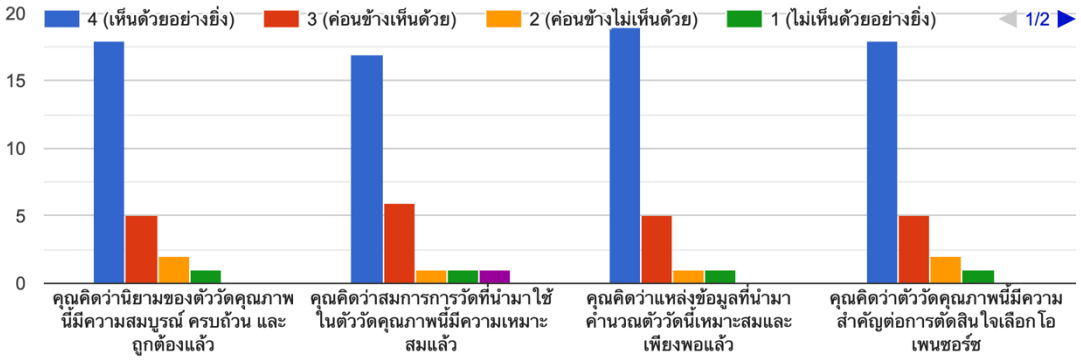
V15 - Maintainability Level



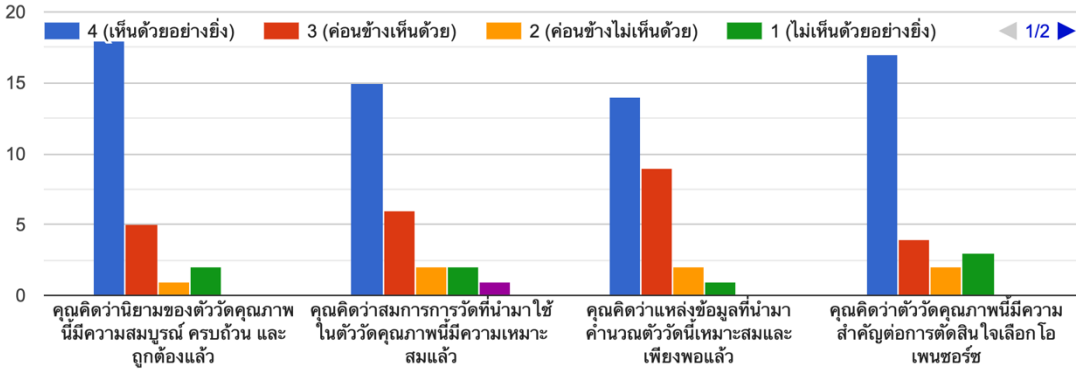
V16 - Security Level



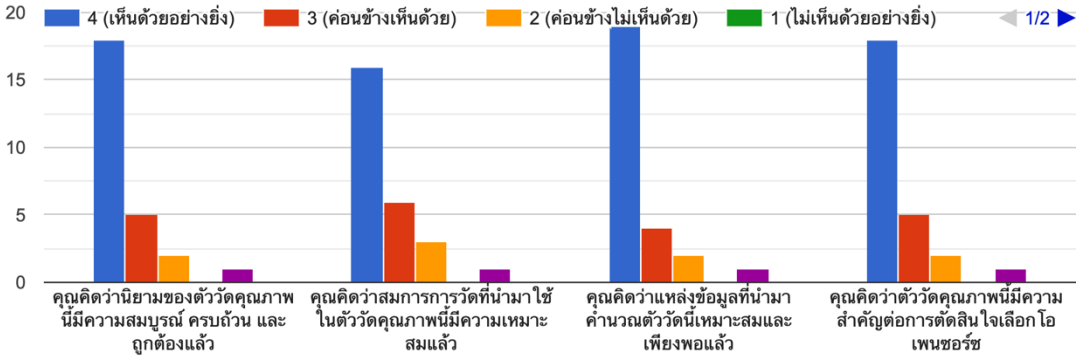
V17 - Testability Level



V18 - Co-existence



V19 - Lack of Performance Issues Level



ตารางที่ ง.1 ข้อมูลหน่วยทดลอง

ผู้สัมภาษณ์	บริษัทกลาง/ใหญ่	บริษัทสตาร์ตอัพ	ทั่วไป	พัฒนาโอเพนซอร์ซ	ประสบการณ์
คนที่ 1	✓		✓		2-3 ปี
คนที่ 2		✓			2-3 ปี
คนที่ 3		✓			> 5 ปี
คนที่ 4	✓		✓		> 5 ปี
คนที่ 5	✓			✓	> 5 ปี
คนที่ 6		✓			> 5 ปี
คนที่ 7		✓			> 5 ปี
คนที่ 8	✓				> X ปี
คนที่ 9	✓				3-5 ปี
คนที่ 10	✓		✓		> 5 ปี
คนที่ 11		✓			1-2 ปี
คนที่ 12	✓		✓		1-2 ปี
คนที่ 13	✓				3-5 ปี
คนที่ 14	✓				2-3 ปี
คนที่ 15	✓				> 5 ปี
คนที่ 16	✓	✓			> 5 ปี
คนที่ 17	✓				3-5 ปี
คนที่ 18	✓		✓		3-5 ปี
คนที่ 19	✓				1-2 ปี

ตารางที่ ง.1 ข้อมูลหน่วยทดลอง (ต่อ)

ผู้สัมภาษณ์	บริษัทกลาง/ใหญ่	บริษัทสตาร์ตอัพ	ทั่วไป	พัฒนาโอเพนซอร์ซ	ประสบการณ์
คนที่ 20	✓		✓		> 5 ปี
คนที่ 21	✓	✓	✓		> 5 ปี
คนที่ 22	✓				3-5 ปี
คนที่ 23	✓	✓			> 5 ปี
คนที่ 24	✓				> 5 ปี
คนที่ 25	✓			✓	> 5 ปี
คนที่ 26	✓	✓	✓		> 5 ปี
รวม	**21**	**9**	**8**	**2**	

ประวัติผู้เขียน

ชื่อ-สกุล	นายอัรกอม มะแดเฮาะ
วัน เดือน ปี เกิด	06 พฤษภาคม 2537
สถานที่เกิด	จังหวัดยะลา
วุฒิการศึกษา	วิทยาศาสตรบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ มหาวิทยาลัยวลัยลักษณ์
ที่อยู่ปัจจุบัน	101 หมู่ 3 ตำบลยะต๊ะ อำเภอรามัน จังหวัดยะลา รหัสไปรษณีย์ 95140