การพัฒนาซอฟต์แวร์ภาษาจาวาร่วมกันบนเว็บเบราว์เซอร์ โดยใช้การล็อคพื้นที่การทำงานเพื่อป้องกันความขัดแย้งเชิงความหมาย

นายชลวิศว์ เสภาศีราภรณ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2557

COLLABORATIVE JAVA SOFTWARE DEVELOPMENT ON A WEB BROWSER USING LOCKING TO PREVENT SEMANTIC CONFLICTS

Chonlawit Sepasiraporn

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Computer Engineering

Suranaree University of Technology

Academic Year 2014

การพัฒนาซอฟต์แวร์ภาษาจาวาร่วมกันบนเว็บเบราว์เซอร์ โดยใช้การล็อคพื้นที่การทำงานเพื่อป้องกันความขัดแย้งเชิงความหมาย

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

_____

(รศ. ดร.กิตติศักดิ์ เกิดประสพ)
ประธานกรรมการ

_____

(ผศ. ดร.พิชโยทัย มหัทธนาภิวัฒน์)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

_____

(ผศ. ดร.ชาญวิทย์ แก้วกสิ)
กรรมการ

_____

(ศ. ดร.ชูกิจ ลิมปิจำนงค์)
รองอธิการบดีฝ่ายวิชาการและนวัตกรรม

_____

(รศ. ร.อ. ดร.กนต์ธร ชำนิประศาสน์)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

ชลวิศว์ เสภาศีราภรณ์ : การพัฒนาซอฟต์แวร์ภาษาจาวาร่วมกันบนเว็บเบราว์เซอร์ โดยใช้การล็อคพื้นที่การทำงานเพื่อป้องกันความขัดแย้งเชิงความหมาย (COLLABORATIVE JAVA SOFTWARE DEVELOPMENT ON A WEB BROWSER USING LOCKING TO PREVENT SEMANTIC CONFLICTS) อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.พิชโยทัย มหัทธนาภิวัฒน์, 77 หน้า

ในปัจจุบันการพัฒนาซอฟต์แวร์ร่วมกันกำลังเป็นที่นิยมอย่างแพร่หลาย เนื่องจากการพัฒนาซอฟต์แวร์ร่วมกันนั้นมีประโยชน์ในด้านของคุณภาพ กล่าวได้ว่าเมื่อมีนักพัฒนามากกว่าหนึ่งคนที่ช่วยแก้ไขและเสนอความคิดเห็นเกี่ยวกับปัญหาที่พบจะสามารถช่วยเพิ่มคุณภาพขึ้นได้ ปัจจุบันการพัฒนาซอฟต์แวร์ร่วมกันจะทำกันบนเครื่องมือการพัฒนาซอฟต์แวร์สำเร็จรูป ซึ่งอาจพบปัญหาในการตั้งค่าซอฟต์แวร์เพื่อให้ตรงกันกับผู้ร่วมพัฒนาคนอื่น และด้วยการพัฒนาซอฟต์แวร์ร่วมกันอาจพบปัญหาความขัดแย้งเชิงความหมายที่อาจเกิดขึ้นจากนักพัฒนาหลายคนที่มีความคิดในการแก้ปัญหาบางอย่างไม่ตรงกัน

ผู้วิจัยได้ทำการศึกษาและพัฒนาระบบต้นแบบสำหรับการพัฒนาซอฟต์แวร์ภาษาจาวาร่วมกันบนเว็บเบราว์เซอร์ โดยได้ทำการพัฒนาให้ระบบต้นแบบสามารถล็อคพื้นที่การทำงานได้อย่างอัตโนมัติ เพื่อป้องกันความขัดแย้งเชิงความหมาย และเมื่อทดสอบความถูกต้องของการล็อคพื้นที่การทำงานในระบบต้นแบบด้วยเครื่องมือทดสอบแบบอัตโนมัติแล้ว พบว่าระบบต้นแบบมีความถูกต้องในการล็อคพื้นที่การทำงานมากถึง 98.18%

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ลายมือชื่อนักศึกษา_____

ปีการศึกษา 2557 ลายมือชื่ออาจารย์ที่ปรึกษา_____

CHONLAWIT SEPASIRAPORN : COLLABORATIVE JAVA SOFTWARE DEVELOPMENT ON A WEB BROWSER USING LOCKING TO PREVENT SEMANTIC CONFLICTS. THESIS ADVISOR : ASST. PROF. PICHAYOTAI MAHATTHANAPIWAT, Ph.D., 77 PP.

COLLABORATIVE/LOCKING/SEMANTIC CONFLICTS/PARSER

At present, the collaborative software development has become widely popular because of its benefits in terms quality. The quality of software will be better from resolved issues and comments of the development team. The collaborative software development is currently done on desktop-based development tools. However, the problem of software setting may be occurred for developers in the team. The problem of semantic conflicts is likely occurred from different views to resolve problems.

The work reported in this thesis is a study and a development of a prototype system for collaborative Java software development on Web browser using automatically locking areas of work to prevent semantic conflicts. The accuracy of locking areas in prototype system is 98.18 percent, when testing with automatic testing tool.

School of Computer Engineering Student's Signature _____

Academic Year 2014 Advisor's Signature_____

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี เนื่องจากได้รับการสนับสนุนจากบุคคลและกลุ่มบุคคลดังต่อไปนี้ที่ได้ให้คำปรึกษา ช่วยเหลือ และแนะนำ ในการดำเนินงานวิจัยจนประสบความสำเร็จ

ผู้ช่วยศาสตราจารย์ ดร.พิชโยทัย มหัทธนาภิวัฒน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้การสนับสนุน การแนะนำแนวทางในการพัฒนา ศึกษา และช่วยเหลือในด้านต่าง ๆ การแนะนำแนวทางในการเขียนบทความ และวิทยานิพนธ์ รวมถึงการตรวจทานแก้ไขเนื้อหาของบทความและวิทยานิพนธ์จนสำเร็จ

ผู้ช่วยศาสตราจารย์ ดร.ชาญวิทย์ แก้วกสิ อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ให้คำแนะนำทางด้านการพัฒนาตัวแจงส่วน และการใช้เครื่องมือสำหรับช่วยสร้างซอฟต์แวร์การจัดการข้อมูลบนเว็บเบราว์เซอร์

คุณศุภกฤษฎิ์ ตั้งเสริมสิทธิ์ นักศึกษาปริญญาโท สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ให้การสนับสนุน และช่วยเหลือทางด้านการใช้เครื่องมือสำหรับช่วยสร้างซอฟต์แวร์การจัดการข้อมูลบนเว็บเบราว์เซอร์

คุณนริศรา ธาระพุทธ นักศึกษาปริญญาโท สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ให้การสนับสนุน ช่วยเหลือทางด้านการจัดทำและการตรวจทานเล่มวิทยานิพนธ์จนสำเร็จ

คุณกัลญา พับโพธิ์ เจ้าหน้าที่บริหารงานทั่วไป สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ให้การสนับสนุนและให้การช่วยเหลือทางด้านเอกสารที่เกี่ยวข้องมาโดยตลอด

สุดท้ายนี้ผู้วิจัยขอกราบขอบพระคุณบิดา มารดา ที่ให้การอุปการะ อบรมเลี้ยงดู ตลอดจนส่งเสริมทางด้านการศึกษา และให้กำลังใจเป็นอย่างดีมาโดยตลอด จนกระทั่งวิทยานิพนธ์ฉบับนี้สำเร็จ

ชลวิศว์ เสภาศีราภรณ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

ในอดีตการพัฒนาซอฟต์แวร์ด้วยนักพัฒนาเพียงคนเดียวนั้น จะทำให้ได้ซอฟต์แวร์ที่ได้มีคุณภาพค่อนข้างน้อย เนื่องจากอาจมีความผิดพลาดในการพัฒนาซอฟต์แวร์เกิดขึ้นบ่อย เมื่อมีความผิดพลาดเกิดขึ้นนักพัฒนาจำเป็นต้องใช้เวลากลับไปแก้ไขความผิดพลาดเหล่านั้น ทำให้การพัฒนาซอฟต์แวร์จำเป็นที่ต้องใช้เวลาเพิ่มขึ้นเพื่อให้ได้ซอฟต์แวร์ที่มีคุณภาพ และหากเป็นการพัฒนาซอฟต์แวร์ที่มีความซับซ้อนด้วยแล้ว อาจจะต้องใช้เวลาเพิ่มมากขึ้นอีกในการพัฒนาซอฟต์แวร์ดังกล่าวให้มีคุณภาพ (Duque and Bravo, 2008)

ปัจจุบันผู้ใช้บริการซอฟต์แวร์มีความต้องการซอฟต์แวร์ที่มีความซับซ้อนมากขึ้น การพัฒนาซอฟแวร์ร่วมกันจึงเป็นทางเลือกหนึ่งที่ช่วยในการพัฒนาซอฟต์แวร์ดังกล่าว โดยวิธีการพัฒนาซอฟต์แวร์ร่วมกันมีด้วยกันหลายวิธี หนึ่งในนั้นคือการพัฒนาซอฟต์แวร์ร่วมกันผ่านทางระบบเครือข่าย (Fan and Sun, 2012a) ซึ่งระบบเครือข่ายในปัจจุบันสามารถเข้าถึงได้ทุกเวลาทุกสถานที่ ทำให้นักพัฒนาสามารถพัฒนาซอฟต์แวร์ได้ตลอดเมื่อต้องการ และยังสามารถทำงาน ได้พร้อมกันหลายคน นักพัฒนาเหล่านั้นสามารถแก้ไข เปลี่ยนแปลง และเพิ่มเติมส่วนต่าง ๆ ของรหัสต้นฉบับ (Source code) ได้ทันทีที่ต้องการ สิ่งเหล่านี้ช่วยเพิ่มคุณภาพให้กับซอฟต์แวร์มากขึ้น (Goldman, 2011) ทำให้การพัฒนาซอฟต์แวร์ด้วยวิธีการดังกล่าวกำลังเป็นที่นิยมอย่างมาก

เครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์จำนวนมากมีการสนับสนุนการพัฒนาซอฟต์แวร์ร่วมกัน หนึ่งในนั้นคือ เว็บเบราว์เซอร์ ซึ่งในปัจจุบันเว็บเบราว์เซอร์มีอยู่เป็นจำนวนมากให้ใช้งานได้โดยไม่เสียค่าใช้จ่ายใด ๆ ทำให้นักพัฒนาที่พัฒนาซอฟต์แวร์บนเว็บเบราว์เซอร์นั้นทำงานสะดวกมากขึ้น เนื่องจากไม่จำเป็นต้องติดตั้งเครื่องมือเพิ่มเติมก็สามารถพัฒนาซอฟต์แวร์ร่วมกันได้ ซึ่งทำให้สามารถพัฒนาซอฟต์แวร์ได้ทุกเวลาทุกสถานที่เพียงมีคอมพิวเตอร์ โดยมีเงื่อนไขเพียงแค่คอมพิวเตอร์เครื่องนั้นจะต้องมีเว็บเบราว์เซอร์และสามารถเชื่อมต่อกับเครือข่ายได้ อย่างไรก็ตามในการใช้ซอฟต์แวร์จัดการข้อมูลบนเว็บเบราว์เซอร์เพื่อให้สามารถพัฒนาซอฟต์แวร์ร่วมกันได้จำเป็นต้องมีการสร้างตัวแจงส่วน (Parser) (Gesser, 2010) เพื่อให้สามารถจัดการกับไวยากรณ์ของภาษาที่ใช้ในการพัฒนาซอฟต์แวร์ ตัวแจงส่วนที่ใช้งานบนเว็บเบราว์เซอร์ได้จำเป็นต้องอยู่ในรูปแบบของภาษาจาวาสคริปต์ (JavaScript) (Carter, 2013) เพื่อให้สามารถทำงานได้บนทุก

เว็บเบราว์เซอร์ โดยเครื่องมือที่นำมาใช้ในการสร้างตัวแจงส่วนคือ PEG.js ที่พัฒนาโดย David Majda (2013a) เป็นเครื่องมือที่มีความสามารถในการสร้างตัวแจงส่วนในรูปแบบไฟล์จาวาสคริปต์ด้วยวิธีการป้อนไวยากรณ์ของภาษาเขียนโปรแกรมเข้าไป

การพัฒนาซอฟต์แวร์ร่วมกันยังมีปัญหาที่สำคัญที่อาจเกิดขึ้นในขั้นตอนของการพัฒนา คือ ความขัดแย้งเชิงความหมายของเหล่านักพัฒนาที่ทำงานร่วมกัน โดยความขัดแย้งเชิงความหมายเกิดจากการที่นักพัฒนาหลายคนเห็นถึงปัญหาของรหัสต้นฉบับหนึ่งในเวลาเดียวกัน แต่มีความคิดและวิธีในการแก้ไขปัญหาที่แตกต่างกันออกไป ทำให้เมื่อนักพัฒนาแต่ละคนแก้ปัญหาด้วยวิธีของตัวเอง ซอฟต์แวร์จึงอาจเกิดข้อผิดพลาดขึ้น ปัญหาดังกล่าวสามารถแก้ไขได้ด้วยวิธีการล็อคพื้นที่การทำงาน ซึ่งเป็นการล็อคพื้นที่การทำงานของนักพัฒนาคนนั้น ๆ ในขณะที่กำลังทำงานอยู่ โดยนักพัฒนาที่เข้าไปก่อนเท่านั้นจึงจะมีสิทธิ์ในการแก้ไขรหัสต้นฉบับในส่วนดังกล่าว และหากนักพัฒนาคนอื่นต้องการทำงานในส่วนเดียวกัน จะต้องรอให้นักพัฒนาที่ทำงานในส่วนนั้นทำงานเสร็จเรียบร้อยก่อน หลังจากนั้นจะต้องทำการปลดล็อคพื้นที่การทำงานดังกล่าว และนักพัฒนาคนอื่นจึงจะสามารถเข้าไปทำงานในส่วนนั้นได้ (Fan and Sun, 2012b)

จากข้อมูลความสามารถและปัญหาในการพัฒนาซอฟต์แวร์ร่วมกัน เป็นเหตุผลหลักที่ทำให้ผู้วิจัยเลือกศึกษาและพัฒนาเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ร่วมกันบนเว็บเบราว์เซอร์ โดยมุ่งเน้นไปที่การเพิ่มความถูกต้องของรหัสต้นฉบับในการพัฒนาซอฟต์แวร์ร่วมกัน และการป้องกันปัญหาความขัดแย้งเชิงความหมาย ซึ่งเครื่องมือที่ใช้ในการสร้างตัวแจงส่วนจากไวยากรณ์ของภาษาที่ต้องการ ให้อยู่ในรูปแบบของตัวแจงส่วนจาวาสคริปต์ ผู้วิจัยเลือกใช้ PEG.js เพราะเป็นเครื่องมือที่สามารถพัฒนาตัวแจงส่วนในรูปแบบไฟล์จาวาสคริปต์ได้ง่าย (Majda, 2013a) รวมถึงใช้วิธีการล็อคพื้นที่การทำงานเข้ามาช่วยในการแก้ไขปัญหาเรื่องความขัดแย้งเชิงความหมาย ซึ่งงานวิจัยนี้ได้เลือกใช้ภาษาจาวา (Java) ในการทดสอบความถูกต้องของระบบต้นแบบ (Duque and Bravo, 2008 ; Goldman, 2011 ; Fan and et al., 2012 ; Fan and Sun, 2012a ; Fan and Sun 2012b) เหตุผลที่เลือกใช้ภาษาจาวาเพราะเป็นภาษาที่รู้จักและเป็นที่นิยมกันอย่างกว้างขวางในการพัฒนาซอฟต์แวร์ โดยในการศึกษาครั้งนี้ได้ทำการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา และทดสอบการป้องกันความขัดแย้งเชิงความหมายด้วยวิธีการล็อคพื้นที่การทำงานแบบอัตโนมัติ

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์หลักเพื่อให้นักพัฒนาสามารถพัฒนาซอฟต์แวร์ร่วมกันบนเว็บเบราว์เซอร์ ที่มีการล็อคพื้นที่การทำงานแบบอัตโนมัติ ซึ่งระบบต้นแบบสำหรับการพัฒนา

ซอฟต์แวร์ร่วมกันจะต้องสามารถป้องกันความขัดแย้งเชิงความหมายที่เกิดขึ้นจากทำงานร่วมกันได้ โดยมีวัตถุประสงค์ย่อยดังต่อไปนี้

1.2.1 เพื่อศึกษาแนวคิดในการพัฒนาซอฟต์แวร์ร่วมกันบนเว็บเบราว์เซอร์ และการป้องกันความขัดแย้งเชิงความหมายที่อาจเกิดขึ้นจากผู้พัฒนาซอฟต์แวร์

1.2.2 เพื่อศึกษาและพัฒนาตัวแจงส่วนภาษาจาวา โดยตัวแจงส่วนภาษาจาวานั้นจะต้องอยู่ในรูปแบบของไฟล์จาวาสคริปต์ ทั้งนี้เพื่อให้สามารถนำไปใช้บนเว็บเบราว์เซอร์ได้

1.2.3 เพื่อศึกษาวิธีการล็อคพื้นที่การทำงานที่ผู้พัฒนาแต่ละคนกำลังทำงานอยู่ เพื่อป้องกันความขัดแย้งเชิงความหมายที่อาจเกิดขึ้น

1.2.4 เพื่อศึกษาประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา

1.2.5 เพื่อประเมินความถูกต้องของระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน

1.3 สมมุติฐานการวิจัย

จากวัตถุประสงค์หลักและวัตถุประสงค์ย่อยในการวิจัยครั้งนี้ ทำให้ผู้วิจัยสามารถตั้งสมมุติฐานของการวิจัยได้ดังนี้

1.3.1 การทำงานบนเว็บเบราว์เซอร์จะช่วยให้นักพัฒนาสามารถพัฒนาซอฟต์แวร์ได้ง่ายและสะดวกมากกว่า เนื่องจากไม่ต้องเรียนรู้การใช้งานเครื่องมือสำหรับการพัฒนาที่ไม่คุ้นเคย

1.3.2 สามารถพัฒนาให้ตัวแจงส่วนที่อยู่ในรูปแบบของไฟล์จาวาสคริปต์ ให้สามารถแจงส่วนภาษาจาวาได้ และสามารถนำไปใช้งานบนเว็บเบราว์เซอร์ได้จริง

1.3.3 วิธีการล็อคพื้นที่การทำงานจะสามารถช่วยป้องกันความขัดแย้งเชิงความหมายที่อาจเกิดขึ้นเมื่อนักพัฒนามีการพัฒนาซอฟต์แวร์ร่วมกัน

1.3.4 การล็อคพื้นที่การทำงานต้องสามารถทำงานได้อย่างอัตโนมัติ

1.4 ขอบเขตของการวิจัย

เพื่อให้การวิจัยเป็นไปตามวัตถุประสงค์ และเป็นไปตามสมมติฐานที่ได้ตั้งไว้ จำเป็นต้องมีการกำหนดขอบเขตของงานวิจัยอย่างชัดเจน ซึ่งผู้วิจัยได้กำหนดขอบเขตของการวิจัยไว้ดังนี้

1.4.1 ตัวแจงส่วนภาษาวาจาจะถูกฝังอยู่ในซอฟต์แวร์จัดการข้อมูลซึ่งจะพัฒนาให้สามารถทำงานได้บนกูเกิลโครม (Google Chrome) เท่านั้น (Google Inc., 2014)

1.4.2 ตัวแจงส่วนภาษาจาวาสคริปต์สามารถทำงานได้กับไวยากรณ์ภาษาจาวา รุ่นที่ 1.6 เท่านั้น (Sun Microsystems Inc., 2005)

1.4.3 การล็อคพื้นที่การทำงานจะสามารถล็อคได้ทั้งระดับเมธอดและตัวแปรเป็นอย่างน้อย

1.5 ประโยชน์ที่คาดว่าจะได้รับ

งานวิจัยนี้มีวัตถุประสงค์ในการพัฒนาระบบต้นแบบเพื่อให้นักพัฒนาสามารถพัฒนาซอฟต์แวร์ร่วมกันบนเว็บเบราว์เซอร์ ที่มีการล็อคพื้นที่การทำงานแบบอัตโนมัติ ซึ่งจะสามารถป้องกันความขัดแย้งเชิงความหมายที่เกิดขึ้นจากทำงานร่วมกัน จากสมมุติฐานที่ตั้งไว้ทำให้สามารถคาดถึงประโยชน์ที่จะได้รับจากงานวิจัย ซึ่งประโยชน์ที่คาดว่าจะได้รับจากงานวิจัยมีดังนี้

1.5.1. เพื่อช่วยให้นักพัฒนาสามารถพัฒนาซอฟต์แวร์ร่วมกันได้ บนเว็บเบราว์เซอร์ผ่านระบบเครือข่าย

1.5.2. เพื่อช่วยป้องกันการเกิดความขัดแย้งเชิงความหมายขึ้น เมื่อนักพัฒนามีการพัฒนาซอฟต์แวร์ร่วมกัน

1.5.3. เพื่อให้การล็อคพื้นที่การทำงานเป็นไปอย่างอัตโนมัติ เพื่อให้นักพัฒนาสามารถทำการพัฒนารหัสต้นฉบับได้สะดวก

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะเป็นการนำเสนอปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง โดยในหัวข้อที่ 2.1 จะเป็นการอธิบายถึงความหมายของตัวแจงส่วน ลักษณะของเครื่องมือสร้างตัวแจงส่วน PEG.js และวิธีการใช้งานเบื้องต้นของเครื่องมือสร้างตัวแจงส่วน PEG.js หัวข้อที่ 2.2 เป็นการอธิบายถึงรูปแบบความขัดแย้งเชิงความหมายที่อาจเกิดขึ้นเมื่อมีการพัฒนาซอฟต์แวร์ร่วมกัน หัวข้อที่ 2.3 เป็นการอธิบายถึงวิธีการในการจัดการความขัดแย้งเชิงความหมาย และในหัวข้อที่ 2.4 เป็นงานวิจัยที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ร่วมกันและการป้องกันปัญหาความขัดแย้งเชิงความหมาย

2.1 ตัวแจงส่วน

ตัวแจงส่วน (Nuansri, 2011) คือขั้นตอนในการวิเคราะห์โครงสร้างของประโยคที่รับเข้ามาว่าถูกต้องตามกฎของไวยากรณ์ที่กำหนดไว้หรือไม่ ซึ่งตัวแจงส่วนจะมีอีกสิ่งที่คู่กันเสมอนั่นคือเครื่องกราดตรวจหรือตัววิเคราะห์คำ (Lexical analyzer) โดยตัววิเคราะห์คำจะทำหน้าที่เป็นตัวแยกข้อมูลประโยคที่เข้ามาออกเป็นชิ้น ๆ และจะเรียกแต่ละชิ้นว่าโทเค็น (Token) ซึ่งโทเค็นเหล่านี้จะถูกนำไปประมวลผลต่อด้วยตัวแจงส่วน หากข้อมูลประโยคที่รับเข้ามาไม่สามารถแยกออกเป็นโทเค็นได้ ตัววิเคราะห์คำจะแจ้งข้อผิดพลาดออกมาทันที เช่นเดียวกันหากข้อมูลประโยคที่รับเข้ามามีโครงสร้างไม่ตรงกับกฎของไวยากรณ์ที่กำหนดเอาไว้ ตัวแจงส่วนจะแจ้งข้อผิดพลาดออกมา ยกตัวอย่างกฎของไวยากรณ์ดังแสดงในรูปที่ 2.1 กฎของไวยากรณ์นี้เป็นชุดของไวยากรณ์ส่วนหนึ่ง

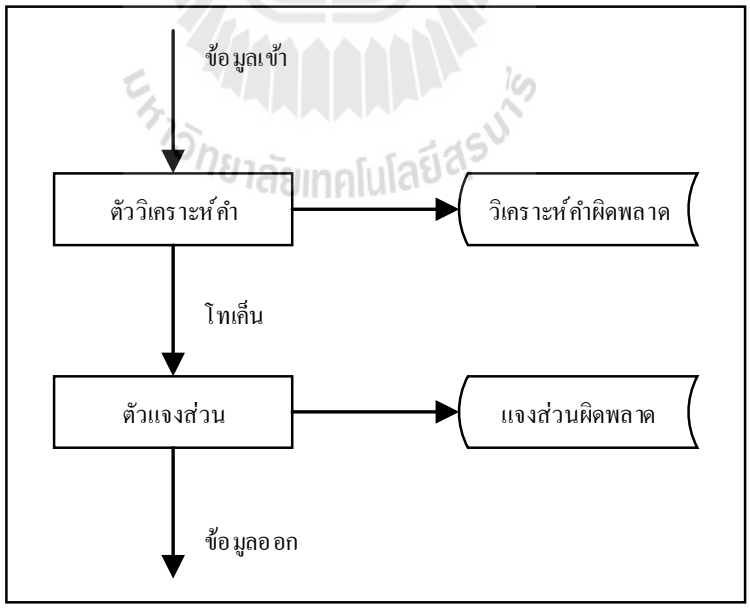
```
Expression ⟶ Integer * integer
Expression ⟶ Integer / integer
Expression ⟶ Integer + integer
Expression ⟶ Integer - integer
```

รูปที่ 2.1 ตัวอย่างกฎของไวยากรณ์

ที่เรียกว่า Expression ซึ่งชุดของไวยากรณ์เหล่านี้สามารถทำได้ทั้งการคูณ การหาร การบวก และการลบ ของเลขจำนวนเต็มบวกสองจำนวน กฎไวยากรณ์ดังกล่าวสามารถรับข้อมูลที่ป้อนเข้ามาในรูปแบบ 33 + 34 ซึ่งเป็นรูปแบบที่ถูกต้อง และหากข้อมูลที่รับเข้ามาไม่ตรงตามกฎของไวยากรณ์ เช่น 33 + 34 – 44 ตัวแจงส่วนจะแจ้งข้อผิดพลาดออกมาทันที โดยกฎของไวยากรณ์มีส่วนประกอบที่สำคัญ ๆ อยู่ 2 ส่วน ดังต่อไปนี้

1. Non-terminal symbols เป็นส่วนที่สามารถแยกออกเป็นส่วนประกอบย่อยได้จนถึง Terminal symbols ซึ่ง Non-terminal symbols มักจะอยู่ด้านซ้ายของกฎไวยากรณ์
2. Terminal symbols เป็นส่วนที่ไม่สามารถแยกย่อยต่อไปอีกได้ และจะไม่สามารถปรากฏอยู่ด้านซ้ายของกฎไวยากรณ์

ในส่วนการทำงานของตัววิเคราะห์คำเมื่อรับข้อมูลที่ตรงตามกฎของไวยากรณ์เข้ามา ตัวอย่างเช่น 33 + 34 ตัววิเคราะห์คำจะดำเนินการแยกข้อมูลที่เข้ามาออกเป็น 3 โทเค็น โทเค็นแรกคือ 33 โทเค็นที่สองคือ + และโทเค็นสุดท้ายคือ 34 จากนั้นตัวแจงส่วนจะนำโทเค็นเหล่านี้ไปประมวลผลตามกฎของไวยากรณ์ต่อไป โดยมีลักษณะการทำงานดังแสดงในรูปที่ 2.2 เป็นการแสดงให้เห็นลักษณะของการทำงานระหว่างตัววิเคราะห์คำและตัวแจงส่วน



รูปที่ 2.2 การทำงานระหว่างตัววิเคราะห์คำและตัวแจงส่วน

2.1.1 เครื่องมือสร้างตัวแจงส่วน PEG.js

การแจกแจงรูปประโยคของไวยากรณ์ (Parsing expression grammars) หรือ PEG มีลักษณะที่คล้ายคลึงกับไวยากรณ์ไม่พึ่งบริบท (Context-free grammars) หรือ CFG ที่มีการปรับปรุงแก้ไขบางส่วนเพื่อลดความคลุมเครือของไวยากรณ์ สำหรับเครื่องมือสร้างตัวแจงส่วนที่น่าสนใจและง่ายในการแจงส่วนของภาษาเขียนโปรแกรม คือ PEG.js (Hayat, 2012)

PEG.js เป็นเครื่องมือในการสร้างตัวแจงส่วนอย่างง่ายสำหรับภาษาจาวาสคริปต์ ที่สามารถสร้างตัวแจงส่วนได้อย่างรวดเร็ว รวมถึงมีการรายงานข้อผิดพลาดที่อาจเกิดขึ้นได้ PEG.js มีความสามารถในการนำไปประมวลผลข้อมูลที่มีความซับซ้อน อินเทอร์พรีเตอร์ (Interpreters) คอมไพเลอร์ (Compiler) และการสร้างเครื่องมืออื่น ๆ ได้ง่าย (Majda, 2013a) ผู้พัฒนา PEG.js คือ David Majda ได้มีการนำเสนอคุณสมบัติของ PEG.js ไว้ดังนี้

1. มีรูปแบบของการจัดการกับไวยากรณ์ที่ง่าย
2. มีการทำงานร่วมกันของทั้ง การวิเคราะห์คำศัพท์ (Lexical analysis) และการวิเคราะห์เชิงวากยสัมพันธ์ (Syntactic analysis)
3. ตัวแจงส่วนมีการรายงานข้อผิดพลาดที่อาจเกิดขึ้นได้อย่างดีเยี่ยม
4. อยู่บนพื้นฐานของการแจกแจงรูปประโยคของไวยากรณ์รูปแบบนิยม (Formalism) ซึ่งมีประสิทธิภาพมากกว่าตัวแจงส่วนแบบ LL(k) และ LR(k)
5. สามารถเลือกเข้าใช้งานได้จาก 3 ช่องทาง คือ เว็บเบราว์เซอร์ บรรทัดคำสั่ง (Command line) หรือผ่านทางส่วนต่อประสานโปรแกรมประยุกต์ (Application programming interface) หรือ API ของจาวาสคริปต์

2.1.2 วิธีการใช้งานเครื่องมือสร้างตัวแจงส่วน PEG.js เบื้องต้น

เครื่องมือสร้างตัวแจงส่วน PEG.js สามารถเรียกใช้งานได้ง่ายโดนผ่านทางบรรทัดคำสั่งในระบบปฏิบัติการวินโดวส์ (Windows) ซึ่งวิธีการใช้งานเครื่องมือสร้างตัวแจงส่วน PEG.js เบื้องต้นมีดังต่อไปนี้ (Majda, 2013b)

1. การติดตั้ง

วิธีการติดตั้งเครื่องมือสร้างตัวแจงส่วน PEG.js นั้น สามารถติดตั้งผ่าน Node.js (Joyent Inc., 2014) ซึ่งเป็นแพลตฟอร์มที่สร้างขึ้นบนจาวาสคริปต์รันไทม์ของกูเกิลโครม โดย Node.js ใช้สำหรับการรันไฟล์จาวาสคริปต์เพื่อให้ทำงานได้อย่างรวดเร็ว และยังมีประสิทธิภาพในการทำงานกับข้อมูลจำนวนมากที่ดำเนินงานบนอุปกรณ์ที่อยู่กระจายกันในช่วงระยะเวลาขณะนั้น

โดยการติดตั้งเครื่องมือสร้างตัวแจงส่วน PEG.js สามารถติดตั้งบนบรรทัดคำสั่งได้โดยใช้คำสั่ง

$ npm install -g pegjs

2. การใช้งานจากบรรทัดคำสั่ง

เครื่องมือสร้างตัวแจงส่วน PEG.js เมื่อป้อนข้อมูลเข้าเป็นไวยากรณ์ของภาษาที่ต้องการ แล้วจะได้ผลลัพธ์ออกมาเป็นตัวแจงส่วนที่อยู่ในรูปแบบของไฟล์จาวาสคริปต์ ตัวอย่างเช่น เมื่อสร้างไวยากรณ์ของภาษาที่ต้องการแล้วเก็บไว้ในไฟล์ที่มีชื่อว่า grammar.pegjs และใช้งานเครื่องมือสร้างตัวแจงส่วน PEG.js แล้วผลลัพธ์ที่ได้ออกมาจะเป็นไฟล์ grammar.js ซึ่งชื่อไฟล์นำเข้าและไฟล์ที่เป็นผลลัพธ์จะมีชื่อเดียวกัน แต่แตกต่างกันที่รูปแบบของไฟล์ ยกตัวอย่างเช่นหากต้องการสร้างตัวแจงส่วนที่สามารถคำนวณค่าทางคณิตศาสตร์พื้นฐาน โดยในการคำนวณค่าทางคณิตศาสตร์จะอยู่ในไฟล์ที่ชื่อว่า calculator.pegjs สามารถทำได้โดยการใช้คำสั่ง

$ pegjs calculator.pegjs

หลังจากใช้คำสั่งข้างต้นแล้ว จะได้ตัวแจงส่วนบนพื้นที่การทำงานส่วนบุคคลซึ่งอยู่ในรูปแบบของไฟล์จาวาสคริปต์ ตัวแจงส่วน calculator.js นั้นสามารถแจกแจงและคำนวณผลบวก ผลลบ ผลคูณ และผลหารทางคณิตศาสตร์ได้ ซึ่งสามารถป้อนข้อมูลเข้าให้กับไฟล์จาวาสคริปต์ได้โดยการใช้คำสั่ง

$ echo "1024 / 16" > testcalc

$ node calculator.js testcalc

หลังจากใช้คำสั่งข้างต้นแล้ว จะได้ผลลัพธ์ของค่าทางคณิตศาสตร์คือ 64 ซึ่งจะถูกแสดงออกทางบรรทัดคำสั่ง

3. การใช้งานตัวแจงส่วนจากเว็บ

เครื่องมือสร้างตัวแจงส่วน PEG.js นั้น เมื่อป้อนไวยกรณ์ของภาษาที่ต้องการเข้าไปแล้ว ผลลัพธ์ที่ได้ออกมาจะเป็นตัวแจงส่วนที่สามารถตรวจสอบไวยากรณ์ตามที่ป้อนเข้าไปได้ และอยู่ในรูปแบบของไฟล์จาวาสคริปต์ จึงสามารถนำไปใช้งานบนเว็บเบราว์เซอร์ได้ทันที การใช้งานสามารถทำได้โดยการเรียกตัวแจงส่วนที่อยู่ในรูปแบบของไฟล์จาวาสคริปต์ ผ่านทางแท็กสคริปต์ (Script tag) ในภาษาเอชทีเอ็มแอล (HTML) (W3C HTML Working Group, 2014) ตัวอย่างการ

เรียกใช้ตัวแจงส่วน calculator.js ผ่านทางแท็กสคริปต์ โดยใช้คำสั่ง

```
<script src = "calculator.js"></script>
<script>
    calculator.parse("42 / 1");
</script>
```

ชื่อตัวแปร calculator ที่เรียกใช้สามารถเปลี่ยนได้โดยการใช้คำสั่งต่อไปนี้ในขั้นตอนการสร้างตัวแจงส่วนในรูปแบบไฟล์จาวาสคริปต์บนบรรทัดคำสั่ง

```
$ pegjs -e JP calculator.pegjs
```

หลังจากใช้คำสั่งข้างต้นบนบรรทัดคำสั่งแล้ว ตัวแปรในการเรียกใช้ตัวแจงส่วนจะถูกเปลี่ยนจาก calculator เป็น JP

```
start
       = additive

additive
       = left:multiplicative "+" right:additive { return left + right; }
       / multiplicative

multiplicative
       = left:primary "*" right:nultiplicative { return left * right; }
       / primary

primary
       = integer
       / "(" additive:additive ")" { return additive; }

integer "integer"
       = digits:[0-9]+ { return parseInt(digits.join(""), 10); }
```

รูปที่ 2.3 ตัวอย่างบางส่วนของไวยากรณ์ในไฟล์ calculator.pegjs (Majda, 2013b)

4. การกำหนดภาษา

กระบวนการในการแจกแจงรูปประโยคของภาษามีด้วยกันทั้งหมด 2 ขั้นตอนคือ ส่วนวิเคราะห์คำและส่วนการแจงรูปประโยค ซึ่งเครื่องมือสร้างตัวแจงส่วน PEG.js นำขั้นตอนทั้งหมดในการแจงรูปประโยคของภาษารวมไว้ในไฟล์เดียวกัน เพื่อช่วยให้ง่ายและสะดวกในการทำงาน โดยการสร้างตัวแจงส่วนด้วยเครื่องมือสร้างตัวแจงส่วน PEG.js จะมีบางส่วนที่คล้ายคลึงกับภาษา

จาวาสคริปต์เพื่อให้ผู้ใช้งานเข้าใจง่าย ดังแสดงในรูปที่ 2.3 ตัวอย่างบางส่วนของไวยากรณ์ที่ใช้สร้างตัวแจงส่วนเพื่อการคำนวณทางคณิตศาสตร์

จากตัวอย่างข้างต้นจะเห็นว่ามีการรวมขั้นตอนการวิเคราะห์คำและการแจกแจงรูปประโยคไว้ด้วยกัน และยังมีการใช้คำสั่งภาษาจาวาสคริปต์อยู่ในเครื่องหมาย { } ซึ่งการที่เครื่องมือสร้างตัวแจงส่วน PEG.js มีรูปแบบการทำงานดังที่กล่าวมา ทำให้ผู้ใช้ง่ายและสะดวกในการจัดการกับการสร้างตัวแจงส่วน

5. ขั้นตอนวิธีในการแจงรูปประโยค

เครื่องมือสร้างตัวแจงส่วน PEG.js ใช้ขั้นตอนวิธีการแจงส่วนแบบ Packrat (Ford, 2014b) การแจงส่วนแบบ Packrat คือวิธีการดำเนินการกับตัวแจงส่วนแบบเวลาเชิงเส้น (Linear-time parsers) สำหรับไวยากรณ์ที่กำหนดในการแจงส่วนภาษาแบบบนลงล่าง (Top-down parsing language) หรือ TDPL โดย TDPL ถูกสร้างขึ้นเป็นรูปแบบพื้นฐานสำหรับตัวแจงส่วนแบบบนลงล่างที่มีความสามารถในการติดตามแบบย้อนกลับ (Backtracking) ซึ่งมีประสิทธิภาพในการอธิบายไวยากรณ์ของภาษามากกว่า CFG โดยอาจมีบางสิ่งที่ CFG ไม่สามารถอธิบายได้ แต่ TDPL สามารถทำได้อย่างง่ายดาย เช่น คำกำกวม เป็นต้น ทำให้สามารถอธิบายไวยากรณ์ที่สมบูรณ์ของภาษาได้ใน TDPL เดียว (Ford, 2014a)

การแจงส่วนแบบ Packrat คือการปรับปรุงขั้นตอนวิธีการแจงส่วน 30 ปีก่อนที่ไม่เคยนำมาใช้จนกระทั่งปัจจุบันนี้ โดยการแจงส่วนแบบ Packrat สามารถรับรู้คำที่ถูกนิยามด้วยไวยากรณ์ TDPL ในเวลาเชิงเส้น ทำให้มีความยืดหยุ่นในการติดตามย้อนกลับแบบวนซ้ำ โดยปราศจากความเสี่ยงในเรื่องของเวลา ซึ่งการแจงส่วนแบบ Packrat สามารถรับรู้ได้ทั้งภาษาแบบ LL(k) หรือ LR(k) และยังสามารถรับรู้ภาษาที่ต้องการการดูล่วงหน้า (Look-ahead) แบบไม่จำกัด ที่ไม่สามารถแจกแจงด้วยการแจกแจงแบบการเปลี่ยน/ลด (Shift/Reduce) ได้ นอกจากนี้การแจงส่วนแบบ Packrat ยังมีคุณสมบัติที่ดีกว่าการแจกแจงแบบ LL/LR ทำให้เหมาะกับภาษาแบบพลวัต (Dynamic language) หรือแบบขยาย (Extensible) แต่การแจงส่วนแบบ packrat ยังมีข้อเสีย ในด้านของการเก็บข้อมูลซึ่งมาจากขั้นตอนการสร้างโครงสร้างของประโยค (Ford, 2014a)

2.2 ความขัดแย้งเชิงความหมายจากการพัฒนาซอฟต์แวร์ร่วมกัน

ตามแนวคิดพื้นฐานในการพัฒนาซอฟต์แวร์ร่วมกัน (Fan and Sun, 2012a) รหัสต้นฉบับที่ถูกใช้ร่วมกันนั้นจำเป็นจะต้องเป็นไปตามกฎวากยสัมพันธ์ (Syntax) ของภาษาที่ใช้ในการพัฒนาและการแก้ไขปัญหาเชิงตรรกะ ซึ่งหากไม่เป็นไปตามที่กล่าวไว้อาจทำให้เกิดความขัดแย้งเชิงความหมายขึ้น โดยเงื่อนไขหลัก ๆ ที่อาจจะทำให้เกิดความขัดแย้งเชิงความหมายนั้นมีดังนี้

1. นักพัฒนาหลายคนทำงานพร้อมกัน
2. นักพัฒนาที่ทำงานพร้อมกัน เมื่อเจอปัญหาในรหัสต้นฉบับจะแก้ไขรหัสต้นฉบับในแบบของตนเอง หรือแก้ไขรหัสต้นฉบับในส่วนที่พึ่งพากับส่วนอื่นภายในรหัสต้นฉบับอยู่
3. นักพัฒนาแก้ไขรหัสต้นฉบับแล้วเกิดความหมายที่ขัดแย้งกัน

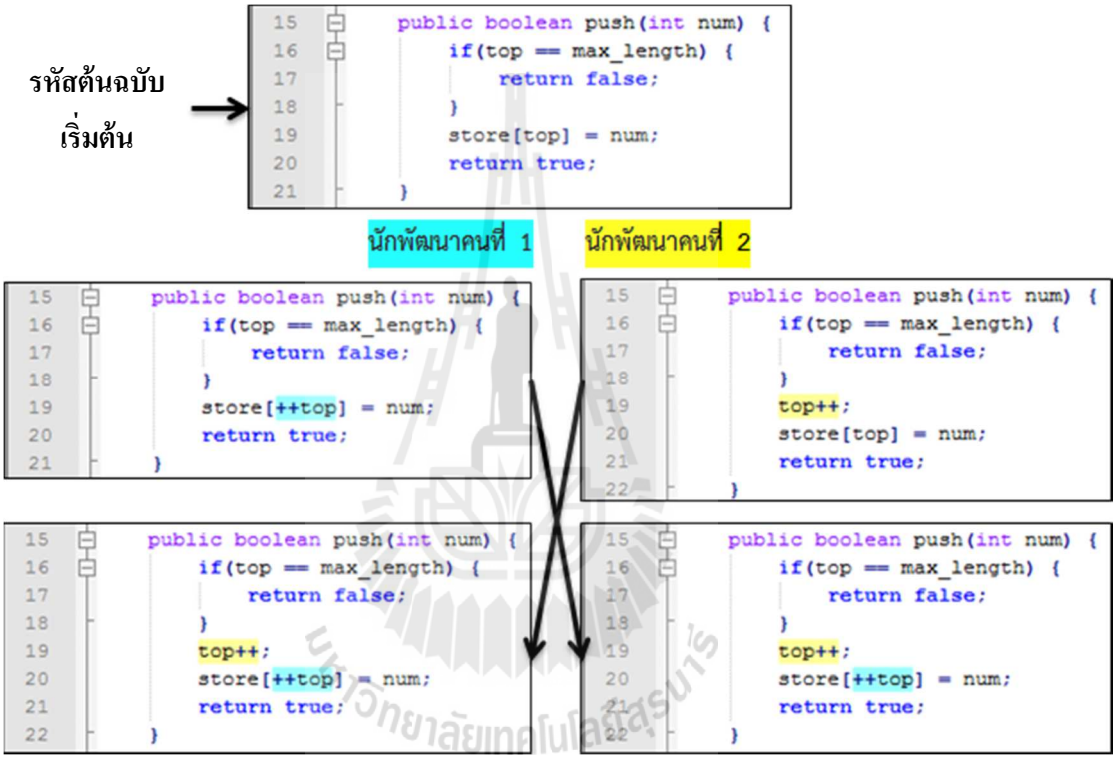
เงื่อนไขที่ได้กล่าวไปข้างต้นแล้วนั้น ครอบคลุมกรณีที่ทำให้เกิดความขัดแย้งเชิงความหมายทั้งหมด ยกตัวอย่างการทำงานของสแตก (Stack) ในภาษาจาวาเพื่อใช้เป็นตัวแทนในการอธิบายความขัดแย้งเชิงความหมายที่อาจเกิดขึ้นในการพัฒนาซอฟต์แวร์ร่วมกัน โดยสแตกจะเป็นหน่วยความจำในการเก็บตัวเลขจำนวนเต็มและใช้เมธอด (Method) อย่างง่ายในการทำงาน ซึ่งประกอบด้วยการดัน (Push) ข้อมูลลงในสแตก การดึง (Pop) ข้อมูลออกจากสแตก การค้น (Retrieve) ข้อมูลตัวบนสุดของสแตก และการตรวจสอบ (Check) ว่าสแตกว่างอยู่หรือไม่ โดยมีรหัสต้นฉบับการทำงานของสแตกดังแสดงในรูปที่ 2.4

```java
// Stack.java
package stack;

public class Stack {
    protected int store[];
    protected int max_length;
    protected int top;

    public Stack(int size) {
        store = new int[size];
        max_length = size - 1;
        top = -1;
    }

    public boolean push(int num) {
        if(top == max_length) {
            return false;
        }
        store[top] = num;
        return true;
    }

    public boolean pop() {
        if(top == -1) {
            return false;
        }
        top--;
        return true;
    }

    public int top() {
        return store[/* TODO */];
    }

    public boolean isEmpty() {
        return (top == -1);
    }
}
```

รูปที่ 2.4 รหัสต้นฉบับการทำงานของสแตกในภาษาจาวา

ในแต่ละเมธอด (Method) และตัวแปรมีหน้าที่การทำงานดังต่อไปนี้ เมธอด Stack(int size) จะรับค่าพารามิเตอร์มาเพื่อกำหนดขนาดสูงสุดของสแตก เมธอด push(int num) จะดันข้อมูลลงไปในสแตก เมธอด pop() จะดึงข้อมูลตัวบนสุดออกจากสแตก เมธอด top() จะเรียกใช้ข้อมูลตัวบนสุดของสแตกโดยที่ข้อมูลนั้นยังอยู่ในสแตก และเมธอด isEmpty() จะตรวจสอบว่าสแตกนั้นว่าง

หรือไม่ โดยตัวแปร store เป็นตัวแปรที่เก็บข้อมูลของสแตก ตัวแปร max_length เป็นตัวแปรที่บอกจำนวนข้อมูลสูงสุดที่สแตกสามารถเก็บได้ และตัวแปร top เป็นตัวแปรที่เก็บจำนวนข้อมูลภายในสแตก ยกตัวอย่างให้มีนักพัฒนากำลังทำงานกับรหัสต้นฉบับข้างต้นอยู่สองคน และทำให้เกิดความขัดแย้งเชิงความหมาย โดยจะแสดงให้เห็นแยกเป็นแต่ละกรณีที่ทำให้เกิดความขัดแย้ง 3 กรณีดังต่อไปนี้

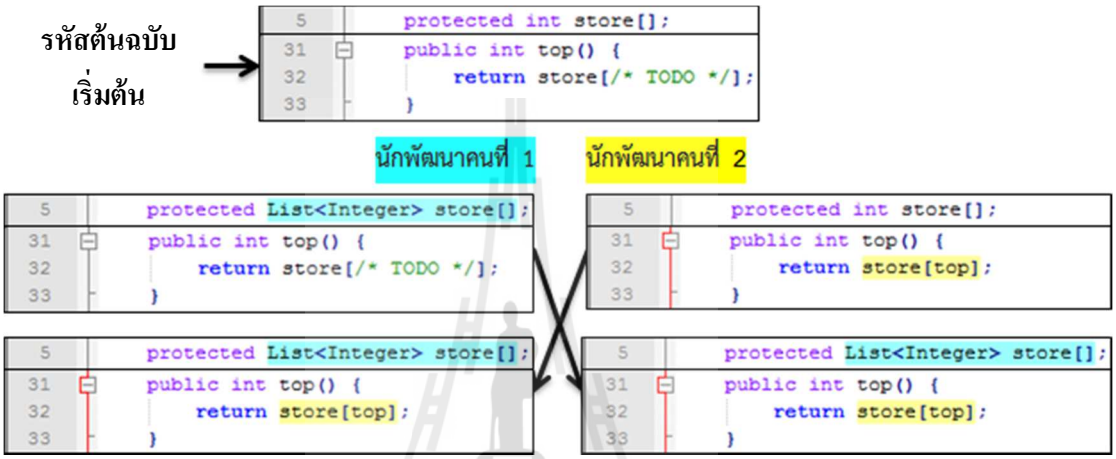


รูปที่ 2.5 ความขัดแย้งเชิงความที่เกิดจากการแก้ไขรหัสต้นฉบับพร้อมกันในส่วนที่ไม่พึ่งพากันกับรหัสต้นฉบับส่วนอื่น

2.2.1 การแก้ไขรหัสต้นฉบับพร้อมกันในส่วนที่ไม่พึ่งพากันกับรหัสต้นฉบับส่วนอื่น

ความขัดแย้งเชิงความหมายที่เกิดจากการแก้ไขในกรณีนี้เกิดขึ้นในเมธอด push(int num) ดังแสดงในรูปที่ 2.5 ในส่วนของรหัสต้นฉบับเริ่มต้นที่มีข้อผิดพลาดอยู่คือ เมื่อมีการดันข้อมูลลงในสแตกแล้วค่าของตัวแปร top ควรเพิ่มขึ้นทีละหนึ่ง ซึ่งนักพัฒนาทั้งสองคนเห็นถึงข้อผิดพลาดดังกล่าวและพยายามที่จะแก้ปัญหาตามแนวคิดของตัวเอง จากรูปที่ 2.5 ในส่วนของนักพัฒนาที่ 1 ทำการเพิ่มค่าให้กับตัวแปร top ด้วยคำสั่ง ++top ในบรรทัดที่ 19 และในส่วนของนักพัฒนาที่ 2 ทำ

การเพิ่มค่าให้กับตัวแปร top ด้วยคำสั่ง top++ ในบรรทัดที่ 19 โดยการแก้ไขปัญหาการเพิ่มค่าของนักพัฒนาทั้งสองคนนั้นถูกต้อง แต่เมื่อปรับปรุงรหัสต้นฉบับที่ทำงานร่วมกันแล้ว แทนที่ค่าของตัวแปร top จะถูกเพิ่มแค่หนึ่งครั้ง กลับถูกเพิ่มค่าของตัวแปรถึงสองครั้ง ซึ่งเกิดจากการแก้ไขรหัสต้นฉบับในส่วนเดียวกัน แต่วิธีการแก้ไขไม่ตรงกัน ทำให้เกิดความขัดแย้งเชิงความหมายขึ้น



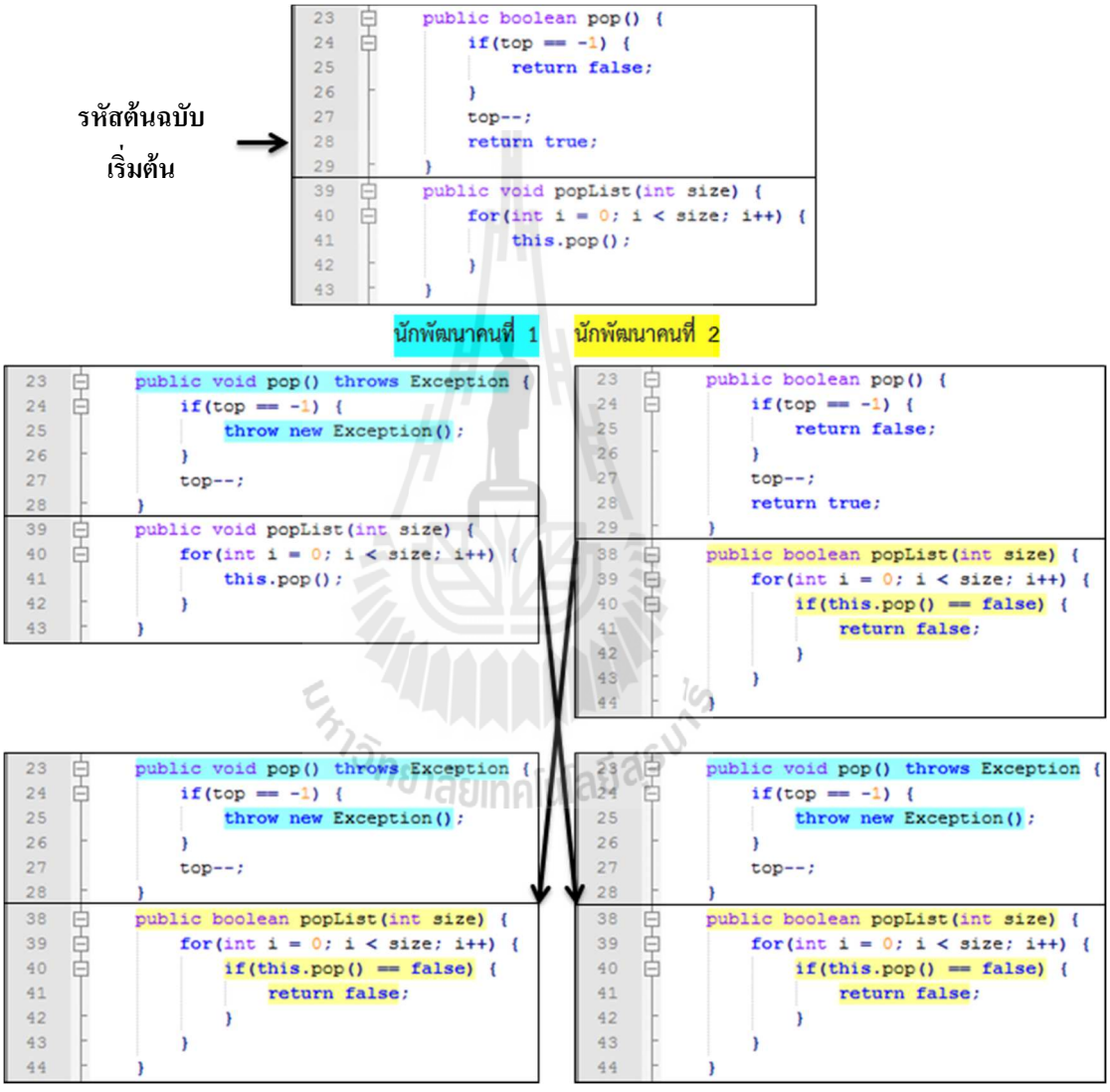
รูปที่ 2.6 ความขัดแย้งเชิงความหมายที่เกิดจากการแก้ไขรหัสต้นฉบับพร้อมกันในตัวแปรที่พึ่งพากัน

2.2.2 การแก้ไขรหัสต้นฉบับพร้อมกันในตัวแปรที่พึ่งพากัน

ความขัดแย้งเชิงความหมายที่เกิดจากการแก้ไขในกรณีนี้จะต่างจากกรณีแรก คือ นักพัฒนาที่ทำงานร่วมกันแก้ไขส่วนของรหัสต้นฉบับที่แตกต่างกันแต่เป็นส่วนที่พึ่งพากันอยู่ โดยที่ส่วนที่เกิดข้อผิดพลาดนั้นคือส่วนของเมธอด top() ดังแสดงในรูปที่ 2.6 ในส่วนของรหัสต้นฉบับเริ่มต้น นักพัฒนาทั้งสองเห็นถึงความผิดพลาดที่แตกต่างกัน โดยนักพัฒนาที่ 1 มุ่งเน้นไปที่การแก้ไขความหมายของตัวแปร store ส่วนนักพัฒนาที่ 2 มุ่งเน้นไปที่การทำงานที่ไม่สมบูรณ์ของเมธอด top() เนื่องจากเป็นแก้ไขรหัสต้นฉบับคนละส่วนกันนักพัฒนาทั้งสองคนจึงอาจไม่สนใจการกระทำของนักพัฒนาอีกคนหนึ่ง จากรูปที่ 2.6 นักพัฒนาที่ 1 ทำการแก้ไขโดยการเปลี่ยนแปลงโครงสร้างของข้อมูลสำหรับการเก็บข้อมูลของตัวแปร stack ในบรรทัดที่ 5 ส่วนนักพัฒนาที่ 2 ทำการแก้ไขที่เมธอด top() ในบรรทัดที่ 32 ซึ่งเป็นส่วนของการส่งค่ากลับเมื่อมีการเรียกใช้เมธอด ให้สามารถทำงานได้อย่างสมบูรณ์

ความขัดแย้งเชิงความหมายที่เกิดจากการแก้ไขรหัสต้นฉบับพร้อมกันในตัวแปรที่

พึ่งพากันนี้ มีลักษณะเช่นเดียวกันกับกรณีแรก จะเห็นได้ว่าการแก้ไขของนักพัฒนาทั้งสองถูกต้องในพื้นที่การทำงานของตัวเอง แต่เมื่อนำรหัสต้นฉบับมารวมกันแล้วกลับเกิดข้อผิดพลาดขึ้นทันที เนื่องจากชนิดของข้อมูลไม่ตรงกัน ซึ่งเกิดจากการแก้ไขส่วนของรหัสต้นฉบับที่พึ่งพากันอยู่ จึงทำให้เกิดความขัดแย้งเชิงความหมายขึ้น



รูปที่ 2.7 ความขัดแย้งเชิงความหมายที่เกิดจากการแก้ไขรหัสต้นฉบับพร้อมกันในเมธอดที่พึ่งพากัน

2.2.3 การแก้ไขรหัสต้นฉบับพร้อมกันในเมธอดที่พึ่งพากัน

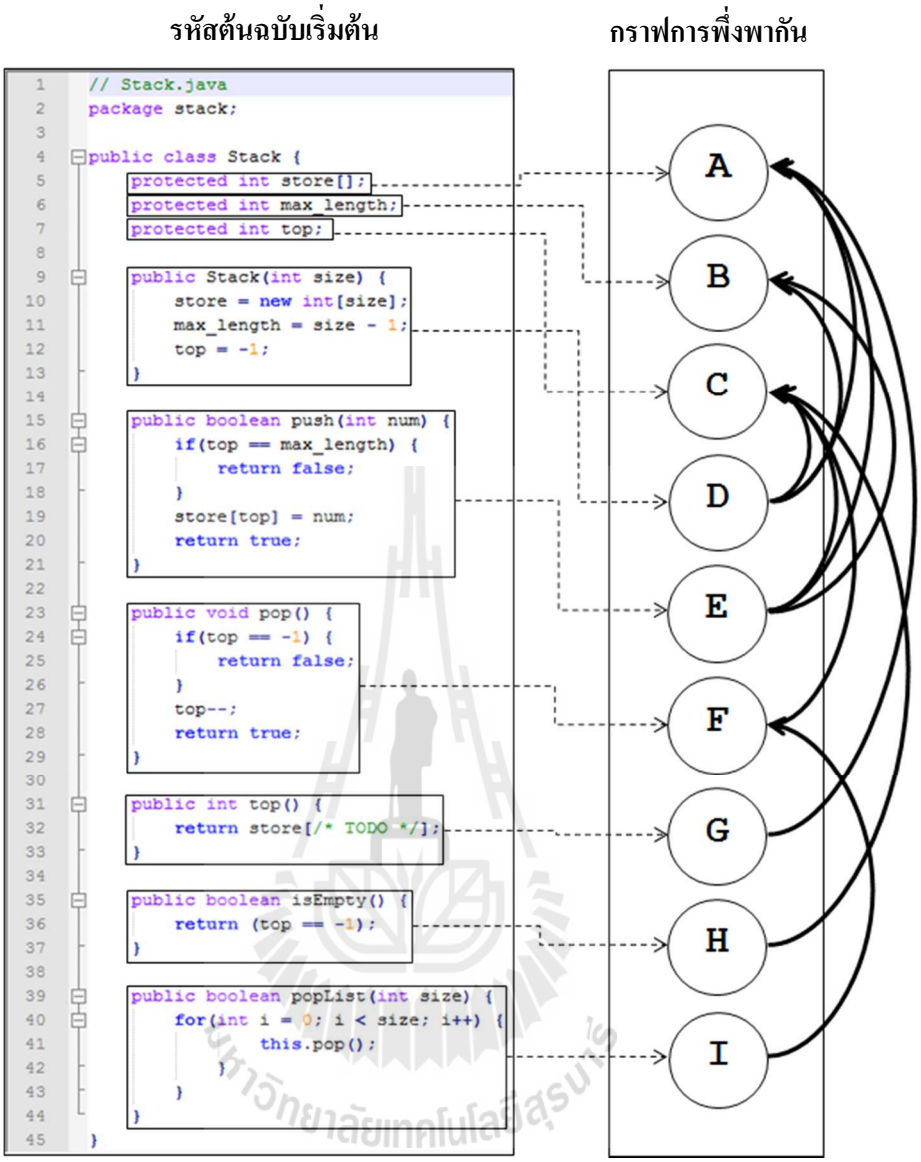
ความขัดแย้งเชิงความหมายสำหรับกรณีที่สามได้เพิ่มเมธอดขึ้นมาอีกหนึ่งเมธอดคือ เมธอด popList(int size) ดังแสดงในรูปที่ 2.7 ในส่วนของรหัสต้นฉบับเริ่มต้น โดยเมธอด popList(int size) มีหน้าที่ในการลบข้อมูลทั้งหมดในสแตก โดยใช้วิธีการเรียกเมธอด pop() ซึ่งมีข้อผิดพลาดที่เกิดขึ้นคือเมื่อการทำงานของเมธอดเกิดข้อผิดพลาด เมธอดจะไม่มีการแจ้งเตือนเกี่ยวกับข้อผิดพลาดที่เกิดขึ้นเลย ซึ่งนักพัฒนาทั้งสองคนก็เห็นถึงข้อผิดพลาดนี้เช่นเดียวกัน จากรูปที่ 2.7 นักพัฒนาที่ 1 ทำการแก้ไขโดยการใช้ข้อยกเว้นของจาวา (Java Exception) ที่เมธอด popList(int size) ในบรรทัดที่ 25 เพื่อดักจับข้อผิดพลาด ส่วนนักพัฒนาที่ 2 ทำการแก้ไขโดยเปลี่ยนการส่งกลับผลลัพธ์ของเมธอด popList(int size) จาก void เป็น boolean พร้อมทั้งตรวจสอบค่าผลลัพธ์จากเมธอด pop() หากผลลัพธ์เป็น false เมธอด popList(int size) จะส่งค่าผลลัพธ์กลับเป็น false เช่นกัน

ความขัดแย้งเชิงความหมายที่เกิดจากการแก้ไขรหัสต้นฉบับพร้อมกันในเมธอดที่พึ่งพากันมีลักษณะเช่นเดียวกันกับสองกรณีก่อนหน้า คือการแก้ไขของนักพัฒนาทั้งสองคนนั้นถูกต้องบนพื้นที่การทำงานของตัวเอง แต่เมื่อนำมารวมกันแล้วจะเกิดข้อผิดพลาดขึ้นทันที ซึ่งเกิดจากการที่นักพัฒนาแก้ไขในเมธอดที่พึ่งพากัน ด้วยวิธีที่แตกต่างกันทำให้เกิดความขัดแย้งเชิงความหมายขึ้น

2.3 การจัดการความขัดแย้งเชิงความหมาย

วิธีการในการจัดการความขัดแย้งเชิงความหมายโดยทั่วไปที่นิยมกันมีอยู่ 2 วิธี คือ การแก้ไขความขัดแย้ง และการป้องกันความขัดแย้ง โดยการแก้ไขความขัดแย้งนั้นจะเป็นการตรวจจับความขัดแย้งที่เกิดขึ้นแล้วทำการแก้ไข ซึ่งวิธีนี้จะเหมาะสำหรับความขัดแย้งที่ง่ายต่อการตรวจจับและแก้ไขได้ง่าย ส่วนการป้องกันความขัดแย้งนั้นเป็นการป้องกันไม่ให้เกิดความขัดแย้งขึ้นโดยวิธีนี้เหมาะสำหรับความขัดแย้งที่ตรวจจับและแก้ไขได้ยาก

ในงานวิจัยนี้มุ่งเน้นไปที่การป้องกันความขัดแย้งเชิงความหมาย ด้วยวิธีการล็อคพื้นที่การทำงานซึ่งเป็นหนึ่งในวิธีที่มีประสิทธิภาพ การล็อคพื้นที่การทำงานนั้นจะเป็นการปิดกั้นพื้นที่ส่วนหนึ่งของรหัสต้นฉบับที่พัฒนาร่วมกันเมื่อมีนักพัฒนาคนหนึ่งกำลังแก้ไขส่วนนั้นอยู่ โดยหากนักพัฒนาคนอื่นต้องการแก้ไขส่วนเดียวกัน ต้องรอให้นักพัฒนาที่ทำงานอยู่ก่อนเปลี่ยนพื้นที่การทำงาน จึงจะเข้าไปทำงานส่วนที่ต้องการนั้นได้ การล็อคพื้นที่การทำงานนั้นจะรวมถึงการล็อคส่วนที่พึ่งพากันกับส่วนที่ล็อคอยู่ด้วย โดยการใช้กราฟการพึ่งพากัน (Dependency graph) ในการระบุว่าส่วนใดของรหัสต้นฉบับที่มีความเกี่ยวข้องกันบ้าง ยกตัวอย่างกราฟการพึ่งพากัน ซึ่งเป็นกราฟการ



รูปที่ 2.8 กราฟการพึ่งพากันของรหัสต้นฉบับการทำงานของสแตกในภาษาจาวา

พึ่งพากันของการทำงานของสแตกในภาษาจาวาดังแสดงในรูปที่ 2.8 เป็นตัวอย่างกราฟการพึ่งพากันของรหัสต้นฉบับการทำงานของสแตก ในภาษาจาวา จะเห็นว่าแต่ละส่วนของรหัสต้นฉบับถูกแบ่งออกเป็นส่วน ๆ โดยแบ่งตามเมธอด เขตข้อมูล (Field) หรือตัวแปรในรหัสต้นฉบับและระบุส่วนที่แบ่งออกมาด้วยตัวอักษรภาษาอังกฤษ สมมุติให้นักพัฒนาคนหนึ่งกำลังทำงานอยู่ในส่วน H กราฟการพึ่งพากัน จะเกิดการล็อคพื้นที่การทำงาน H ขึ้น และล็อคในส่วนที่มีการพึ่งพากันคือส่วน C เมื่อนักพัฒนาคนดังกล่าวออกจากพื้นที่การทำงานในส่วน H แล้วจะทำให้ส่วน H และส่วน C ถูกปลดล็อคพร้อมกัน

นอกจากนี้การล็อคพื้นที่การทำงานยังแบ่งได้อีก 2 วิธี คือ การล็อคพื้นที่การทำงานด้วยตัวนักพัฒนาเอง และการล็อคพื้นที่การทำงานแบบอัตโนมัติ ซึ่งการล็อคพื้นที่การทำงานด้วยนักพัฒนาเองนั้น นักพัฒนาจะเป็นคนตัดสินใจเองว่าจะล็อคและปล่อยพื้นที่การทำงานเมื่อใด และกำหนดขอบเขตของพื้นที่การทำงานที่จะล็อคด้วยตัวเอง การกระทำในลักษณะนี้จะทำให้นักพัฒนามีภาระเพิ่มขึ้น และยังอาจเกิดความผิดพลาดเพิ่มเติมจากตัวนักพัฒนาเอง ส่วนอีกวิธีหนึ่งคือการล็อคพื้นที่การทำงานแบบอัตโนมัติ โดยระบบจะเป็นตัวกำหนดขอบเขตการล็อคและตัดสินใจแทนนักพัฒนาเองว่าควรล็อคและปล่อยพื้นที่การทำงานเมื่อใด ทำให้นักพัฒนาไม่ต้องกังวลในการล็อคพื้นที่การทำงานและพัฒนาซอฟต์แวร์ได้มีประสิทธิภาพขึ้น

2.4 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ร่วมกันและการแก้ปัญหาที่เกิดจากการพัฒนาซอฟต์แวร์ร่วมกันมีอยู่เป็นจำนวนมาก ในการพัฒนาซอฟต์แวร์ภาษาจาวาร่วมกันบนเว็บเบราว์เซอร์โดยใช้การล็อคพื้นที่การทำงานเพื่อป้องกันความขัดแย้งเชิงความหมาย ผู้วิจัยได้ทำการศึกษาค้นคว้างานวิจัยที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ร่วมกันและการป้องกันความขัดแย้งเชิงความหมายที่น่าสนใจ โดยสรุปไว้ในตารางที่ 2.1 และมีรายละเอียดดังต่อไปนี้

1. งานวิจัยของ Jon A Preston, Xiaolin Hu และ Sushil K Prasad (2007) เรื่อง “Simulation-based Architectural Design and Implementation of Real-Time Collaborative Editing System” ได้นำเสนอการพัฒนาซอฟต์แวร์ร่วมกันโดยใช้เทคนิคที่เรียกว่าการแปลงการดำเนินการ (operation transformation) เพื่อให้รหัสต้นฉบับที่กระจายอยู่บนพื้นที่การทำงานของนักพัฒนาแต่ละคนมีความสอดคล้องกัน รวมถึงยังใช้เทคนิคการล็อคพื้นที่การทำงานเพื่อกำหนดสิทธิ์การเข้าถึงรหัสต้นฉบับของนักพัฒนาแต่ละคน ซึ่งนักพัฒนาแต่ละคนนั้นจะทำงานบนซอฟต์แวร์การจัดการข้อมูล (Editor) ที่เชื่อมต่อกับเซิร์ฟเวอร์ผ่านบริการเว็บ (Web-service) โดยการทดลองจะมุ่งเน้นไปที่การเปรียบเทียบค่าใช้จ่ายด้านการสื่อสารซึ่งมีมากเมื่อมีการพัฒนาซอฟต์แวร์ร่วมกัน และมีการทดสอบซอฟต์แวร์ด้วยการจำลองการทำงานแบบวนซ้ำ

2. งานวิจัยของ Max Goldman (2011) เรื่อง “Role-Based Interfaces for Collaborative Software Development” ได้นำเสนอระบบ Collabode ซึ่งเป็นสภาพแวดล้อมการพัฒนา (IDE) ที่ใช้ในการพัฒนาซอฟต์แวร์ร่วมกัน ระบบ Collabode ทำงานบนเว็บเบราว์เซอร์โดยมีซอฟต์แวร์จัดการข้อมูลเป็นโปรแกรมประยุกต์บนเว็บ (Web application) ส่วนติดต่อกับผู้ใช้ถูกพัฒนาด้วยภาษาเอชทีเอ็มแอล และภาษาจาวาสคริปต์ ส่วนของไคลเอนต์จะเข้ามาทำงานในรหัสต้นฉบับเดียวกันได้

ผ่านทาง URL เดียวกัน ส่วนในด้านของเซิร์ฟเวอร์จะใช้อีคลิปส์เพื่อจัดการคอมไพล์ (Complie) รหัสต้นฉบับที่ใช้งานร่วมกันและแจ้งเตือนข้อผิดพลาดที่เกิดจากการคอมไพล์

3. งานวิจัยของ Hongfei Fan และ Chengzheng Sun (2012a) เรื่อง “Dependency-based Automatic Locking for Semantic Conflict Prevention in Real-Time Collaborative Programming” ได้นำเสนอปัญหาและวิธีแก้ไขความขัดแย้งเชิงความหมายที่เกิดขึ้นจากการพัฒนาซอฟต์แวร์ร่วมกัน โดยได้นำเสนอการป้องกันความขัดแย้งเชิงความหมายด้วยเทคนิคที่เรียกว่า การล็อคส่วนที่พึ่งพากันแบบอัตโนมัติ (Dependency-based Automatic Locking) เป็นเทคนิคการป้องกันความขัดแย้งเชิงความหมายด้วยการล็อคพื้นที่การทำงานแบบอัตโนมัติ เมื่อมีนักพัฒนากำลังแก้ไขรหัสต้นฉบับอยู่ การล็อคนี้รวมไปถึงล็อคส่วนที่พึ่งพากับส่วนที่นักพัฒนากำลังทำงานอยู่ด้วย ซึ่งเมื่อนักพัฒนาทำงานเสร็จแล้วและออกจากพื้นที่ทำงานนั้น จะปล่อยพื้นที่การทำงานส่วนดังกล่าวโดยอัตโนมัติเพื่อให้นักพัฒนาคนอื่นที่ต้องการทำงานในพื้นที่เดียวกันได้ใช้งาน การล็อคพื้นที่การทำงานกับส่วนที่พึ่งพากันนั้นใช้กราฟการพึ่งพากัน ซึ่งจะเป็นกราฟที่ระบุว่าส่วนใดของรหัสต้นฉบับที่มีความเกี่ยวข้องกันบ้าง เพื่อให้การล็อคพื้นที่การทำงานและส่วนที่พึ่งพากันนั้นเป็นไปได้อย่างถูกต้อง

4. งานวิจัยของ Hongfei Fan และ Chengzheng Sun (2012b) เรื่อง “Supporting Semantic Conflict Prevention in Real-Time Collaborative Programming Environments” ได้นำเสนอวิธีการแก้ไขปัญหาความขัดแย้งเชิงความหมายที่เกิดขึ้นจากการพัฒนาซอฟต์แวร์ร่วมกัน โดยได้นำเทคนิคการล็อคส่วนที่พึ่งพากันแบบอัตโนมัติ ซึ่งเป็นเทคนิคที่ป้องกันความขัดแย้งเชิงความหมายไม่ให้เกิดขึ้น โดยในงานวิจัยนี้เป็นการอธิบายถึงเทคนิคและวิธีการเกี่ยวกับเทคนิคการล็อคส่วนที่พึ่งพากันแบบอัตโนมัติ รวมถึงกราฟการพึ่งพากันที่จะใช้ในการล็อคส่วนที่พึ่งพากันอย่างละเอียด และได้แสดงถึงตัวอย่างของอัลกอริทึมที่ใช้จริงรวมถึงระบบต้นแบบที่ชื่อ CoEclipse ซึ่งสนับสนุนการพัฒนาซอฟต์แวร์ร่วมกันโดยนำเทคนิคดังกล่าวไปใช้ ระบบต้นแบบจะอยู่บนซอฟต์แวร์การจัดการข้อมูลจาวาอีคลิปส์ (Eclipse Java Editor) ส่วนของไคลเอนต์ (Client) จะทำงานเป็นอีคลิปส์ไอดีอี (Eclipse IDE) ซึ่งจะกระจายตัวอยู่ตามพื้นที่การทำงานที่เชื่อมต่อกับเซิร์ฟเวอร์กลางของระบบ CoEclipse โดยแต่ละพื้นที่การทำงานนักพัฒนาจะเห็นรหัสต้นฉบับเดียวกันและสามารถแก้ไขรหัสต้นฉบับเหล่านั้นได้ หากรหัสต้นฉบับส่วนนั้นไม่ถูกล็อคอยู่

จากการศึกษาปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้องได้แสดงให้เห็นว่า งานวิจัยในส่วนของการพัฒนาซอฟต์แวร์ร่วมกันนั้น จะมีปัญหาที่เหมือนกันคือความขัดแย้งเชิงความหมายและ

งานวิจัยเหล่านี้ได้แสดงการป้องกันและแก้ปัญหาความขัดแย้งเชิงความหมายมากมาย ซึ่งมีทั้งข้อดีและข้อเสียที่แตกต่างกันออกไป อีกทั้งงานวิจัยเหล่านี้ยังได้เสนอระบบต้นแบบในการพัฒนาซอฟต์แวร์ร่วมกัน โดยได้ใช้เทคนิคในการป้องกันหรือแก้ปัญหาความขัดแย้งเชิงความหมายในระบบต้นแบบด้วย ซึ่งในปัจจุบันปฏิเสธไม่ได้เลยว่าการพัฒนาซอฟต์แวร์ร่วมกันนั้นมีบทบาทเพิ่มมากขึ้นเรื่อย ๆ ด้วยข้อดีมากมายของการพัฒนาซอฟต์แวร์ร่วมกัน ผู้วิจัยจึงได้นำแนวทางและเทคนิคในการป้องกันและแก้ไขมาใช้ร่วมกันกับระบบต้นแบบที่มีคุณภาพ เพื่อเพิ่มความถูกต้องและลดปัญหาในการพัฒนาซอฟต์แวร์ร่วมกัน โดยทำการสรุปเทคนิคและรูปแบบการทำงานของระบบต้นแบบของงานวิจัยที่เกี่ยวข้องไว้ในตารางที่ 2.1 และแทนงานวิจัยแต่ละงานด้วยตัวอักษรภาษาไทยดังนี้

ก. แทนงานวิจัยของ Jon A Preston, Xiaolin Hu และ Sushil K Prasad เรื่อง “Simulation-based Architectural Design and Implementation of Real-Time Collaborative Editing System”

ข. แทนงานวิจัยของ Max Goldman เรื่อง “Role-Based Interfaces for Collaborative Software Development”

ค. แทนงานวิจัยของ Hongfei Fan และ Chengzheng Sun เรื่อง “Dependency-based Automatic Locking for Semantic Conflict Prevention in Real-Time Collaborative Programming”

ง. แทนงานวิจัยของ Hongfei Fan และ Chengzheng Sun เรื่อง “Supporting Semantic Conflict Prevention in Real-Time Collaborative Programming Environments”

จ. แทนงานวิจัยเรื่อง “การพัฒนาซอฟต์แวร์ภาษาจาวาร่วมกันบนเว็บเบราว์เซอร์โดยใช้การล็อคพื้นที่การทำงานเพื่อป้องกันความขัดแย้งเชิงความหมาย” (งานวิจัยของวิทยานิพนธ์ฉบับนี้)

ตารางที่ 2.1 สรุปเปรียบเทียบงานวิจัยที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ร่วมกัน

กระบวนการทำงาน	งานวิจัยที่เกี่ยวข้อง				
	ก	ข	ค	ง	จ
วัตถุประสงค์ของการวิจัย					
เพื่อป้องกันความขัดแย้งเชิงความหมาย			✓	✓	✓
เพื่อให้สามารถทำงานบนเอกสารเดียวกันได้	✓	✓		✓	✓
เพื่อลดค่าใช้จ่ายด้านการสื่อสารระหว่างผู้ที่ทำงานร่วมกัน	✓				
เพื่อเสนอระบบต้นแบบในการทำงานร่วมกัน	✓	✓		✓	✓
เพื่อสร้างตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์					✓

ตารางที่ 2.1 สรุปเปรียบเทียบงานวิจัยที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ร่วมกัน (ต่อ)

กระบวนการทำงาน	งานวิจัยที่เกี่ยวข้อง				
	ก	ข	ค	ง	จ
ภาษาที่ใช้ในการทำงานร่วมกัน					
จาวา		✓	✓	✓	✓
อังกฤษ	✓				
รูปแบบการทำงานของระบบต้นแบบ					
อีคลิปส์		✓	✓	✓	
เว็บเบราว์เซอร์		✓			✓
บริการเว็บ	✓				
การประเมินประสิทธิภาพของระบบต้นแบบ					
การจำลองการใช้งานแบบวนซ้ำ	✓				✓

บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้จะเป็นการนำเสนอวิธีการดำเนินการวิจัย โดยในหัวข้อที่ 3.1 เป็นการนำเสนอระเบียบวิธีในการดำเนินการวิจัย หัวข้อที่ 3.2 เป็นการนำเสนอการออกแบบระบบต้นแบบที่ใช้ในการพัฒนาซอฟต์แวร์ร่วมกันโดยใช้การล็อคพื้นที่การทำงานเพื่อช่วยในการป้องกันความขัดแย้งเชิงความหมาย หัวข้อที่ 3.3 เป็นการนำเสนอการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา หัวข้อที่ 3.4 เป็นการนำเสนอวิธีการประเมินความถูกต้องในการล็อคพื้นที่การทำงานของระบบต้นแบบ และหัวข้อที่ 3.5 กล่าวถึงเครื่องมือที่ใช้ในการวิจัย โดยมีรายละเอียดในแต่ละหัวข้อดังต่อไปนี้

3.1 ระเบียบวิธีวิจัย

1. ศึกษาค้นคว้าและรวบรวมงานวิจัยที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ร่วมกันและความขัดแย้งเชิงความหมาย
2. ศึกษาค้นคว้าการสร้างตัวแจงส่วนภาษาจาวาบนเว็บเบราว์เซอร์
3. ศึกษาและค้นคว้าการป้องกันความความขัดแย้งเชิงความหมายที่เกิดจากการพัฒนาซอฟต์แวร์ร่วมกัน และวิธีการนำไปใช้กับระบบต้นแบบ
4. ออกแบบระบบต้นแบบที่ทำงานบนเว็บเบราว์เซอร์ และเทคนิคในการป้องกันความขัดแย้งเชิงความหมายสำหรับใช้ในระบบต้นแบบ
5. ทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา ในด้านของเวลาและหน่วยความจำที่ใช้ในการแจงส่วน
6. ประเมินความถูกต้องในการล็อคพื้นที่การทำงานของระบบต้นแบบด้วยการจำลองการทำงานแบบวนซ้ำ เพื่อศึกษาโอกาสในการเกิดข้อผิดพลาดในด้านของการเกิดความขัดแย้งเชิงความหมาย
7. วิเคราะห์ และสรุปผลการดำเนินการวิจัย

3.2 การออกแบบระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน

ในขั้นตอนของการออกแบบระบบต้นแบบผู้วิจัยแบ่งออกเป็น 3 ส่วนด้วยกันคือ การใช้เครื่องมือสร้างตัวแจงส่วน PEG.js ที่ได้นำเสนอไว้ในบทที่ 2 ในหัวข้อที่ 2.1.1 เรื่องเครื่องมือสร้าง

ตัวแจงส่วน PEG.js เพื่อสร้างตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์ การออกแบบตัวแจงส่วนภาษาจาวาให้สามารถสร้างต้นไม้วากยสัมพันธ์เชิงนามธรรม (Abstract syntax tree) เพื่อที่จะนำต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ได้มาสร้างต้นไม้การพึ่งพาและสถานะล็อค และการออกแบบการทำงานของระบบต้นแบบ

รูปที่ 3.1 การทำงานของระบบต้นแบบโดยรวม

3.2.1 การออกแบบภาพรวมในการทำงานของระบบต้นแบบ

ขั้นตอนในการทำงานของระบบต้นแบบสำหรับการพัฒนาซอฟต์แวร์ร่วมกันบนเว็บเบราว์เซอร์ ด้วยวิธีการการล็อคพื้นที่การทำงานที่เกี่ยวข้องกัน เพื่อป้องกันความขัดแย้งเชิง

ความหมาย มีการออกแบบภาพรวมของระบบโดยมีส่วนที่เกี่ยวข้องกับการทำงานดังแสดงในรูปที่ 3.1 ซึ่งมีรายละเอียดในแต่ละขั้นตอนดังต่อไปนี้

1. รหัสต้นฉบับภาษาจาวา

เป็นการนำเข้ารหัสต้นฉบับภาษาจาวา ซึ่งเป็นรหัสต้นฉบับที่นักพัฒนาได้ทำการพัฒนาขึ้นมาบนซอฟต์แวร์การจัดการข้อมูลบนเว็บเบราว์เซอร์ โดยอยู่ในพื้นที่การทำงานส่วนบุคคลและเป็นรหัสต้นฉบับที่มีการพัฒนาร่วมกันกับนักพัฒนาคนอื่นในช่วงเวลาเดียวกัน

2. วิเคราะห์คำและแจงส่วน

ในส่วนของการวิเคราะห์คำและแจงส่วนนั้น จะเป็นการนำตัวแจงส่วนภาษาจาวาที่สร้างขึ้นจากเครื่องมือสำหรับสร้างตัวแจงส่วน PEG.js มาใช้ในการวิเคราะห์คำและแจงส่วนรหัสต้นฉบับภาษาจาวาที่มีการป้อนเข้ามา จะได้ผลลัพธ์เป็นต้นไม้วากยสัมพันธ์เชิงนามธรรม โดยต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ได้นี้ เป็นต้นไม้พื้นฐานที่จะนำไปปรับปรุงให้เหลือเฉพาะส่วนที่จะนำไปใช้งานต่อเท่านั้น

3. สร้างต้นไม้วากยสัมพันธ์เชิงนามธรรม

ต้นไม้วากยสัมพันธ์เชิงนามธรรมเป็นต้นไม้ที่ถูกสร้างขึ้นสำหรับการอธิบายและบอกถึงส่วนประกอบต่าง ๆ ของรหัสต้นฉบับที่กำลังมีการพัฒนาร่วมกันอยู่ในขณะนั้น โดยจะนำส่วนประกอบของรหัสต้นฉบับเหล่านี้ไปเก็บไว้ในแต่ละโหนด เพื่อจะนำต้นไม้วากยสัมพันธ์เชิงนามธรรมไปสร้างเป็นต้นไม้การพึ่งพาและสถานะล็อค สำหรับใช้ในการกำหนดสถานะล็อคให้กับนักพัฒนาโดยอัตโนมัติ

4. สร้างต้นไม้การพึ่งพาและสถานะล็อค

ต้นไม้การพึ่งพาและสถานะล็อคจะเป็นต้นไม้ที่อธิบายว่าส่วนใดของรหัสต้นฉบับมีความเกี่ยวข้องหรือมีความสัมพันธ์กัน ดังตัวอย่างที่อธิบายในบทที่ 2 หัวข้อที่ 2.3 เรื่องการแก้ไขความขัดแย้งเชิงความหมาย ซึ่งต้นไม้การพึ่งพาและสถานะล็อคจะเป็นต้นไม้ที่บอกว่าเมื่อมีการทำงานบนส่วนหนึ่งของรหัสต้นฉบับควรล็อคส่วนของรหัสต้นฉบับที่กำลังทำงานอยู่พร้อมกับส่วนที่มีการพึ่งพากันของรหัสต้นฉบับ เพื่อป้องกันความขัดแย้งเชิงความหมายที่อาจเกิดขึ้นจากการที่นักพัฒนาหลายคนทำงานบนพื้นที่เดียวกันในช่วงระยะเวลาเดียวกัน

นอกจากนี้ยังสามารถนำต้นไม้การพึ่งพาและสถานะล็อคนี้ไปใช้ในการตรวจสอบว่ามีนักพัฒนาทำงานอยู่บนส่วนใดของรหัสต้นฉบับบ้าง โดยการตรวจสอบหากพบว่ามีนักพัฒนาทำงานอยู่บนรหัสต้นฉบับส่วนใดอยู่รหัสต้นฉบับส่วนนั้นจะถูกล็อค ซึ่งจะเป็นผลทำให้นักพัฒนา

คนอื่นไม่สามารถทำงานบนรหัสต้นฉบับส่วนนั้นได้ จำเป็นต้องรอให้นักพัฒนาที่เป็นเจ้าของส่วนของรหัสต้นฉบับนั้น ออกจากส่วนของรหัสต้นฉบับดังกล่าวก่อนจึงจะสามารถเข้าไปทำงานในส่วนของรหัสต้นฉบับที่ต้องการได้ แต่ในกรณีที่รหัสต้นฉบับส่วนที่ต้องการจะพัฒนาไม่ได้ถูกล็อคพื้นที่อยู่ นักพัฒนาสามารถเข้าไปทำงานในส่วนของรหัสต้นฉบับดังกล่าวได้ทันที

5. ปรับปรุงสถานะล็อค

การปรับปรุงสถานะล็อคจะเกิดขึ้นเมื่อนักพัฒนาทำงานบนส่วนของรหัสต้นฉบับที่ไม่มีนักพัฒนาคนอื่นเป็นเจ้าของอยู่ โดยการปรับปรุงสถานะล็อคจะดูจากต้นไม้การพึ่งพาและสถานะล็อคว่าส่วนที่นักพัฒนาทำงานอยู่นั้นพึ่งพากันกับส่วนอื่นของรหัสต้นฉบับอยู่หรือไม่ หากพึ่งพากันอยู่จะล็อคส่วนที่พึ่งพากันด้วย หากไม่พึ่งพากันกับรหัสต้นฉบับส่วนอื่นก็จะล็อคเฉพาะส่วนที่นักพัฒนาทำงานอยู่ ซึ่งเมื่อทราบถึงส่วนที่จะล็อคภายในรหัสต้นฉบับแล้วต้นไม้การพึ่งพาและสถานะล็อคจะถูกสร้างขึ้นใหม่เพื่อรอการปรับปรุงสถานะล็อคต่อไป

3.2.2 การออกแบบตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์

ในการออกแบบตัวแจงส่วนภาษาจาวาให้ทำงานได้บนเว็บเบราว์เซอร์นั้น ผู้วิจัยได้ใช้เครื่องมือสร้างตัวแจงส่วน PEG.js ซึ่งในการทำงานของเครื่องมือนี้จะทำงานโดยการป้อนไวยากรณ์ของภาษาที่ต้องการเข้าไป แล้วจะให้ผลลัพธ์ออกมาเป็นตัวแจงส่วนที่อยู่ในรูปแบบของไฟล์จาวาสคริปต์ ทำให้สามารถนำไปใช้งานบนเว็บเบราว์เซอร์ได้ ในการเรียกใช้งานตัวแจงส่วนภาษาจาวาที่อยู่ในรูปแบบของไฟล์จาวาสคริปต์นั้นจะเหมือนกับการเรียกใช้งานไฟล์จาวาสคริปต์ทั่วไป หากเป็นการเรียกใช้งานบนเว็บที่พัฒนาด้วยภาษาเอชทีเอ็มแอล สามารถเรียกใช้งานตัวแจงส่วนด้วยการเรียกผ่านแท็กสคริปต์ได้ การสร้างตัวแจงส่วนภาษาจาวาในงานวิจัยนี้ได้กำหนดให้สามารถแจงส่วนภาษาได้เฉพาะไวยากรณ์ภาษาจาวารุ่นที่ 1.6 เท่านั้น โดยได้พัฒนาตัวแจงส่วนที่ใช้กับไวยากรณ์ภาษาจาวาต่อยอดมาจาก Morgan Ericsson (2013) ซึ่งความสามารถที่ Ericsson ได้ทำการพัฒนาไว้นั้นเป็นการเรียกใช้งานผ่านทางบรรทัดคำสั่ง ทั้งนี้ผู้วิจัยได้ทำการพัฒนาเพิ่มเพื่อให้สามารถเรียกใช้งานผ่านทางเว็บเบราว์เซอร์ และได้เพิ่มการเก็บบรรทัดการทำงานเพื่อใช้ในงานวิจัยนี้

สำหรับการพัฒนาตัวแจงส่วนภาษาจาวา จะประกอบไปด้วยการสร้างส่วนตัววิเคราะห์คำ และส่วนการแจงส่วนของภาษา โดยจะแสดงตัวอย่างและอธิบายไวยากรณ์ที่ใช้ในการสร้างตัวแจงส่วนภาษาจาวาด้วยเครื่องมือสร้างตัวแจงส่วน PEG.js ดังต่อไปนี้

1. ส่วนวิเคราะห์คำ

ในส่วนนี้จะเป็นส่วนที่รับข้อมูลประโยคเข้าแล้วแยกออกเป็นชิ้น ๆ แต่ละชิ้นจะถูกเรียกว่า โทเค็น ซึ่งจะแยกเป็นโทเค็นอะไรบ้างขึ้นอยู่ที่ผู้ใช้งานจะกำหนด ส่วนวิเคราะห์คำมีหน้าที่ในการแยกโทเค็นแล้วนำแต่ละโทเค็นไปสร้างเป็นไวยากรณ์ของภาษาที่ต้องการ เพื่อนำไปตรวจสอบข้อมูลประโยคที่เข้ามาว่าเป็นไปตามไวยากรณ์ที่กำหนดหรือไม่ ดังแสดงในรูปที่ 3.2 เป็นการแสดงการสร้างโทเค็นของคำสงวน (Keywords) ในภาษาจาวา

```
ASSERT    = t:"assert" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
BREAK     = t:"break" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
CASE      = t:"case" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
CATCH     = t:"catch" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
CLASS     = t:"class" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
CONTINUE = t:"continue" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
DEFAULT   = t:"default" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
DO        = t:"do" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
ELSE      = t:"else" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
ENUM      = t:"enum" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
EXTENDS   = t:"extends" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
FINALLY   = t:"finally" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
FINAL     = t:"final" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
FOR       = t:"for" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
IF        = t:"if" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
IMPLEMENTS = t:"implements" !LetterOrDigit Spacing
             { return new nodes.Terminal(t);}
IMPORT     = t:"import" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
INTERFACE  = t:"interface" !LetterOrDigit Spacing
             { return new nodes.Terminal(t);}
INSTANCEOF = t:"instanceof" !LetterOrDigit Spacing
             { return new nodes.Terminal(t);}
NEW       = t:"new" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
PACKAGE   = t:"package" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
RETURN    = t:"return" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
STATIC    = t:"static" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
SUPER     = t:"super" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
SWITCH    = t:"switch" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
SYNCHRONIZED = t:"synchronized" !LetterOrDigit Spacing
               { return new nodes.Terminal(t);}
THIS      = t:"this" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
THROWS    = t:"throws" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
THROW     = t:"throw" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
TRY       = t:"try" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
VOID      = t:"void" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
WHILE     = t:"while" !LetterOrDigit Spacing { return new nodes.Terminal(t);}
```

รูปที่ 3.2 ตัวอย่างการสร้างโทเค็นของคำสงวนในภาษาจาวา

ยกตัวอย่างโทเค็น ASSERT ซึ่งจะมีส่วนประกอบต่าง ๆ ของการสร้างโทเค็นดังแสดงในตัวอย่าง

ASSERT = t:"assert" !LetterOrDigit Spacing

{ return new nodes.Terminal(t);}

คำสั่งที่อยู่ทางด้านซ้ายของเครื่องหมายเท่ากับ จะถูกแทนให้เป็นโทเค็น ซึ่งก็คือ ASSERT และคำสั่งที่อยู่ทางด้านขวาของเครื่องหมายเท่ากับ ซึ่งก็คือ t:"assert" !LetterOrDigit Spacing จะเป็นรายละเอียดและเงื่อนไขในการสร้างโทเค็น ที่มีการเก็บค่าไว้ในตัวแปร t สำหรับไว้ใช้ในคำสั่ง return new nodes.Terminal(t) ซึ่งเป็นคำสั่งในการสร้างโหนดต่อไป

```
ClassDeclaration = c:CLASS i:Identifier t:TypeParameters? e:(EXTENDS ClassType)?
                   im:(IMPLEMENTS ClassTypeList)? b:ClassBody { return new
                   nodes.ASTNode('ClassDeclaration', pr([ c, i, t, e, im, b ]),
                   createLog(line, column)); }

Identifier = !Keyword l1:Letter ls:LetterOrDigit* Spacing { return new
             nodes.ASTNode('Identifier', [ new nodes.Terminal(l1+ls.join('')) ],
             createLog(line, column)); }

TypeParameters = l:LPOINT t:TypeParameter e:( COMMA TypeParameter )* r:RPOINT {
                 return new nodes.ASTNode('TypeParameters', pr([ l, t, e, r ]),
                 createLog(line, column)); }

ClassType  = i:Identifier t:TypeArguments? r:( DOT Identifier TypeArguments? )* {
             return new nodes.ASTNode('ClassType', pr([ i, t, r]), createLog(
             line,column)); }

ClassTypeList = c:ClassType r:( COMMA ClassType )* { return new nodes.ASTNode(
                'ClassTypeList', pr([ c , r]), createLog(line, column)); }

ClassBody  = l:LWING c:ClassBodyDeclaration* r:RWING { return new nodes.ASTNode(
             'ClassBody', pr([l,c,r]), createLog(line, column));}

CLASS      = t:"class" !LetterOrDigit Spacing { return new nodes.Terminal(t);}

EXTENDS    = t:"extends" !LetterOrDigit Spacing { return new nodes.Terminal(t);}

IMPLEMENTS = t:"implements" !LetterOrDigit Spacing
             { return new nodes.Terminal(t);}
```

รูปที่ 3.3 ตัวอย่างไวยากรณ์ที่ใช้กับเครื่องมือ PEG.js ในส่วนของการประกาศคลาสในภาษาจาวา

2. ส่วนการแจงส่วนของภาษา

ในส่วนของตัวแจงส่วนของภาษานั้น จะเป็นการนำโทเค็นที่สร้างขึ้นมาจัดเรียงหรือเชื่อมต่อกันใหม่ เพื่อให้ส่วนการแจงส่วนสามารถแจงส่วนข้อมูลประโยคตามที่ต้องการได้ โดยส่วนการแจงส่วนของภาษานั้นจะมีหน้าที่ตรวจสอบว่าข้อมูลประโยคที่เข้ามาเป็นไปตามโทเค็นที่จัดเรียงไว้หรือไม่ ดังแสดงในรูปที่ 3.3 เป็นส่วนของการประกาศคลาสในภาษาจาวา ที่แทนด้วย

ส่วนการแจงส่วน ClassDeclaration โดยมีการจัดเรียงส่วนการแจงส่วนและโทเค็นที่เป็นส่วนประกอบของการประกาศคลาสคือ โทเค็น CLASS ส่วนการแจงส่วน Identifier ส่วนการแจงส่วน TypeParameters และส่วนการแจงส่วน ClassBody ที่จำเป็นต้องมีในการประกาศคลาสของภาษาจาวา และอาจมีโทเค็น EXTENDS ส่วนการแจงส่วน ClassType โทเค็น IMPLEMENTS และส่วนการแจงส่วน ClassTypeList หรือไม่ก็ได้

3.2.3 การออกแบบต้นไม้สำหรับนำไปใช้ในระบบต้นแบบ

ในการออกแบบต้นไม้ที่จะนำไปใช้ในระบบต้นแบบ จะมีต้นไม้ที่เกี่ยวข้องกันอยู่ 2 ชนิดด้วยกันคือ 1) การออกแบบต้นไม้วากยสัมพันธ์เชิงนามธรรม 2) การออกแบบต้นไม้การพึ่งพาและสถานะล็อค โดยมีรายละเอียดของการออกแบบต้นไม้ทั้ง 2 ชนิด ดังต่อไปนี้

```
exports.ASTNode = new JS.Class({
        init: function(type, children, log) {
                this.type = type;
                this.children = children;
                this.log = log;
        }
})


exports.Terminal = new JS.Class({
        init: function(value) {
                this.value = value;
        }
})
```

รูปที่ 3.4 รหัสต้นฉบับสำหรับสร้างโหนดเพื่อจัดเก็บข้อมูลการแจงส่วน

1. การออกแบบต้นไม้วากยสัมพันธ์เชิงนามธรรม

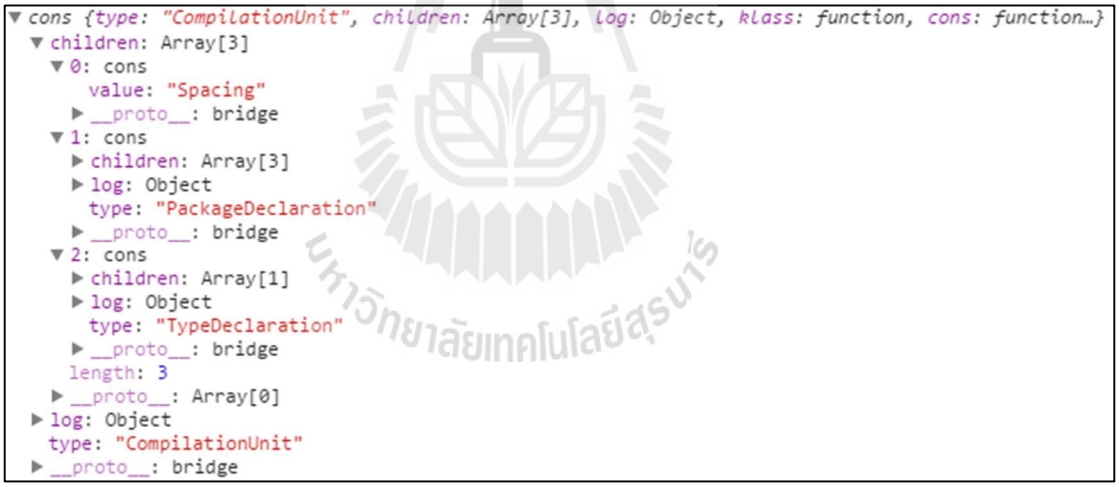
ในการออกแบบต้นไม้วากยสัมพันธ์เชิงนามธรรมนั้น จะสร้างโดยกำหนดให้แต่ละโหนดมีการเก็บชื่อของโหนดนั้น ๆ รวมถึงการเก็บข้อมูลต่าง ๆ ของโหนดในระดับที่ต่ำกว่า และเก็บหมายเลขของบรรทัดและคอลัมน์ของโหนดนั้น ๆ เพื่อใช้ในการอ้างอิงถึงพื้นที่การทำงานภายในรหัสต้นฉบับ ซึ่งจะเป็นประโยชน์ในการกำหนดสถานะล็อคให้กับระบบต้นแบบ จากรูปที่ 3.4 เป็นรหัสต้นฉบับสำหรับสร้างต้นไม้วากยสัมพันธ์เชิงนามธรรม ซึ่งประกอบไปด้วยการสร้าง ASTNode และการสร้าง Terminal โดยใช้คลัง (Library) สำหรับการสร้างต้นไม้ คือ jsclass (Coglan, 2014) ซึ่งผลลัพธ์ที่ได้มาจะเป็นต้นไม้วากยสัมพันธ์เชิงนามธรรมดังแสดงในรูปที่ 3.5

การสร้าง ASTNode จะประกอบไปด้วยตัวแปรดังต่อไปนี้

- **ตัวแปร type :** เป็นตัวแปรสำหรับเก็บชื่อของโหนดแต่ละโหนดในต้นไม้วากยสัมพันธ์เชิงนามธรรม
- **ตัวแปร children :** เป็นตัวแปรสำหรับเก็บโหนดที่อยู่ในระดับที่ต่ำกว่าในต้นไม้วากยสัมพันธ์เชิงนามธรรม โดยหากโหนดในระดับที่ต่ำกว่าเป็นโหนดสุดท้าย (Terminal node) จะเก็บค่าคงที่ของโหนดสุดท้ายแทน
- **ตัวแปร log :** เป็นตัวแปรสำหรับเก็บบรรทัดที่โหนดเริ่มทำงาน โดยแยกเก็บเป็น 2 ส่วน ประกอบด้วย 1) ส่วนของ line คือ บรรทัดที่โหนดเริ่มทำงาน และ 2) ส่วนของ colume คือ คอลัมน์ที่โหนดเริ่มทำงาน

ส่วนการสร้าง Terminal จะมีเพียงหนึ่งตัวแปรคือ

- **ตัวแปร value :** เป็นตัวแปรสำหรับเก็บค่าคงที่ของโหนดสุดท้าย



รูปที่ 3.5 ต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ได้จากการรหัสต้นฉบับสำหรับสร้างโหนด

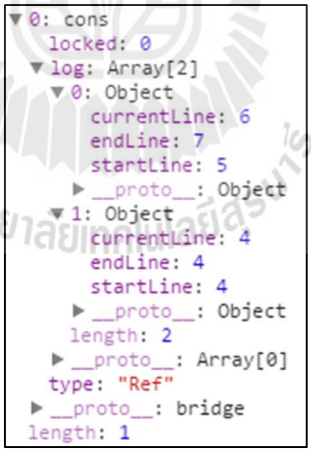
2. การออกแบบต้นไม้การพึ่งพาและสถานะล็อค

การออกแบบต้นไม้การพึ่งพาและสถานะล็อคได้ออกแบบให้มีลักษณะเหมือนกับตัวอย่างกราฟการพึ่งพากันที่อธิบายในบทที่ 2 หัวข้อที่ 2.3 เรื่องการจัดการความขัดแย้งเชิงความหมาย ซึ่งต้นไม้การพึ่งพาและสถานะล็อคจะเป็นตัวบ่งบอกว่าส่วนใดของรหัสต้นฉบับมีความเกี่ยวข้องกัน กล่าวคือเมื่อมีพื้นที่การล็อคเกิดขึ้น ต้นไม้การพึ่งพาและสถานะล็อคจะเป็นต้นไม้ที่

บอกว่าจะมีส่วนอื่นของรหัสต้นฉบับที่จะถูกล็อคตามไปด้วยหรือไม่ การสร้าง REFNode จากคำสั่งดังแสดงในรูปที่ 3.6 โดยใช้คลังสำหรับการสร้างต้นไม้ jsclass ซึ่งผลลัพธ์ที่ได้มาจะเป็นต้นไม้การพึ่งพาและสถานะล็อคดังแสดงในรูปที่ 3.7 เมื่อมีการล็อคเกิดขึ้นต้นไม้การพึ่งพาและสถานะล็อคจะทำการปรับปรุงสถานะล็อค ซึ่งการล็อคเหล่านี้จะเกิดขึ้นตามการเลื่อนเคอร์เซอร์ของนักพัฒนา

```
exports.REFNode = new JS.Class({
        init: function(type, log, locked) {
                this.type = type;
                this.log = log;
                this.locked = locked;
        }
})
```

รูปที่ 3.6 รหัสต้นฉบับสำหรับสร้างโหนดที่ใช้ในการสร้างต้นไม้การพึ่งพาและสถานะล็อค



รูปที่ 3.7 ต้นไม้การพึ่งพาและสถานะล็อคจากการรหัสต้นฉบับสำหรับสร้างโหนด

การสร้าง REFNode จะประกอบไปด้วยตัวแปรดังต่อไปนี้

- **ตัวแปร type :** เป็นตัวแปรสำหรับเก็บชื่อของโหนดแต่ละโหนดในต้นไม้การพึ่งพาและสถานะล็อค โดยมีค่าเริ่มต้น คือ “Ref”
- **ตัวแปร log :** เป็นตัวแปรสำหรับเก็บบรรทัดที่มีการพึ่งพากัน โดยแบ่ง

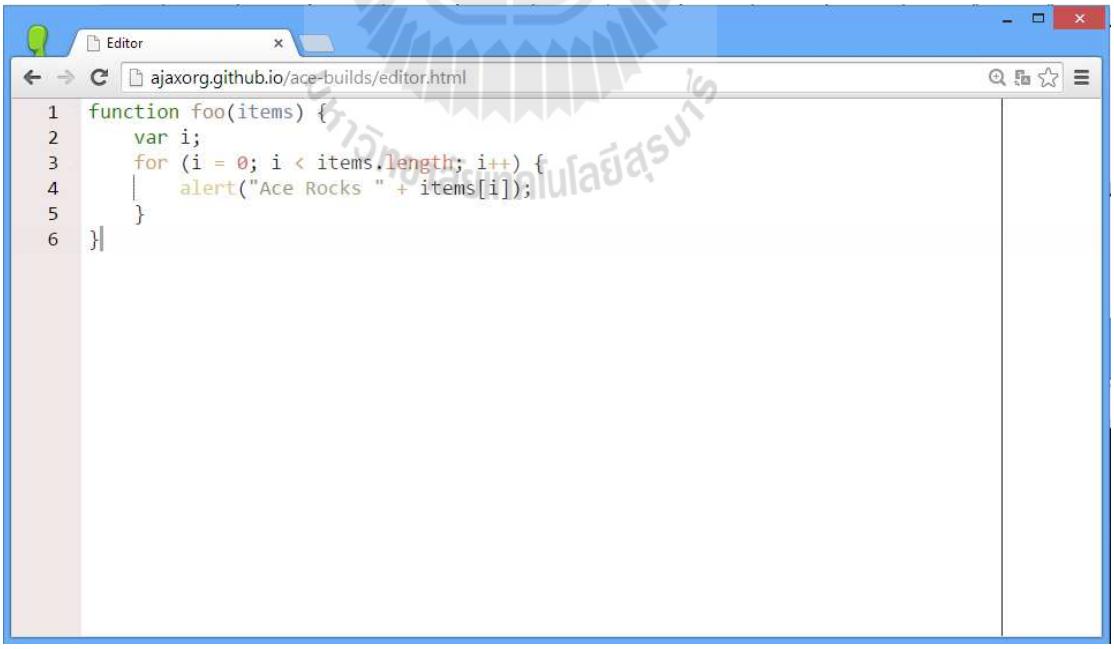
ออกเป็น 2 ส่วน ประกอบด้วย 1) บรรทัดที่เรียกใช้ตัวแปรหรือเมธอดอื่น และ 2) บรรทัดของตัวแปรหรือเมธอดที่ถูกเรียกใช้

- **ตัวแปร locked :** เป็นตัวแปรสำหรับเก็บไอดีของเจ้าของพื้นที่การทำงานที่มีการพึ่งพากัน โดยมีค่าเริ่มต้นเป็น 0

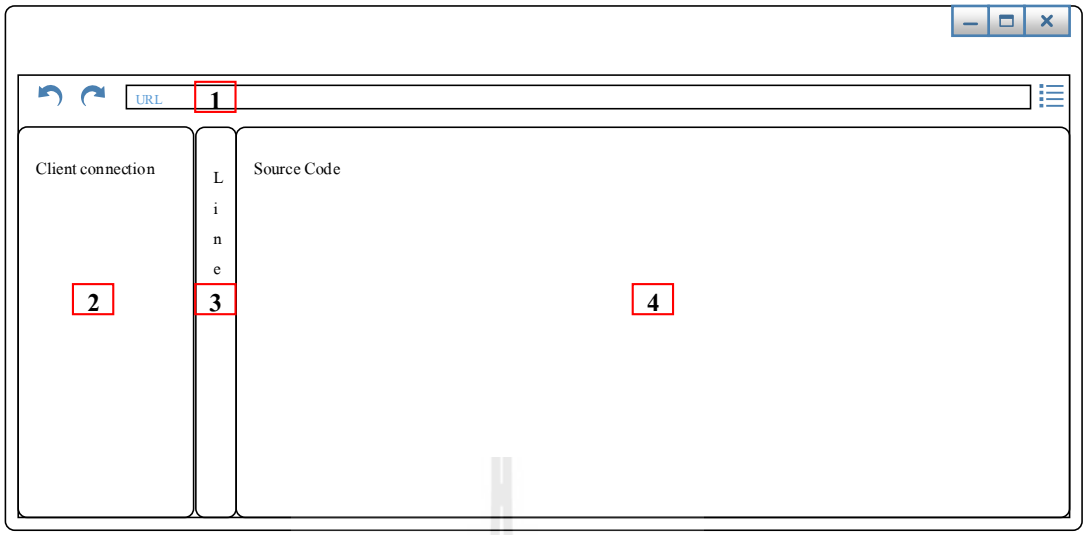
3.2.4 การพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน

ในการพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน จะเริ่มจากการพัฒนาซอฟต์แวร์การจัดการข้อมูลบนเว็บเบราว์เซอร์ ผู้วิจัยได้เลือกใช้เครื่องมือที่เรียกว่า ACE ซึ่งเป็นซอฟต์แวร์การจัดการข้อมูลเบื้องต้น ที่นิยมนำไปใช้สำหรับการพัฒนาซอฟต์แวร์การจัดการข้อมูลที่มีลักษณะเฉพาะ โดยเลือกใช้ Node.js สำหรับการรันไฟล์จาวาสคริปต์ และเพิ่มโมดูล socket.io (Schlueter, 2014) ที่เป็นโมดูลสำหรับการเชื่อมต่อกันระหว่างเซิร์ฟเวอร์กับไคลเอนต์

ซอฟต์แวร์ ACE (The ACE project, 2014) เป็นซอฟต์แวร์สำหรับการจัดการข้อมูลที่เป็นโอเพนส์ซอร์ส (Open source) เมื่อทำการดาวน์โหลดไฟล์ และทำการรันไฟล์ตัวอย่างในส่วนที่เราจะนำมาพัฒนาต่อจะได้ซอฟต์แวร์สำหรับการจัดการข้อมูลเบื้องต้นดังแสดงในรูปที่ 3.8 ซึ่งเป็นลักษณะการทำงานแบบเดี่ยว (Standalone) และสามารถเน้นคำสงวนของภาษาที่นำมาใช้ได้



รูปที่ 3.8 ซอฟต์แวร์ ACE สำหรับการจัดการข้อมูลเบื้องต้นที่รันบนเว็บเบราว์เซอร์

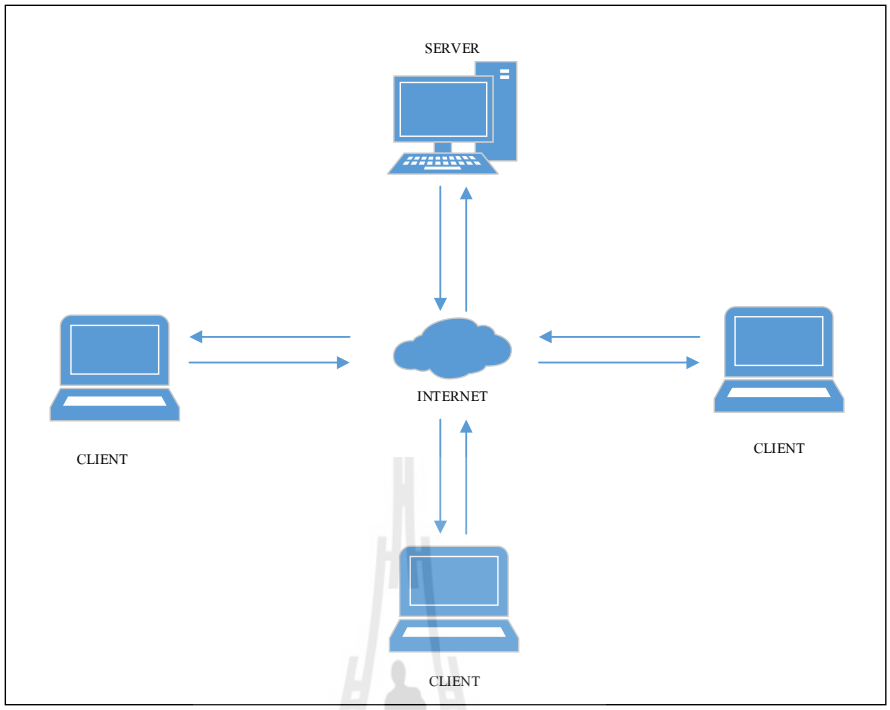


รูปที่ 3.9 การออกแบบส่วนติดต่อกับผู้ใช้

การออกแบบส่วนติดต่อกับผู้ใช้ได้ทำการออกแบบดังแสดงในรูปที่ 3.9 ซึ่งประกอบไปด้วยส่วนต่าง ๆ ดังต่อไปนี้

- **หมายเลข 1 :** ช่องสำหรับกรอก URL ของเครื่องเซิร์ฟเวอร์
- **หมายเลข 2 :** การแสดงจำนวนของผู้พัฒนาที่มีการเชื่อมต่อเข้าไปยังเซิร์ฟเวอร์
- **หมายเลข 3 :** การแสดงหมายเลขของบรรทัดในการพัฒนารหัสต้นฉบับ
- **หมายเลข 4 :** ส่วนของพื้นที่การทำงานสำหรับการพัฒนาซอร์ดโค้ด

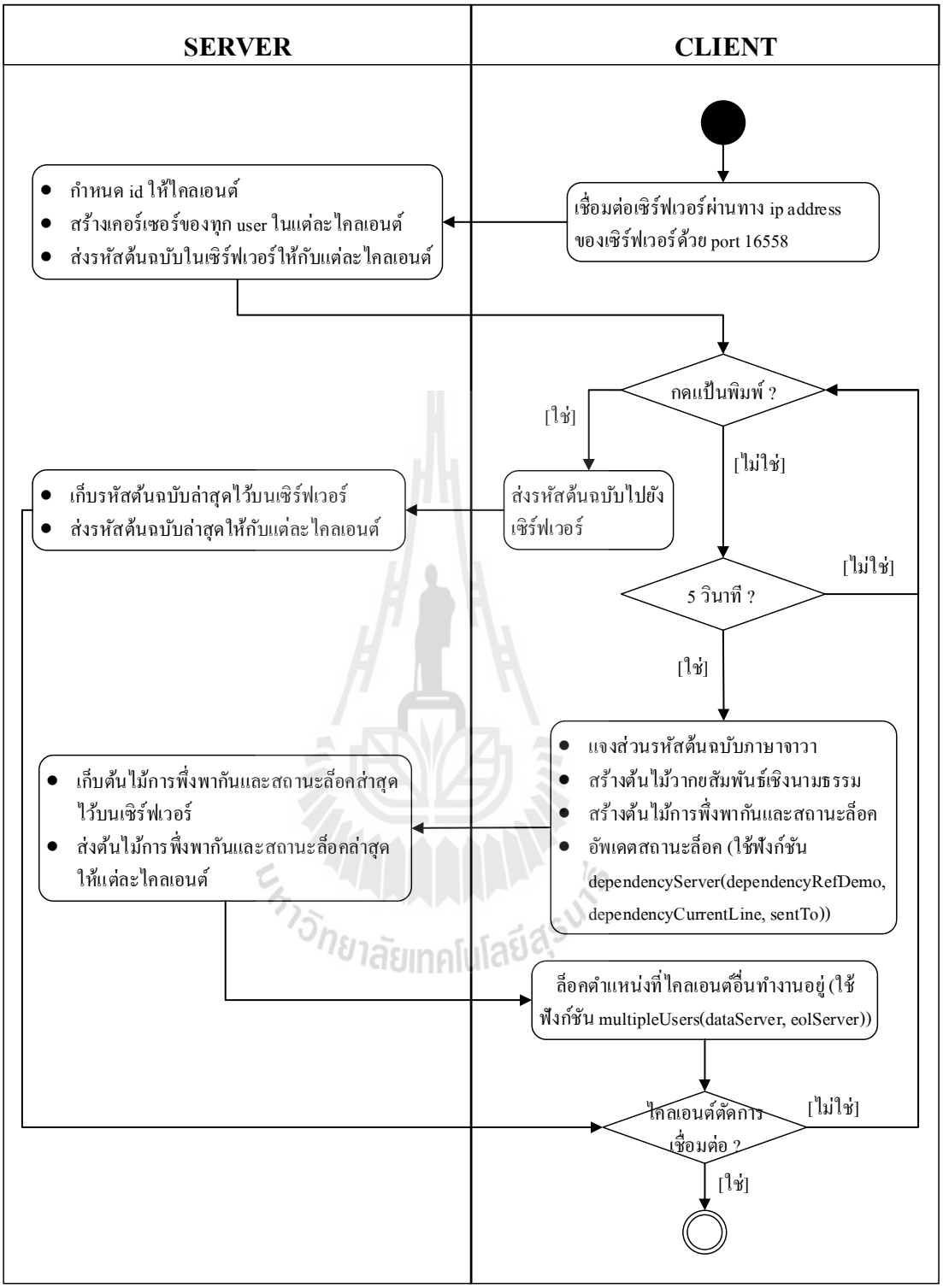
ในส่วนของการออกแบบภาพรวมระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกันบนเว็บเบราว์เซอร์ มีการออกแบบดังแสดงในรูปที่ 3.10 โดยมีประกอบไปด้วยเครื่องเซิร์ฟเวอร์ สวิตซ์ และเครื่องไคลเอนต์



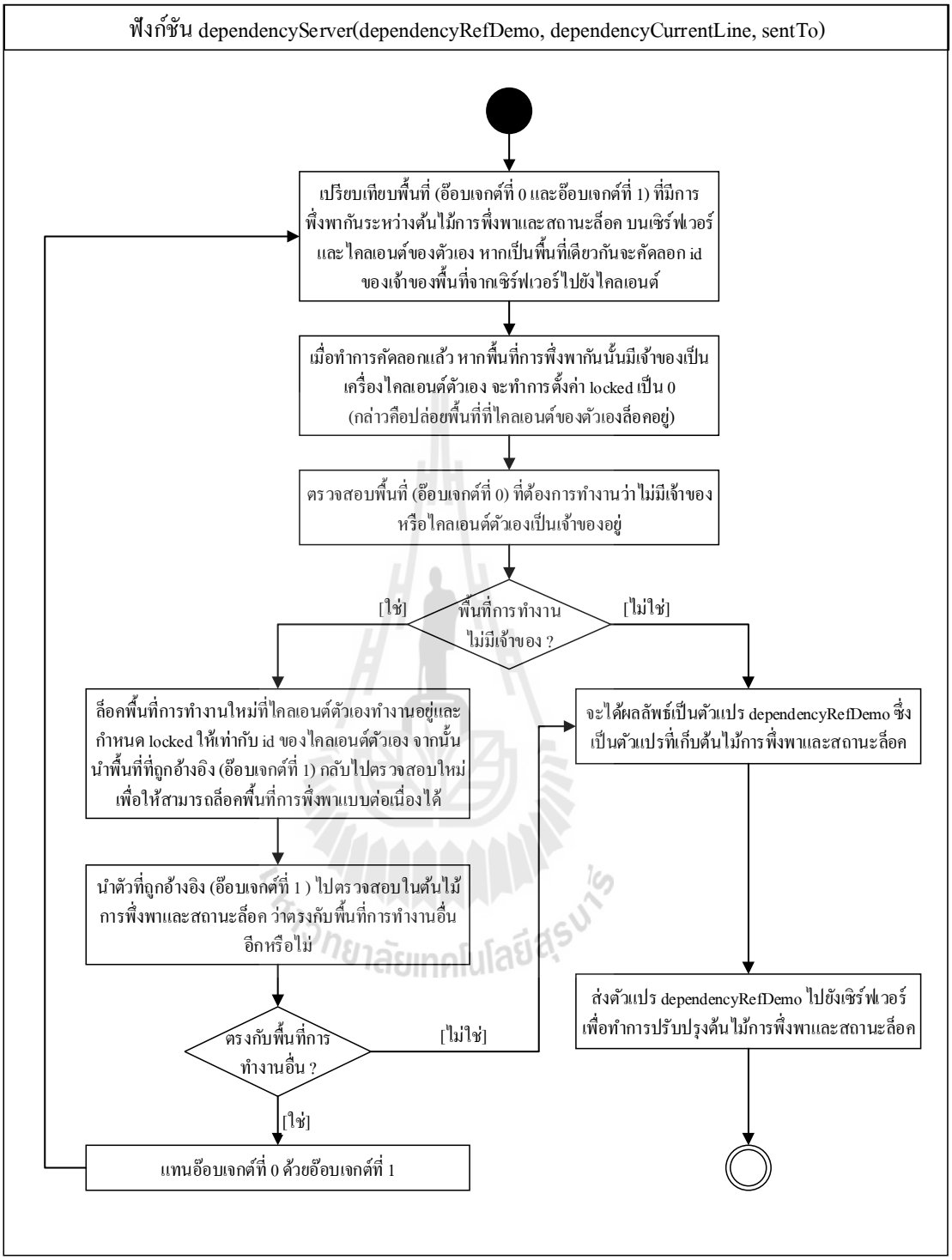
รูปที่ 3.10 ภาพรวมของระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกันบนเว็บเบราว์เซอร์

การออกแบบระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับด้วยลักษณะของภาพรวมระบบดังที่ได้กล่าวมานั้น มีการออกแบบลักษณะของการเชื่อมต่อและการทำงานระหว่างเครื่องเซิร์ฟเวอร์และเครื่องไคลเอนต์ดังแสดงในรูปที่ 3.11

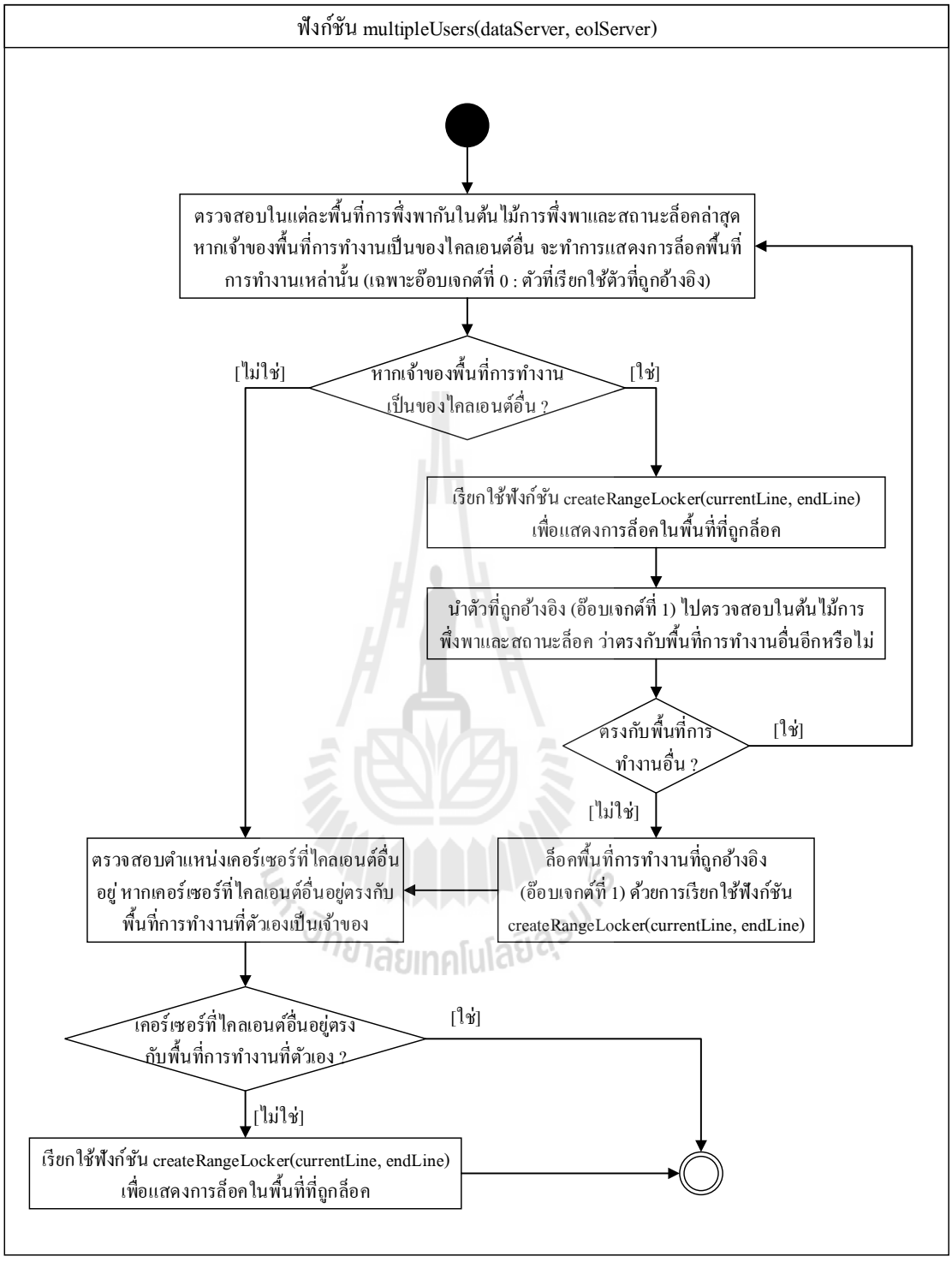
สำหรับการพัฒนาให้ระบบต้นแบบสามารถล็อคพื้นที่การทำงานได้นั้น จะพัฒนาให้สามารถกำหนดเจ้าของพื้นที่การทำงานให้กับต้นไม้การพึ่งพาและสถานะล็อค จะถูกพัฒนาเป็นฟังก์ชัน dependencyServer(dependencyRefDemo, dependencyCurrentLine, sentTo) โดยมีลักษณะการทำงานดังแสดงในรูปที่ 3.12 และการแสดงพื้นที่การทำงานที่ถูกล็อคบนไคลเอนต์ของตัวเอง จะถูกพัฒนาเป็นฟังก์ชัน multipleUsers(dataServer, eolServer) โดยมีลักษณะการทำงานดังแสดงในรูปที่ 3.12



รูปที่ 3.11 แผนภาพการทำงานของระบบต้นแบบระหว่างเครื่องเซิร์ฟเวอร์และเครื่องไคลเอนต์



รูปที่ 3.12 แผนภาพการทำงานของฟังก์ชัน dependencyServer(dependencyRefDemo, dependencyCurrentLine, sentTo)



รูปที่ 3.13 แผนภาพการทำงานของฟังก์ชัน multipleUsers(dataServer, eolServer)

3.3 การออกแบบการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา

การออกแบบการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา จะทำการทดสอบโดยใช้ตัวแจงส่วนที่ได้ทำการออกแบบไว้ในหัวข้อที่ 3.2.2 เรื่องการออกแบบตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์ ในการทดสอบประสิทธิภาพจะทำการทดสอบทั้งหมด 2 ด้าน คือ

- การทดสอบเวลาที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวา
- การทดสอบหน่วยความจำที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวา

โดยใช้ข้อมูลประโยคภาษาจาวาจากโปรเจค H2 ซึ่งเป็นระบบฐานข้อมูลที่พัฒนาด้วยภาษาจาวา (Mueller, 2013) ที่ได้มีการพัฒนาอย่างต่อเนื่อง ด้วยการเลือกไฟล์ที่มีนามสกุลจาวาจากโปรเจค H2 จำนวน 4 ไฟล์ ในการเลือกไฟล์นั้นใช้วิธีการดูจากบรรทัดของรหัสต้นฉบับภายในไฟล์ ซึ่งทั้ง 4 ไฟล์ที่ได้ทำการเลือกมีจำนวนบรรทัดของแต่ละไฟล์คือ

- ไฟล์ที่ 1 มีจำนวนบรรทัดทั้งหมด 344 บรรทัด
- ไฟล์ที่ 2 มีจำนวนบรรทัดทั้งหมด 1,395 บรรทัด
- ไฟล์ที่ 3 มีจำนวนบรรทัดทั้งหมด 2,511 บรรทัด
- ไฟล์ที่ 4 มีจำนวนบรรทัดทั้งหมด 5,535 บรรทัด ซึ่งเป็นไฟล์ที่มีจำนวนบรรทัดมากที่สุดของโปรเจค H2

ในการทดสอบประสิทธิภาพจะทดสอบจากการแจงส่วนข้อมูลประโยคของภาษาจาวา ด้วยตัวแจงส่วนภาษาจาวาที่อยู่ในรูปแบบของไฟล์จาวาสคริปต์เพื่อให้ได้ต้นไม้วากยสัมพันธ์เชิงนามธรรม โดยการทดสอบประสิทธิภาพจะทำบนเว็บเบราว์เซอร์ที่แตกต่างกัน 5 เว็บเบราว์เซอร์ ได้แก่

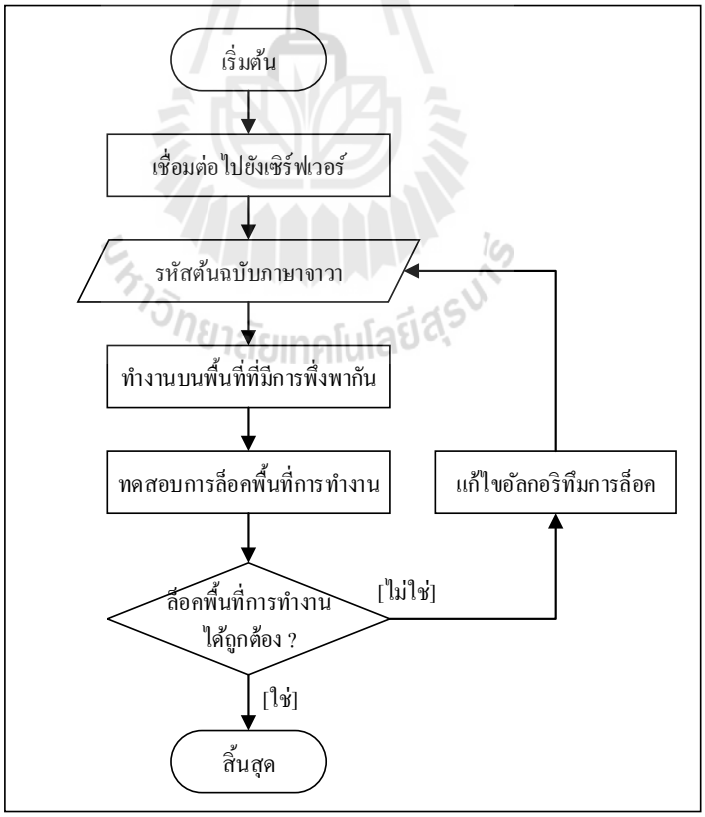
- เว็บเบราว์เซอร์กูเกิลโครม รุ่นที่ 31
- เว็บเบราว์เซอร์มอซิลลาไฟร์ฟอกซ์ (Mozilla firefox) รุ่นที่ 25
- เว็บเบราว์เซอร์ซาฟารี (Safari) รุ่นที่ 5
- เว็บเบราว์เซอร์โอเปร่า (Opera) รุ่นที่ 18
- เว็บเบราว์เซอร์กูเกิลโครมคานารี (Google Chrome Canary) รุ่นที่ 33

ในการทดสอบการแจงส่วนข้อมูลประโยคภาษาจาวาในไฟล์แต่ละไฟล์ที่นำมาทดสอบจะทำการทดสอบไฟล์ละ 10 รอบ และจับเวลาในแต่ละรอบ เมื่อครบทั้ง 10 รอบแล้วจะเรียงเวลาที่ได้จาก

น้อยไปมาก และทำการตัดเวลาที่น้อยที่สุด 2 อันดับและมากที่สุด 2 อันดับออก แล้วจึงนำเวลา 6 อันดับที่เหลือมาหาค่าเฉลี่ย

3.4 การออกแบบการประเมินความถูกต้องของระบบต้นแบบ

การประเมินประสิทธิภาพของระบบต้นแบบนั้น จะใช้เทคนิคการจำลองการใช้งานแบบวนซ้ำ ซึ่งเป็นการจำลองการทำงานของนักพัฒนา 2 คนบนพื้นที่การทำงานที่ได้ออกแบบแยกตามกรณีทดสอบ เพื่อประเมินความถูกต้องของการล็อคพื้นที่การทำงาน เครื่องมือที่สนับสนุนการประเมินประสิทธิภาพแบบดังกล่าวมีด้วยกันมากมาย สำหรับงานวิจัยนี้จะใช้ซิลิเนียมเว็บไดร์เวอร์ (Selenium WebDriver) (Jmleyba, 2014) เป็นเครื่องมือที่ใช้ในการจำลองการทำงานบนเว็บเบราว์เซอร์แบบอัตโนมัติ การประเมินความถูกต้องนั้นจะเป็นลักษณะแบบวนซ้ำเพื่อหาเปอร์เซ็นต์ของความผิดพลาดของระบบต้นแบบที่อาจจะเกิดขึ้น โดยการทำงานของตัวทดสอบจะมีลักษณะดังแสดงในรูปที่ 3.14



รูปที่ 3.14 ขั้นตอนการประเมินและปรับแก้ประสิทธิภาพการล็อค

ผู้วิจัยได้ออกแบบกรณีทดสอบสำหรับทดสอบระบบต้นแบบ โดยกำหนดให้นักพัฒนาที่ 1 เป็นเจ้าของพื้นที่การทำงานก่อนเสมอ กรณีทดสอบที่ได้ทำการออกแบบมีทั้งหมด 11 กรณี ดังต่อไปนี้

กรณีที่ 1 : นักพัฒนาที่ 2 ทำงานบนพื้นที่ที่นักพัฒนาที่ 1 ไม่ได้เป็นเจ้าของ

กรณีที่ 2 : นักพัฒนาที่ 2 ทำงานบนบรรทัดเดียวกันกับที่นักพัฒนาที่ 1 กำลังทำงานอยู่

กรณีที่ 3 : นักพัฒนาที่ 2 ทำงานบนตัวแปรที่ถูกล็อคจากการเรียกใช้ตัวแปรของนักพัฒนาที่ 1 ที่ทำงานอยู่ภายในคลาสเดียวกัน

กรณีที่ 4 : นักพัฒนาที่ 2 ทำงานบนเมธอดที่นักพัฒนาที่ 1 กำลังทำงานอยู่

กรณีที่ 5 : นักพัฒนาที่ 2 ทำงานบนคอนสตรัคเตอร์ของคลาสแม่ที่ถูกล็อคจากการเรียกใช้ super() ของนักพัฒนาที่ 1 ภายในคลาสลูก

กรณีที่ 6 : นักพัฒนาที่ 2 ทำงานบนเมธอดที่ถูกล็อคจากการเรียกใช้เมธอดในคลาสอื่นของนักพัฒนาที่ 1

กรณีที่ 7 : นักพัฒนาที่ 2 ทำงานบนตัวแปรที่ถูกล็อคจากการเรียกใช้ตัวแปรในคลาสอื่นของนักพัฒนาที่ 1

กรณีที่ 8 : นักพัฒนาที่ 2 ทำงานบนตัวแปรที่ถูกล็อคจากการเรียกใช้ตัวแปรแบบ this ของนักพัฒนาที่ 1

กรณีที่ 9 : นักพัฒนาที่ 2 ทำงานบนเมธอดของคลาสแม่ที่ถูกล็อคจากการเรียกใช้เมธอด super.method() ของนักพัฒนาที่ 1 ภายในคลาสลูก

กรณีที่ 10 : นักพัฒนาที่ 2 ทำงานบนคอนสตรัคเตอร์ที่ถูกล็อคจากการสร้างอินสแตนซ์ของนักพัฒนาที่ 1 ภายในคลาสอื่น

กรณีที่ 11 : นักพัฒนาที่ 2 ทำงานบนตัวแปรที่ถูกล็อคจากการเรียกใช้ภายในอาร์กิวเมนต์ของเมธอดที่นักพัฒนาที่ 1 ทำงานอยู่

3.5 เครื่องมือที่ใช้ในการวิจัย

เครื่องมือและอุปกรณ์ที่ใช้สำหรับการวิจัยคือเครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับการศึกษาและพัฒนาระบบต้นแบบการพัฒนาซอฟต์แวร์ภาษาจาวาร่วมกันบนเว็บเบราว์เซอร์โดยใช้การล็อคพื้นที่การทำงานเพื่อป้องกันความขัดแย้ง โดยมีรายละเอียดดังนี้

- หน่วยประมวลผลกลาง : Intel® Core™ i3-3120M CPU ความถี่สัญญาณนาฬิกา/ความเร็ว 2.50 GHz
- หน่วยความจำสำรอง : 500 GB

- หน่วยความจำหลัก : 4.00 GB
- ระบบปฏิบัติการ : Windows 8 Pro 64-bit Operating System
- อุปกรณ์อื่น ๆ เช่น เมาส์ แป้นพิมพ์ เป็นต้น
- เครื่องมือสำหรับสร้างตัวแจงส่วนใช้ PEG.js รุ่นที่ 0.7.0
- ซอฟต์แวร์จัดการข้อมูลเบื้องต้นใช้ ACE รุ่นที่ 1.1.3
- ไวยากรณ์ของภาษาจาวาที่นำมาใช้ในการศึกษาเป็นภาษาจาวา รุ่นที่ 1.6
- คลังสำหรับการสร้างต้นไม้ใช้ jsclass รุ่นที่ 4.0.3
- เครื่องมือที่ใช้ในการทดสอบ คือ ซิลิเนียมเว็บไดร์เวอร์ รุ่นที่ 2.44.0

บทที่ 4
ผลการวิจัยและอภิปรายผล

ในบทนี้เป็นการนำเสนอผลการวิจัยและการอภิปรายผล โดยในหัวข้อที่ 4.1 เป็นการนำเสนอการพัฒนาตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์ ต้นไม้สำหรับนำไปใช้ในระบบต้นแบบ ซึ่งประกอบด้วยต้นไม้วากยสัมพันธ์เชิงนามธรรม ต้นไม้การพึ่งพาและสถานะล็อค และการพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน หัวข้อที่ 4.2 การทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา และหัวข้อที่ 4.3 เป็นการนำเสนอการประเมินความถูกต้องของระบบต้นแบบที่ได้ทำการพัฒนา โดยมีรายละเอียดของแต่ละหัวข้อดังนี้

4.1 การพัฒนาตัวแจงส่วนภาษาจาวา การสร้างต้นไม้ที่ใช้ในระบบต้นแบบ และการพัฒนาระบบต้นแบบ

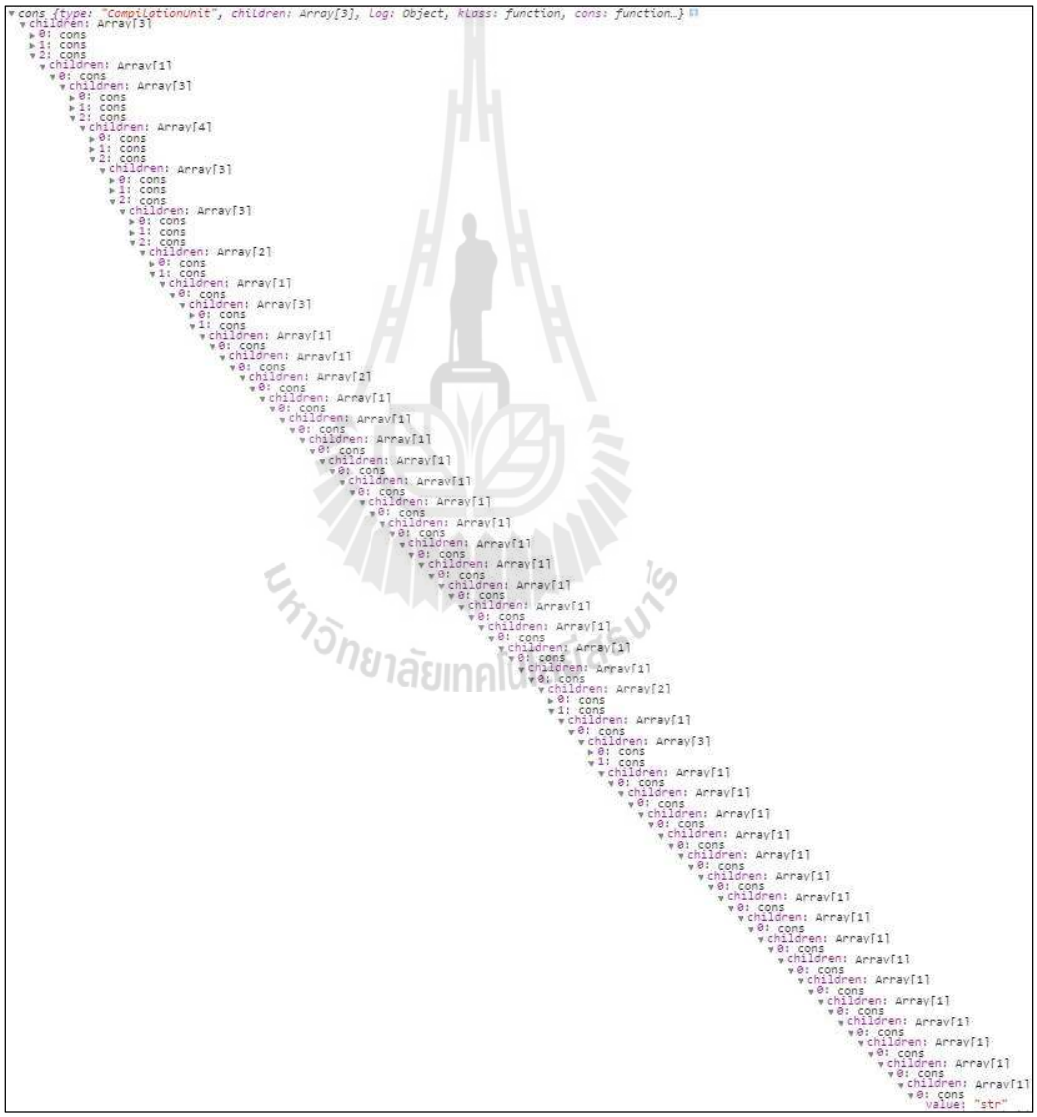
ในการออกแบบและการพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกันบนเว็บเบราว์เซอร์ ผู้วิจัยได้นำเสนอรายละเอียดในส่วนของการสร้างตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์ การสร้างต้นไม้สำหรับนำไปใช้ในระบบต้นแบบ ซึ่งประกอบด้วยต้นไม้วากยสัมพันธ์เชิงนามธรรม ต้นไม้การพึ่งพาและสถานะล็อค และการพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน โดยมีรายละเอียดในการสร้างและพัฒนาดังต่อไปนี้

4.1.1 ตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์

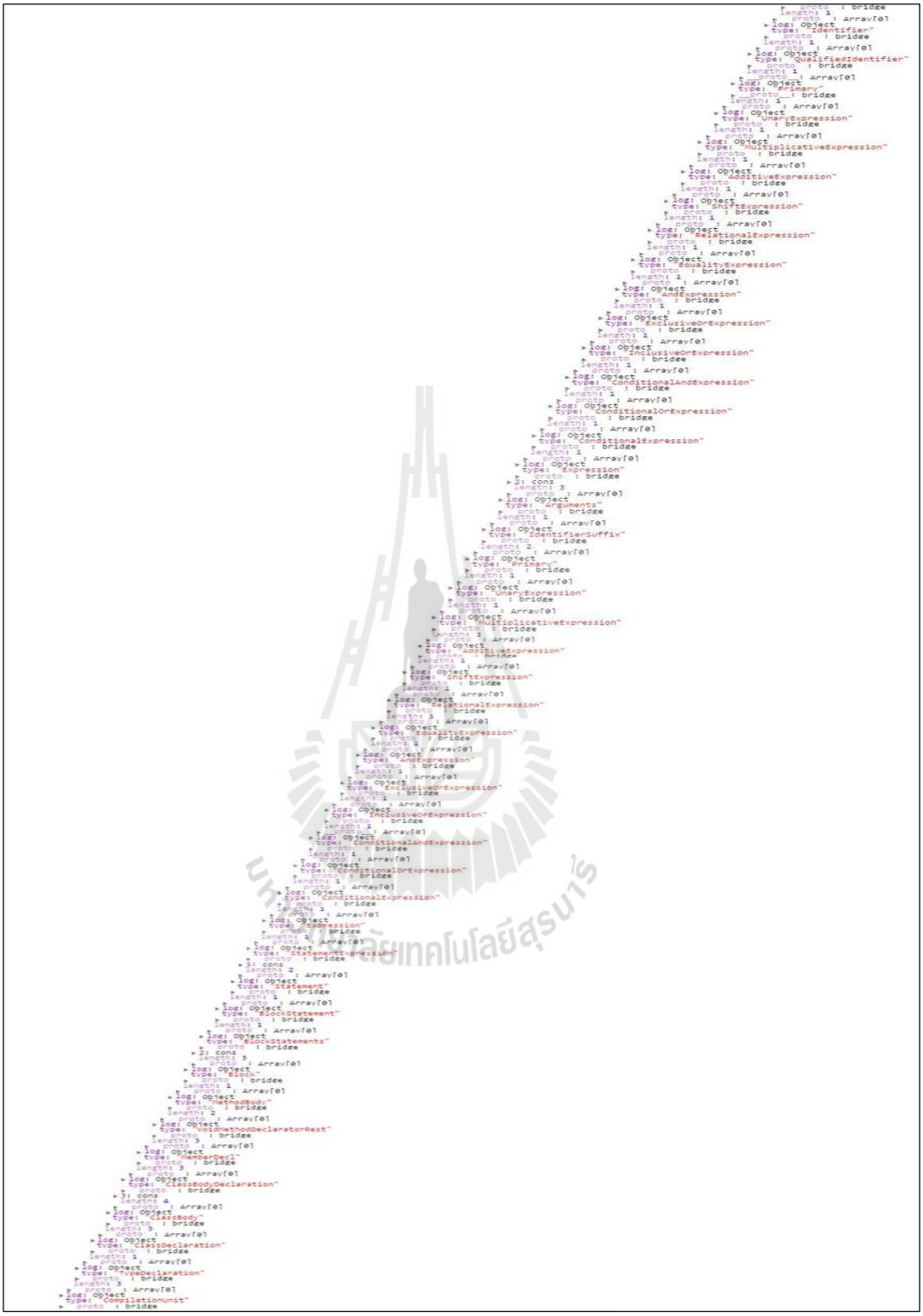
ในการสร้างตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์ โดยใช้เครื่องมือสร้างตัวแจงส่วน PEG.js ซึ่งแบ่งการสร้างออกเป็นสองส่วนด้วยกันคือ ส่วนตัววิเคราะห์คำ และส่วนการแจงส่วนของภาษา จากการสร้างสองส่วนนี้จะทำให้ได้ผลลัพธ์ออกมาเป็นต้นไม้วากยสัมพันธ์เชิงนามธรรมของภาษาจาวา โดยเมื่อเราทำการนำเข้ารหัสต้นฉบับภาษาจาวาดังแสดงในรูปที่ 4.1 ซึ่งเป็นรหัสต้นฉบับที่มีการแสดงผลลัพธ์ออกมาเป็นคำว่า “Hello world” และเมื่อนำไปผ่านตัวแจงส่วนภาษาจาวาที่ได้ทำการพัฒนาขึ้นมาแล้วจะได้ผลลัพธ์ที่มีลักษณะดังแสดงในรูปที่ 4.2 และรูปที่ 4.3 เป็นรูปที่แสดงโหนดที่ลึกที่สุดของผลลัพธ์ที่ได้จากการแจงส่วนนั้นคือตัวแปร str โดยอยู่ในโหนดระดับที่ 45

```
package th.ac.sut

class AST {
    String str = "Hello world";
    public static void main(String[] args) {
        System.out.println(str);
    }
}
```

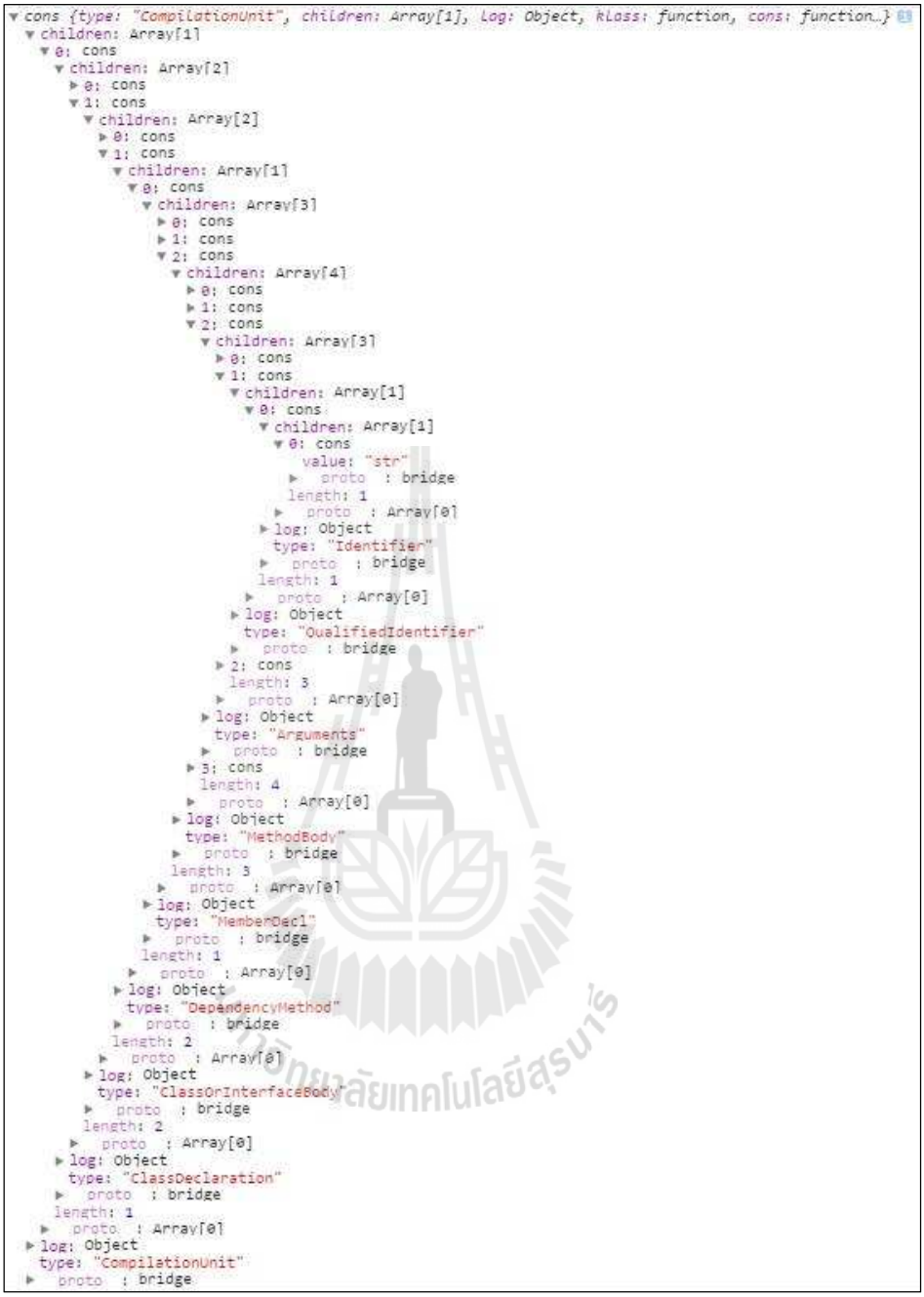
รูปที่ 4.1 รหัสต้นฉบับภาษาจาวาที่แสดงผลลัพธ์คำว่า “Hello world”



รูปที่ 4.2 ต้นไม้เชิงนามธรรมที่ได้จากการแจงส่วนรหัสต้นฉบับภาษาจาวาที่แสดงผลลัพธ์คำว่า “Hello world” ส่วนที่ 1



รูปที่ 4.3 ต้นไม้เชิงนามธรรมที่ได้จากการแจงส่วนรหัสต้นฉบับภาษาจาวาที่แสดงผลลัพธ์คำว่า “Hello world” ส่วนที่ 2



รูปที่ 4.4 ต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ได้ทำการปรับปรุง

4.1.2 ต้นไม้สำหรับนำไปใช้ในระบบต้นแบบ

ต้นไม้ที่มีการสร้างสำหรับนำไปใช้ในระบบต้นแบบประกอบด้วยต้นไม้วากยสัมพันธ์เชิงนามธรรมที่มีเฉพาะโหนดที่ต้องการนำไปใช้งาน และต้นไม้การพึ่งพาและสถานะล็อค โดยมีรายละเอียดแต่ละส่วนดังต่อไปนี้

1. ต้นไม้วากยสัมพันธ์เชิงนามธรรม

สำหรับต้นไม้วากยสัมพันธ์เชิงนามธรรมในส่วนนี้ เป็นการจัดการต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ผ่านตัวแจงส่วนดังแสดงในรูปที่ 4.2 และรูปที่ 4.3 ให้เหลือเฉพาะโหนดที่ต้องการนำไปใช้งานต่อเท่านั้น ผลลัพธ์ที่ได้จากการปรับปรุงจะออกมาเป็นต้นไม้วากยสัมพันธ์เชิงนามธรรมดังแสดงในรูปที่ 4.4 ซึ่งเป็นรูปที่แสดงโหนดที่ลึกที่สุดของผลลัพธ์ที่ได้จากการแจงส่วน ซึ่งก็คือตัวแปร str ที่อยู่ในโหนดระดับที่ 9 ซึ่งลดลงจากเดิมที่อยู่ในโหนดระดับที่ 45 โดยนอกจากจะลดระดับของโหนดลงได้แล้ว ยังสามารถช่วยลดระยะเวลาในการนำไปใช้สร้างต้นไม้การพึ่งพาและสถานะล็อคที่อยู่ในขั้นตอนถัดไปอีกด้วย การปรับปรุงต้นไม้วากยสัมพันธ์ดังกล่าวใช้หลักการของแพทเทิร์นเยี่ยมเยือน (Visitor pattern) ซึ่งตัวอย่างรหัสต้นฉบับที่ใช้ในการจัดการเพื่อให้ได้ต้นไม้วากยสัมพันธ์เชิงนามธรรมเฉพาะโหนดที่ต้องการนำไปใช้ดังแสดงในรูปที่ 4.5

```
var visitor = {

/* Source Code… */

ClassDeclaration: function(node) {
        var classDec = [];
        classDec.push(visit(this, node.children[1]));
        for(var n in node.children) {
            if(node.children[n].value == 'extends') {
                classDec.push(new nodes.Terminal(node.children[n].value));
                classDec.push(visit(this, node.children[++n]));
            }
            if(node.children[n].value == 'implements') {
                classDec.push(new nodes.Terminal(node.children[n].value));
                classDec.push(visit(this, node.children[++n]));
            }
        }
        classDec.push(visit(this, node.children[node.children.length-1]));
        return new nodes.ASTNode('ClassDeclaration', pr(classDec), node.log);
    },

/* Source Code… */

};

function visit(visitor, node) {
    return visitor[node.type].call(visitor, node);
}
```

รูปที่ 4.5 ตัวอย่างรหัสต้นฉบับที่ใช้ในการจัดการเพื่อให้ได้ต้นไม้วากยสัมพันธ์เชิงนามธรรมเฉพาะโหนดที่ต้องการ

```
// ฟังก์ชันสำหรับการสร้างต้นไม้พึ่งพาและสถานะล็อค
function createReferences(node) {
    /* Source Code */
    return ret;
}
```

รูปที่ 4.6 ฟังก์ชันสำหรับการสร้างต้นไม้การพึ่งพาและสถานะล็อค

2. ต้นไม้การพึ่งพาและสถานะล็อค

สำหรับขั้นตอนการสร้างต้นไม้การพึ่งพาและสถานะล็อค จะถูกสร้างขึ้นจากต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ได้ทำการปรับปรุงให้เหลือเฉพาะโหนดที่ต้องการ โดยการนำต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ได้ทำการปรับปรุงไปผ่านฟังก์ชัน createReferences(node) ดังแสดงในรูปที่ 4.6 ซึ่งฟังก์ชัน createReferences(node) จะส่งตัวแปรไปให้กับฟังก์ชันดังแสดงในรูปที่ 4.7 ตามพื้นที่การทำงานของตัวแปร ยกตัวอย่างเช่น หากตัวแปรเป็นตัวแปรที่อยู่ในอาร์กิวเมนต์ จะถูกส่งไปยังฟังก์ชัน manageArgument(node, len) เพื่อเก็บตัวแปรไว้ในอะเรย์ (Array) โดยแยกตาม

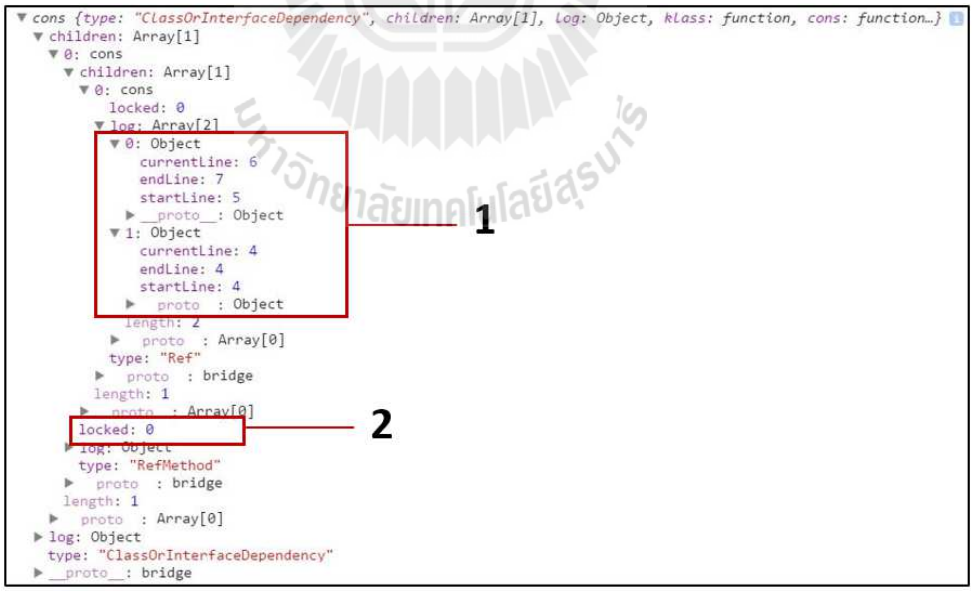
```
// ฟังก์ชันสำหรับค้นหาตัวแปรที่ถูกประกาศจากคลาสที่ถูกสืบทอดหรืออิมพลิเมนต์
function findExtendsAndImplement(walked, node) {
    /* Source Code */
}

// ฟังก์ชันสำหรับค้นหาตัวแปรที่ถูกประกาศจากอิมพลิเมนต์ที่ถูกสืบทอด
function findImplement(walked, node) {
    /* Source Code */
}

// ฟังก์ชันสำหรับเก็บการเรียกใช้ตัวแปรหรือเมธอดที่มีการประกาศค่าไว้ก่อนหน้าแล้ว
function walkSuperOrThis(node, position) {
    /* Source Code */
}

// ฟังก์ชันสำหรับเก็บการเรียกใช้ตัวแปรหรือเมธอดที่มีการประกาศค่าไว้ในคลาสอื่น
function findOtherClassDependency(child, log) {
    /* Source Code */
}

// ฟังก์ชันสำหรับการเรียกใช้ตัวแปรหรือเมธอดที่อยู่ภายในอาร์กิวเมนต์
function manageArgument(node, len) {
    /* Source Code */
    return len;
}
```

รูปที่ 4.7 ฟังก์ชันสำหรับการเก็บตัวแปรแยกตามประเภทของการเรียกใช้

ประเภทของการเรียกใช้ และส่งต่อไปยังฟังก์ชันในรูปที่ 4.8 เพื่อตรวจสอบระหว่างตัวแปรหรือเมธอดที่เป็นการประกาศและตัวแปรหรือเมธอดที่เป็นการเรียกใช้ หลังจากนั้นจะสร้างโหนดการพึ่งพาและสถานะล็อคขึ้นและส่งค่ากลับไปยังฟังก์ชัน createReferences(node) เพื่อทำรวมเข้ากับโหนดการพึ่งพาและสถานะล็อคที่มีอยู่ก่อนหน้า

```
// ฟังก์ชันสำหรับการตรวจสอบตัวแปรระหว่างตัวแปรประกาศและตัวแปรที่ถูกเรียกใช้
function manageFormat(rangeOfMethodOrigin) {
   /* Source Code */
    return ret;
}

// ฟังก์ชันสำหรับการตรวจสอบเมธอดระหว่างเมธอดประกาศและเมธอดที่ถูกเรียกใช้
function levelMethod(rangeOfMethodOrigin) {
    /* Source Code */
    return ret;
}
```

รูปที่ 4.8 ฟังก์ชันสำหรับการตรวจสอบระหว่างการประกาศและการเรียกใช้



รูปที่ 4.9 ต้นไม้การพึ่งพาและสถานะล็อค หมายเลข 1 คือ ส่วนของรหัสต้นฉบับที่มีการพึ่งพากัน และหมายเลข 2 คือ ส่วนที่ใช้สำหรับเก็บไอดีของไคลเอนต์ที่เป็นเจ้าของพื้นที่การทำงาน

จากขั้นตอนดังที่ได้กล่าวมาข้างต้น โดยใช้รหัสต้นฉบับดังแสดงในรูปที่ 4.1 จะทำให้ได้ต้นไม้การพึ่งพาและสถานะล็อคดังแสดงในรูปที่ 4.9 ซึ่งการพึ่งพากันจะแสดงให้เห็นในกรอบสี่เหลี่ยมหมายเลข 1 โดย

- **อ๊อบเจกต์ที่ 0** : จะเป็นการเรียกใช้ตัวแปร คือ ตัวแปร str ในบรรทัดที่ 6 ของรหัสต้นฉบับ อยู่ในขอบเขตของเมธอด main() ในบรรทัดที่ 5 - 7
- **อ๊อบเจกต์ที่ 1** : เป็นตัวแปรที่ถูกอ๊อบเจกต์ที่ 0 เรียกใช้ซึ่งอยู่ในบรรทัดที่ 4

กรอบสี่เหลี่ยมหมายเลข 2 เป็นส่วนที่ถูกพัฒนาให้เป็นที่เก็บไอดีของไคลเอนต์ที่เป็นเจ้าของพื้นที่การทำงาน หากไม่มีไคลเอนต์ใดเป็นเจ้าของพื้นที่การทำงานดังกล่าวแล้วจะถูกแสดงด้วยตัวเลข 0 ดังภาพ

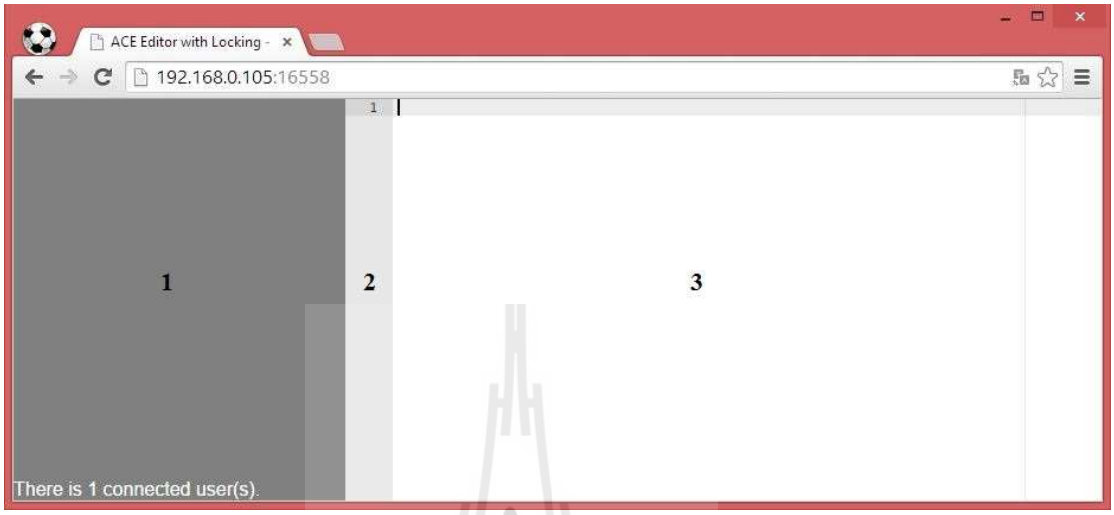
4.1.3 ระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน

การพัฒนาตัวแจงส่วนภาษาจาวาบนเว็บเบราว์เซอร์จะทำให้ได้ผลลัพธ์คือ ต้นไม้วากยสัมพันธ์เชิงนามธรรม ซึ่งจะนำต้นไม้ดังกล่าวมาทำการปรับปรุงให้เหลือเฉพาะส่วนที่จะนำไปใช้งานต่อเพื่อสร้างต้นไม้การพึ่งพาและสถานะล็อค หลังจากที่ได้ทำการสร้างต้นไม้การพึ่งพาและสถานะล็อคแล้ว จะนำตัวแจงส่วนดังกล่าวใส่ลงในซอฟต์แวร์สำหรับการจัดการข้อมูล เพื่อพัฒนาเป็นระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน โดยซอฟต์แวร์สำหรับการจัดการข้อมูลได้มีการพัฒนาให้สามารถเชื่อมต่อกันระหว่างเซิร์ฟเวอร์และไคลเอนต์ด้วยการใช้โมดูล socket.io รวมถึงมีการพัฒนาให้มีไคลเอนต์มากกว่า 1 ไคลเอนต์สามารถเข้าถึงพื้นที่การทำงานเดียวกันได้

ในขั้นตอนต่อไปหลังจากนำตัวแจงส่วนที่สามารถแจงส่วนรหัสต้นฉบับภาษาจาวาที่สามารถสร้างนำต้นไม้ที่เกี่ยวข้องใส่ลงในซอฟต์แวร์สำหรับการจัดการข้อมูลที่ได้ทำการพัฒนาต่อจาก ACE แล้ว ผู้วิจัยได้ทำการพัฒนาฟังก์ชัน dependencyServer(dependencyRefDemo, dependencyCurrentLine, sentTo) ซึ่งเป็นฟังก์ชันสำหรับกำหนดเจ้าของพื้นที่การทำงานให้กับต้นไม้การพึ่งพาและสถานะล็อค และพัฒนาฟังก์ชัน multipleUsers(dataServer, eolServer) ที่เป็นฟังก์ชันในการแสดงพื้นที่ที่ถูกล็อคการทำงานบนไคลเอนต์ของตัวเอง ตามลักษณะการทำงานที่ได้ทำการออกแบบไว้ในรูปที่ 3.12 และรูปที่ 3.13 ตามลำดับ

จากขั้นตอนที่ได้กล่าวมาก่อนจะได้ผลลัพธ์เป็นระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน ทั้งนี้ผู้วิจัยได้ทำการพัฒนาส่วนติดต่อกับผู้ใช้ตามที่ได้ออกแบบไว้ในรูปที่ 3.9 ทำให้ได้ผลลัพธ์ดังแสดงในรูปที่ 4.10 ซึ่งประกอบไปด้วย 3 ส่วนคือ ส่วนที่ 1 เป็นส่วนสำหรับบอกจำนวน ของนักพัฒนาที่เชื่อมต่อเข้าเซิร์ฟเวอร์ ส่วนที่ 2 เป็นส่วนของการบอกจำนวนบรรทัดของ

รหัสต้นฉบับ และส่วนที่ 3 เป็นส่วนของพื้นที่สำหรับการพัฒนารหัสต้นฉบับ

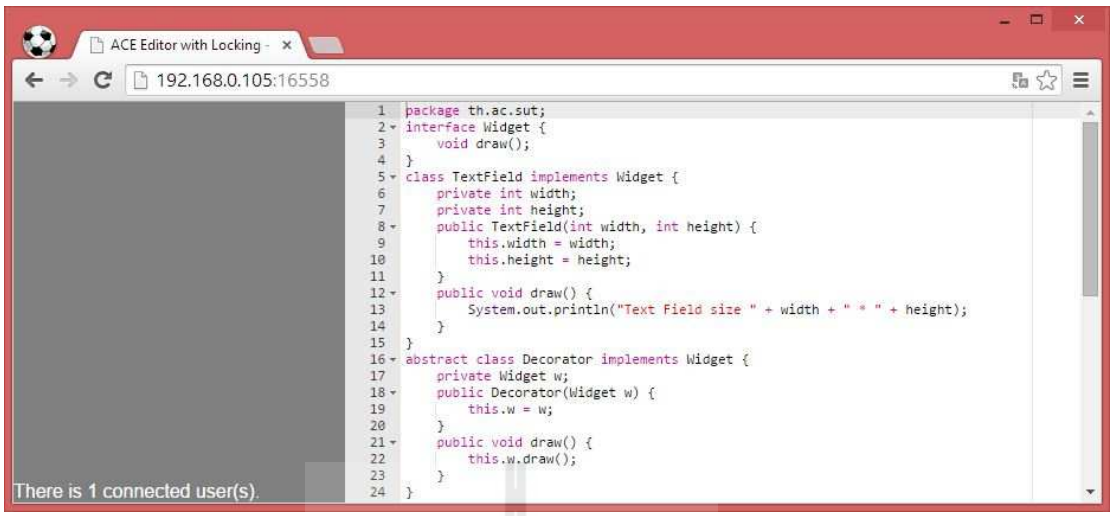


รูปที่ 4.10 ส่วนติดต่อกับผู้ใช้ที่ได้ทำการพัฒนา

สำหรับรหัสต้นฉบับที่นำมาใช้เพื่อสังเกตลักษณะการทำงานของระบบต้นแบบที่ได้ทำการพัฒนาขึ้น จะใช้รหัสต้นฉบับดังแสดงในรูปที่ 4.11 ที่เป็นรหัสต้นฉบับที่เป็นตัวอย่างการใช้งานแพทเทิร์นตัวตกแต่ง (Decorator pattern) เมื่อทำการพิมพ์รหัสต้นฉบับดังกล่าวลงในระบบต้นแบบแล้ว จะได้ผลลัพธ์ดังแสดงในรูปที่ 4.12

```
 1  package th.ac.sut;
 2  interface Widget {
 3      void draw();
 4  }
 5  class TextField implements Widget {
 6      private int width;
 7      private int height;
 8      public TextField(int width, int height) {
 9          this.width = width;
10          this.height = height;
11      }
12      public void draw() {
13          System.out.println("Text Field size " + width + " * " + height);
14      }
15  }
16  abstract class Decorator implements Widget {
17      private Widget w;
18      public Decorator(Widget w) {
19          this.w = w;
20      }
21      public void draw() {
22          this.w.draw();
23      }
24  }
25  class BorderDecorator extends Decorator {
26      public BorderDecorator(Widget w) {
27          super(w);
28      }
29      public void draw() {
30          super.draw();
31          System.out.println("  Border Decorator");
32      }
33  }
34  class ScrollDecorator extends Decorator {
35      public ScrollDecorator(Widget w) {
36          super(w);
37      }
38      public void draw() {
39          super.draw();
40          System.out.println("  Scroll Decorator");
41      }
42  }
43  public class Client {
44      public static void main(String[] args) {
45          Widget t = new TextField(60, 20);
46          Widget s = new ScrollDecorator(t);
47          Widget b = new BorderDecorator(s);
48          b.draw();
49      }
50  }
```

รูปที่ 4.11 รหัสต้นฉบับที่ทำการพัฒนาสำหรับการทดสอบลักษณะการล็อคพื้นที่การทำงาน

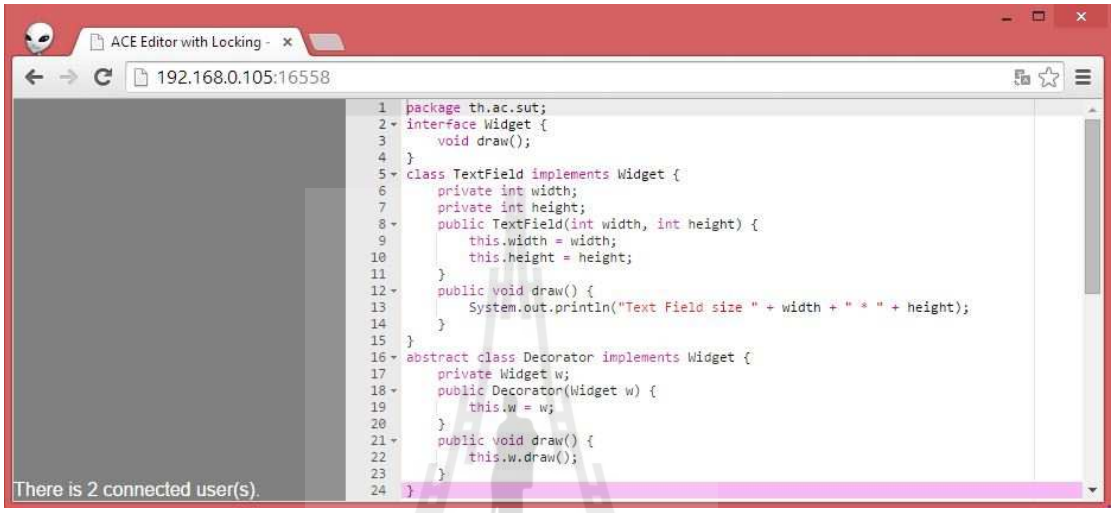


รูปที่ 4.12 นักพัฒนาที่ 1 พิมพ์รหัสต้นฉบับลงบนพื้นที่การทำงาน

จากรูปที่ 4.12 สังเกตด้านล่างซ้ายว่ามีนักพัฒนาเพียง 1 คนเท่านั้น ทำให้ยังไม่มีการล็อคพื้นที่การทำงานเกิดขึ้น แต่เมื่อมีนักพัฒนาคนที่ 2 ทำการเชื่อมต่อเข้าไปยังเซิร์ฟเวอร์แล้ว จะพบว่าด้านล่างซ้ายว่ามีนักพัฒนาเป็น 2 คน และมีการล็อคพื้นที่ในบรรทัดที่ 1 ดังแสดงในรูปที่ 4.13

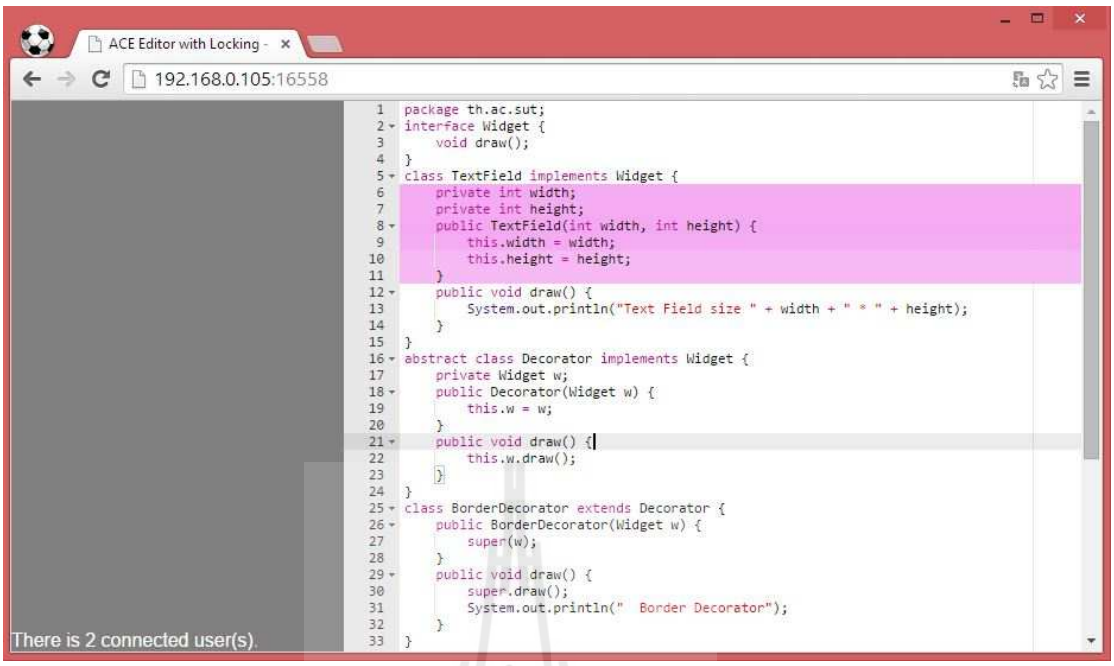
รูปที่ 4.13 นักพัฒนาที่ 2 เชื่อมต่อเข้าไปยังเซิร์ฟเวอร์

เนื่องจากเคอร์เซอร์ของนักพัฒนาที่ 2 อยู่ที่บรรทัดที่ 1 ในขณะเดียวกันเคอร์เซอร์ของนักพัฒนาคนที่ 1 อยู่ในบรรทัดที่ 24 ส่งผลให้บรรทัดที่ 24 บนพื้นที่การทำงานของนักพัฒนาที่ 2 ถูกล็อคด้วยเช่นกัน ดังแสดงในรูปที่ 4.14



รูปที่ 4.14 พื้นที่การทำงานของนักพัฒนาที่ 2 ถูกล็อคในบรรทัดที่ 24

เมื่อนักพัฒนาทั้งสองคนเริ่มมีการพัฒนารหัสต้นฉบับบนส่วนของรหัสต้นฉบับที่แตกต่างกัน โดยรูปที่ 4.15 เป็นพื้นที่การทำงานของนักพัฒนาที่ 1 จะพบว่ามีพื้นที่การทำงานที่ถูกล็อคอยู่ (พื้นที่การทำงานที่ถูกล็อคจะอยู่ภายในกรอบสี่เหลี่ยม) ในบรรทัดที่ 6 - 11 เนื่องมาจากมีนักพัฒนาที่ 2 กำลังทำงานอยู่ในพื้นที่การทำงานดังกล่าว ในขณะที่นักพัฒนาที่ 1 กำลังทำงานอยู่ในเมธอด draw() ของคลาสนามธรรม Decorator (บรรทัดที่ 21 - 23) จะส่งผลให้เมธอด draw() ของคลาสนามธรรม Decorator บนพื้นที่การทำงานของนักพัฒนาที่ 2 ถูกล็อค พร้อมทั้งล็อคส่วนที่พึ่งพากันซึ่งก็คือบรรทัดที่ 17 ดังแสดงในรูปที่ 4.16



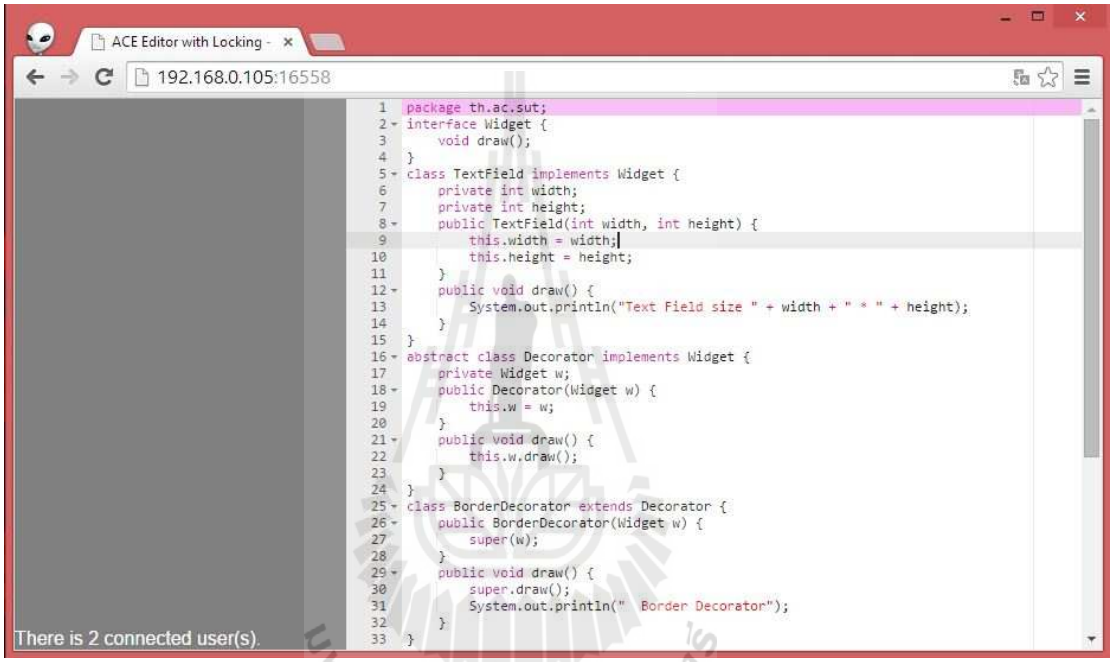

```
1  package th.ac.sut;
2  interface Widget {
3      void draw();
4  }
5  class TextField implements Widget {
6      private int width;
7      private int height;
8      public TextField(int width, int height) {
9          this.width = width;
10         this.height = height;
11     }
12     public void draw() {
13         System.out.println("Text Field size " + width + " * " + height);
14     }
15 }
16 abstract class Decorator implements Widget {
17     private Widget w;
18     public Decorator(Widget w) {
19         this.w = w;
20     }
21     public void draw() {
22         this.w.draw();
23     }
24 }
25 class BorderDecorator extends Decorator {
26     public BorderDecorator(Widget w) {
27         super(w);
28     }
29     public void draw() {
30         super.draw();
31         System.out.println("  Border Decorator");
32     }
33 }
```

รูปที่ 4.15 นักพัฒนาที่ 1 ทำงานบนเมธอด draw() ของคลาสนามธรรม Decorator


```
1  package th.ac.sut;
2  interface Widget {
3      void draw();
4  }
5  class TextField implements Widget {
6      private int width;
7      private int height;
8      public TextField(int width, int height) {
9          this.width = width;
10         this.height = height;
11     }
12     public void draw() {
13         System.out.println("Text Field size " + width + " * " + height);
14     }
15 }
16 abstract class Decorator implements Widget {
17     private Widget w;
18     public Decorator(Widget w) {
19         this.w = w;
20     }
21     public void draw() {
22         this.w.draw();
23     }
24 }
25 class BorderDecorator extends Decorator {
26     public BorderDecorator(Widget w) {
27         super(w);
28     }
29     public void draw() {
30         super.draw();
31         System.out.println("  Border Decorator");
32     }
33 }
```

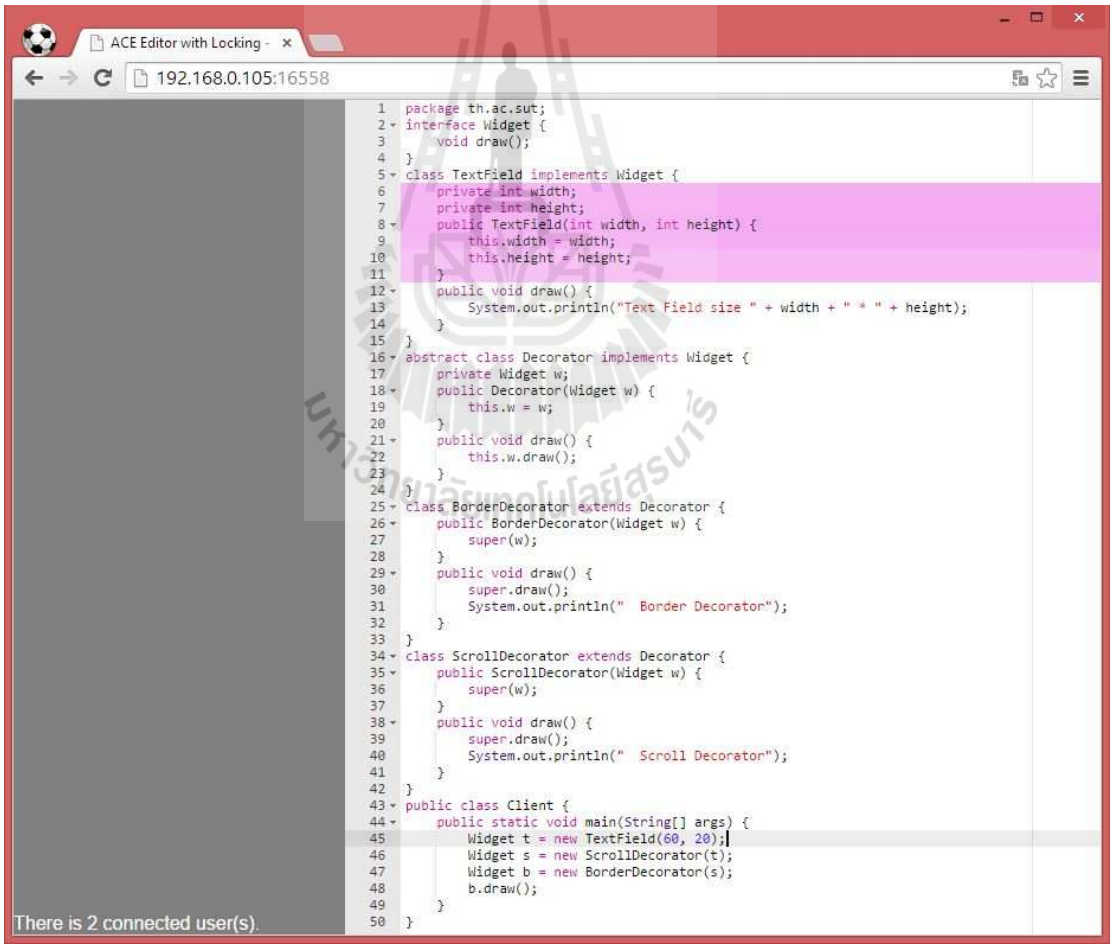
รูปที่ 4.16 นักพัฒนาที่ 2 ทำงานบนคอนสตรัคเตอร์ของคลาส TextField

เมื่อนักพัฒนาที่ 1 ออกจากการทำงานในส่วนของเมธอด draw() ของคลาสนามธรรม Decorator จะทำให้พื้นที่การทำงานดังกล่าวถูกปลดล็อคโดยอัตโนมัติ ดังแสดงในรูปที่ 4.17 ซึ่งเป็นพื้นที่การทำงานของนักพัฒนาคนที่ 2 จะเห็นว่าไม่มีการล็อคในส่วนของเมธอด draw() ของคลาสนามธรรม Decorator แล้ว แต่มีการล็อคในส่วนของบรรทัดที่ 1 เนื่องจากนักพัฒนาคนที่ 1 ย้ายเคอร์เซอร์ไปยังบรรทัดดังกล่าว



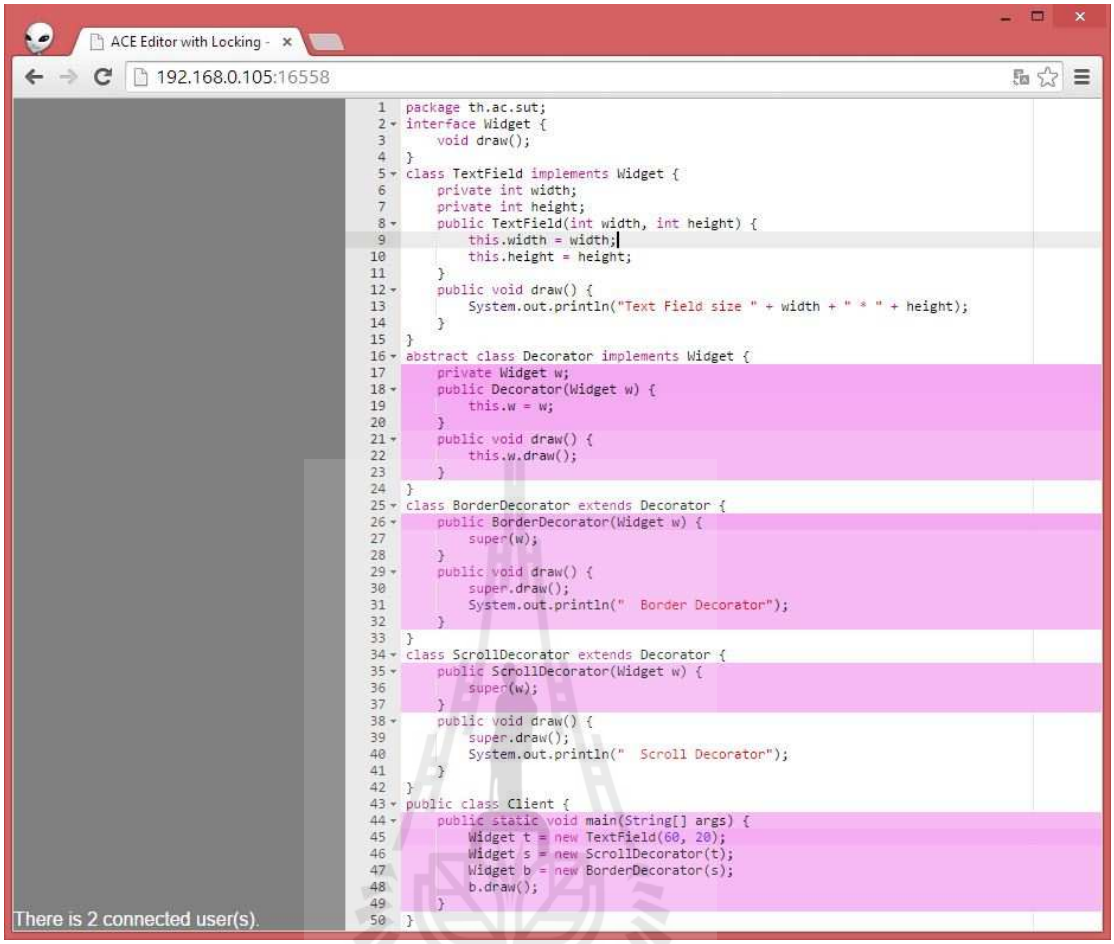
รูปที่ 4.17 พื้นที่การทำงานของนักพัฒนาที่ 2 ถูกปลดล็อค ในเมธอด draw() ของคลาสนามธรรม Decorator

ต่อมาเมื่อนักพัฒนาที่ 1 ย้ายพื้นที่การทำงานไปอยู่ในเมธอด main() ของคลาส Client ดังแสดงในรูปที่ 4.18 จะทำให้พื้นที่การทำงานของนักพัฒนาที่ 2 ถูกล็อคดังแสดงในรูปที่ 4.19 ซึ่งพื้นที่การทำงานที่ถูกล็อคประกอบไปด้วย 3 ส่วนที่พึ่งพากัน โดยส่วนแรกคือคลาส Decorator ในส่วนของการประกาศตัวแปร w คอนสตรัคเตอร์ Decorator และเมธอด draw() ส่วนที่สองคือคลาส BorderDecorator ในส่วนของคอนสตรัคเตอร์ BorderDecorator และเมธอด draw() และส่วนสุดท้ายคือ คอนสตรัคเตอร์ ScrollDecorator ของคลาส ScrollDecorator แต่ในส่วนของคลาส TextField นั้นไม่ถูกล็อคแม้ว่าจะเป็นส่วนที่พึ่งพากันอยู่กับเมธอด main() เนื่องมาจากพื้นที่การทำงานดังกล่าวนั้นผู้พัฒนาที่ 2 เป็นเจ้าของสิทธิ์อยู่ก่อนที่นักพัฒนาที่ 1 จะย้ายการทำงานไปยังเมธอด main()




```
1  package th.ac.sut;
2  interface Widget {
3      void draw();
4  }
5  class TextField implements Widget {
6      private int width;
7      private int height;
8      public TextField(int width, int height) {
9          this.width = width;
10         this.height = height;
11     }
12     public void draw() {
13         System.out.println("Text Field size " + width + " * " + height);
14     }
15 }
16 abstract class Decorator implements Widget {
17     private Widget w;
18     public Decorator(Widget w) {
19         this.w = w;
20     }
21     public void draw() {
22         this.w.draw();
23     }
24 }
25 class BorderDecorator extends Decorator {
26     public BorderDecorator(Widget w) {
27         super(w);
28     }
29     public void draw() {
30         super.draw();
31         System.out.println("  Border Decorator");
32     }
33 }
34 class ScrollDecorator extends Decorator {
35     public ScrollDecorator(Widget w) {
36         super(w);
37     }
38     public void draw() {
39         super.draw();
40         System.out.println("  Scroll Decorator");
41     }
42 }
43 public class Client {
44     public static void main(String[] args) {
45         Widget t = new TextField(60, 20);
46         Widget s = new ScrollDecorator(t);
47         Widget b = new BorderDecorator(s);
48         b.draw();
49     }
50 }
```

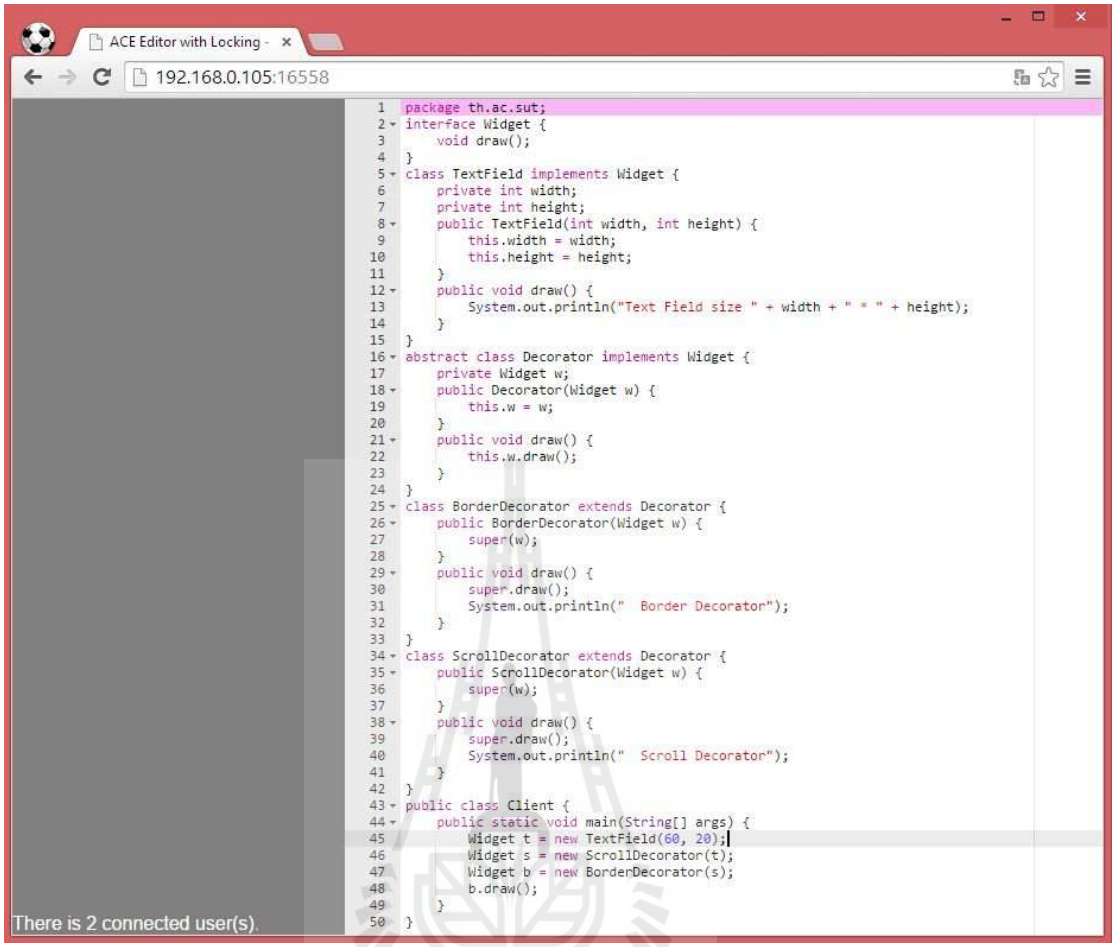
รูปที่ 4.18 นักพัฒนาที่ 1 ทำงานในเมธอด main() ของคลาส Client



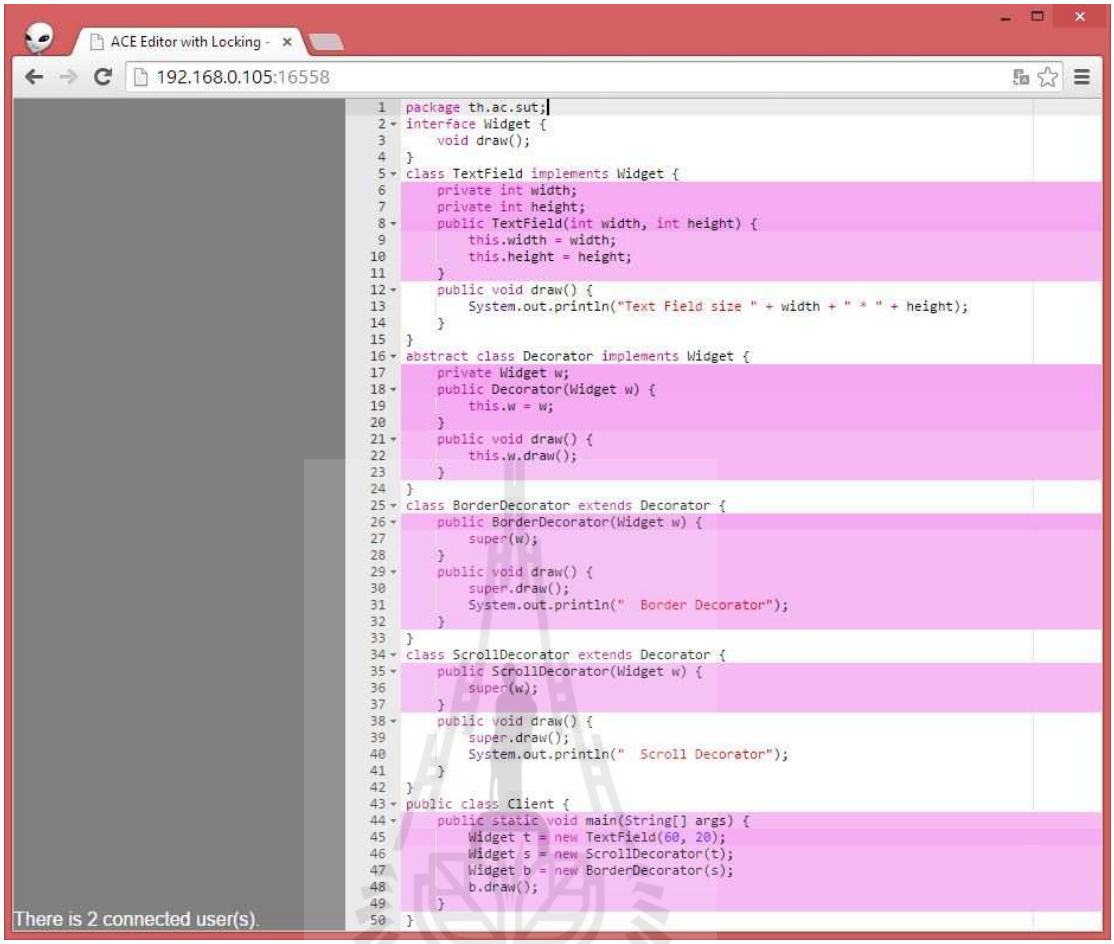

```
1  package th.ac.sut;
2  interface Widget {
3      void draw();
4  }
5  class TextField implements Widget {
6      private int width;
7      private int height;
8      public TextField(int width, int height) {
9          this.width = width;
10         this.height = height;
11     }
12     public void draw() {
13         System.out.println("Text Field size " + width + " * " + height);
14     }
15 }
16 abstract class Decorator implements Widget {
17     private Widget w;
18     public Decorator(Widget w) {
19         this.w = w;
20     }
21     public void draw() {
22         this.w.draw();
23     }
24 }
25 class BorderDecorator extends Decorator {
26     public BorderDecorator(Widget w) {
27         super(w);
28     }
29     public void draw() {
30         super.draw();
31         System.out.println("  Border Decorator");
32     }
33 }
34 class ScrollDecorator extends Decorator {
35     public ScrollDecorator(Widget w) {
36         super(w);
37     }
38     public void draw() {
39         super.draw();
40         System.out.println("  Scroll Decorator");
41     }
42 }
43 public class Client {
44     public static void main(String[] args) {
45         Widget t = new TextField(60, 20);
46         Widget s = new ScrollDecorator(t);
47         Widget b = new BorderDecorator(s);
48         b.draw();
49     }
50 }
```

รูปที่ 4.19 พื้นที่การทำงานของนักพัฒนาที่ 2 ถูกล็อคจากการที่นักพัฒนาที่ 1 ทำงานในเมธอด main() ของคลาส Client

เมื่อนักพัฒนาที่ 2 ได้ทำการย้ายเคอร์เซอร์ไปทำงานยังบรรทัดที่ 1 แล้วพื้นที่การทำงานเดิมซึ่งก็คือคอนสตรัคเตอร์ TextField และส่วนที่พึ่งพากันบนพื้นที่การทำงานของนักพัฒนาที่ 1 จะถูกปลดล็อคโดยอัตโนมัติ ดังแสดงในรูปที่ 4.20 ในขณะเดียวกันพื้นที่การทำงานของนักพัฒนาที่ 2 จะถูกล็อคเพิ่มในส่วนคอนสตรัคเตอร์ TextField และส่วนที่พึ่งพากันโดยอัตโนมัติดังแสดงในรูปที่ 4.21



รูปที่ 4.20 พื้นที่การทำงานของนักพัฒนาที่ 1 เมื่อคอนสตรัคเตอร์ TextField ถูกปลดล็อค



รูปที่ 4.21 พื้นที่การทำงานของนักพัฒนาที่ 2 ถูกล็อคเพิ่มในส่วนของคอนสตรัคเตอร์ TextField และส่วนที่พึ่งพากัน

4.2 การทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา

ในการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์จะทดสอบบนเครื่องคอมพิวเตอร์ส่วนบุคคล โดยมีรายละเอียดดังนี้

- หน่วยประมวลผลกลาง : Intel® Core™ i5-2430M CPU ความถี่สัญญาณนาฬิกา/ความเร็ว 2.40 GHz
- หน่วยความจำสำรอง : 500 GB
- หน่วยความจำหลัก : 8.00 GB
- ระบบปฏิบัติการ : Windows 7 Ultimate 64-bit Operating System

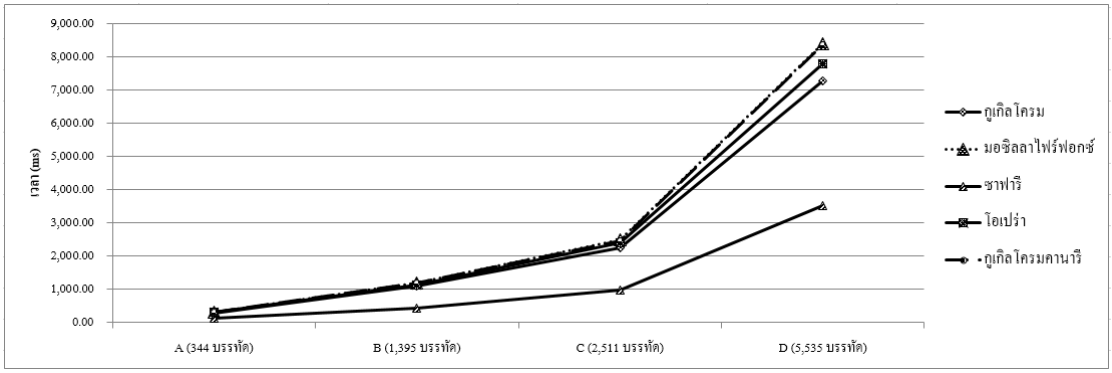
สำหรับการทดสอบการแจงส่วนข้อมูลประโยคภาษาจาวา จะทำโดยการนำแต่ละไฟล์ที่มาทำทดสอบไฟล์ละ 10 รอบ และทำการบันทึกเวลาที่ใช้ในการทดสอบในแต่ละรอบ เมื่อครบทั้ง 10 รอบ แล้วจะเรียงเวลาที่ได้จากการทดสอบจากน้อยไปมาก และทำการตัดเวลาที่น้อยที่สุด 2 อันดับ และมากที่สุด 2 อันดับออก จากนั้นนำเวลาของการทดสอบ 6 อันดับที่เหลือมาทำการหาค่าเฉลี่ย

ในการทดสอบเวลาที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวา ค่าเฉลี่ยของเวลาที่ใช้ในการแจงส่วนที่ได้มาจากการทดสอบดังแสดงในตารางที่ 4.1 โดยมีการกำหนดให้

- **A** คือไฟล์ h2-command-Command.java ที่จำนวนบรรทัดทั้งหมด 344 บรรทัด
- **B** คือไฟล์ h2-engine-Session.java ที่จำนวนบรรทัดทั้งหมด 1,395 บรรทัด
- **C** คือไฟล์ h2-engine-Database.java ที่จำนวนบรรทัดทั้งหมด 2,511 บรรทัด
- **D** คือไฟล์ h2-command-Parser.java ที่จำนวนบรรทัดทั้งหมด 5,535 บรรทัด

ตารางที่ 4.1 ค่าเฉลี่ยของหน่วยความจำสูงสุดที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วน

เว็บเบราว์เซอร์	**เวลา (มิลลิวินาที)**			
	A	**B**	**C**	**D**
กูเกิลโครม	272.83	1,074.00	2,238.83	7,274.83
มอซิลลาไฟร์ฟอกซ์	286.67	1,182.67	2,478.33	8,386.33
ซาฟารี	129.17	435.33	952.17	3,522.67
โอเปร่า	296.50	1,131.33	2,376.00	7,779.50
กูเกิลโครมคานารี	308.33	1,184.00	2,465.33	8,352.00



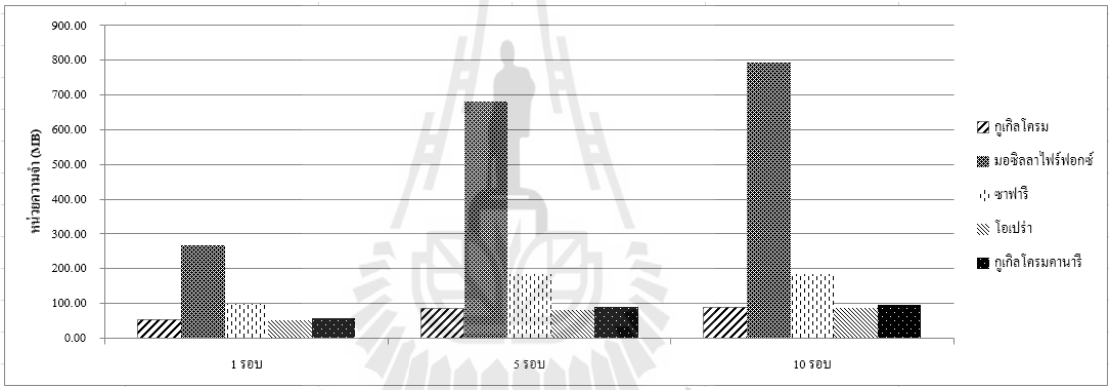
รูปที่ 4.22 กราฟแสดงค่าเฉลี่ยของเวลาที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วน

จากตารางที่ 4.1 จะสังเกตว่าค่าเฉลี่ยของเวลาในการแจงส่วนไฟล์ที่มีจำนวนของบรรทัดเป็น 344 บรรทัด 1,395 บรรทัด 2,511 บรรทัด และ 5,535 บรรทัด ซาฟารีใช้เวลาในการแจงส่วนน้อยที่สุดคือ 129.17 มิลลิวินาที 435.33 มิลลิวินาที 952.17 มิลลิวินาที และ 3,522.67 มิลลิวินาที ตามลำดับ โดยที่เว็บเบราว์เซอร์อื่นที่ทำการทดสอบมีค่าเฉลี่ยของเวลาที่ใช้ในการแจงส่วนมากกว่าซาฟารีและมีค่าที่ใกล้เคียงกัน โดยค่าเฉลี่ยของเวลาของเว็บเบราว์เซอร์อื่นที่ทำการทดสอบจะอยู่ที่ประมาณ 296.5 มิลลิวินาที 1,131.33 มิลลิวินาที 2,376 มิลลิวินาที และ 7,779.5 มิลลิวินาที ตามลำดับ เมื่อสร้างเป็นกราฟค่าเฉลี่ยของเวลาที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วนดังแสดงในรูปที่ 4.22 จะเห็นว่าซาฟารีมีแนวโน้มในการใช้เวลาสำหรับการแจงส่วนที่น้อยกว่าเว็บเบราว์เซอร์อื่นที่ทำการทดสอบ และจะเห็นได้ว่าเวลาที่ใช้ในการแจงส่วนของกูเกิลโครม กูเกิลโครมคานารี โอเปร่า และมอซิลลาไฟร์ฟอกซ์มีค่าที่ใกล้เคียงกัน

ส่วนการทดสอบเพื่อหาค่าเฉลี่ยของหน่วยความจำสูงสุดที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวา จะทำการทดสอบโดยใช้ไฟล์ h2-command-Parser.java ที่มีจำนวนบรรทัดทั้งหมด 5,535 บรรทัด ในการทดสอบด้วยจำนวนรอบที่แตกต่างกัน คือ การทดสอบจำนวน 1 รอบ 5 รอบ และ 10 รอบตามลำดับ จากตารางที่ 4.2 จะเห็นว่าเมื่อแจงส่วนไฟล์ที่มีจำนวนบรรทัดมากที่สุดในโปรเจค H2 ติดต่อกัน 1 รอบ 5 รอบและ 10 รอบตามลำดับ แล้วมอซิลลาไฟร์ฟอกซ์ใช้หน่วยความจำในการแจงส่วนมากกว่าเว็บเบราว์เซอร์อื่นที่ทำการทดสอบคือใช้หน่วยความจำสูงสุดเท่ากับ 265.55 เมกะไบต์ 681.36 เมกะไบต์ และ 794.24 เมกะไบต์ตามลำดับ ซาฟารีใช้หน่วยความจำสูงสุดรองลองมาคือ 98.78 เมกะไบต์ 184.38 เมกะไบต์ และ 184.48 เมกะไบต์ตามลำดับ ส่วนเว็บเบราว์เซอร์ที่เหลือที่ทดสอบคือ กูเกิลโครม กูเกิลโครมคานารี และโอเปร่าใช้หน่วยความจำสูงสุดใกล้เคียงกันคือประมาณ 51.79 เมกะไบต์ 86.12 เมกะไบต์ และ 89.33 เมกะไบต์ตามลำดับ และเมื่อนำค่าเฉลี่ยที่ได้มาสร้างเป็นกราฟดังแสดงในรูปที่ 4.23 จะเห็นได้อย่างชัดเจนว่ามอซิลลาไฟร์ฟอกซ์ใช้หน่วยความจำมากที่สุดในการแจงส่วนข้อมูลประโยค รองลงมาคือ ซาฟารี และเว็บเบราว์เซอร์ที่ใช้หน่วยความจำน้อยที่สุดคือ กูเกิลโครม กูเกิลโครมคานารี และโอเปร่า ซึ่งใช้หน่วยความจำสูงสุดในการแจงส่วนที่ใกล้เคียงกัน

ตารางที่ 4.2 ค่าเฉลี่ยของหน่วยความจำสูงสุดที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วน

เว็บเบราว์เซอร์	หน่วยความจำ (เมกะไบต์)		
	1 รอบ	**5 รอบ**	**10 รอบ**
กูเกิลโครม	51.79	86.12	89.33
มอซิลลาไฟร์ฟอกซ์	266.55	681.36	794.24
ซาฟารี	98.78	184.38	184.48
โอเปร่า	51.79	81.47	87.81
กูเกิลโครมคานารี	56.11	89.59	94.38



รูปที่ 4.23 กราฟแสดงค่าเฉลี่ยของหน่วยความจำสูงสุดที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วน

4.3 การประเมินความถูกต้องของระบบต้นแบบ

ในการประเมินความถูกต้องของระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน ได้ใช้เครื่องคอมพิวเตอร์ส่วนบุคคลจำนวน 2 เครื่องในการทดสอบ โดยกำหนดให้เครื่องคอมพิวเตอร์ส่วนบุคคลเครื่องที่ 1 ทำหน้าที่เป็นเครื่องเซิร์ฟเวอร์ และเป็นไคลเอนต์สำหรับนักพัฒนาที่ 1 โดยมีรายละเอียดของเครื่องคอมพิวเตอร์ดังนี้

- หน่วยประมวลผลกลาง : Intel® Core™ i5-3230M CPU ความถี่สัญญาณนาฬิกา/ความเร็ว 2.60 GHz
- หน่วยความจำสำรอง : 500 GB
- หน่วยความจำหลัก : 4.00 GB

- ระบบปฏิบัติการ : Windows 8 Pro 64-bit Operating System

และเครื่องคอมพิวเตอร์ส่วนบุคคลเครื่องที่ 2 ทำหน้าที่เป็นเครื่องไคลเอนต์สำหรับนักพัฒนาที่ 2 โดยมีรายละเอียดของเครื่องคอมพิวเตอร์ดังนี้

- หน่วยประมวลผลกลาง : Intel® Core™ i3-3120M CPU ความถี่สัญญาณนาฬิกา/ความเร็ว 2.50 GHz
- หน่วยความจำสำรอง : 500 GB
- หน่วยความจำหลัก : 4.00 GB
- ระบบปฏิบัติการ : Windows 8 Pro 64-bit Operating System

ในการประเมินความถูกต้องของระบบต้นแบบ จะทำโดยการใช้กรณีทดสอบทั้งหมด 11 กรณีที่ได้ทำการออกแบบไว้ในบทที่ 3 หัวข้อที่ 3.3 เรื่องการออกแบบการประเมินความถูกต้องของระบบต้นแบบ โดยในการทดสอบจะมีนักพัฒนาจำนวน 2 คน พัฒนารหัสต้นฉบับร่วมกัน ทั้งนี้ในการทำงานของนักพัฒนาที่ 1 จะเป็นการเลือกตำแหน่งการทำงานด้วยตัวของผู้วิจัยเอง โดยการเลือกการทำงานดูทำการเลือกจากกรณีทดสอบที่ได้ทำการออกแบบไว้ และสำหรับนักพัฒนาที่ 2 จะเป็นการจำลองการทำงานของซิลิเนียมเว็บไดร์เวอร์ตามกรณีทดสอบ โดยมีเงื่อนไขในการทดสอบตามกรณีต่าง ๆ ทั้ง 11 กรณี ที่ได้ทำการออกแบบไว้คือ กำหนดให้นักพัฒนาที่ 1 เป็นเจ้าของพื้นที่การทำงานก่อนเสมอ

ตารางที่ 4.3 ผลการทดสอบระบบต้นแบบตามกรณีทดสอบทั้ง 11 กรณี

กรณีทดสอบที่	**การทดสอบครั้งที่**										**เปอร์เซนต์ความถูกต้อง**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00
2	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	90.00
3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00
4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00
5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00
6	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00
7	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00
8	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00

ตารางที่ 4.3 ผลการทดสอบระบบต้นแบบตามกรณีทดสอบทั้ง 11 กรณี (ต่อ)

กรณีทดสอบที่	การทดสอบครั้งที่										เปอร์เซนต์ความถูกต้อง
	1	2	3	4	5	6	7	8	9	10	
9	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00
10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	100.00
11	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	90.00
			เปอร์เซนต์ความถูกต้องรวมของทุกกรณีทดสอบ								98.18

จากตารางที่ 4.3 เป็นผลที่ได้จากการทดสอบระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกันตามกรณีทดสอบที่ได้ทำการออกแบบไว้ทั้ง 11 กรณี พบว่ากรณีทดสอบที่ 2 ระบบมีความผิดพลาดเกิดขึ้นในการทดสอบครั้งที่ 6 ซึ่งคิดเป็นเปอร์เซ็นต์ความถูกต้องของกรณีนี้อยู่ที่ 90% และกรณีทดสอบที่ 11 ระบบมีความผิดพลาดเกิดขึ้นในการทดสอบครั้งที่ 3 ซึ่งคิดเป็นเปอร์เซ็นต์ความถูกต้องของกรณีนี้อยู่ที่ 90% เช่นกัน ซึ่งความผิดพลาดที่เกิดขึ้นเนื่องจากระบบต้นแบบใช้เวลาในการตรวจสอบการล็อคพื้นที่การทำงานช่วงระยะเวลาหนึ่ง และเมื่อการพิมพ์รหัสต้นฉบับไปยังพื้นที่การทำงานด้วยความเร็วระดับหนึ่งจะทำให้ระบบทำการตรวจสอบการล็อคพื้นที่การทำงานไม่ทัน ซึ่งการพิมพ์รหัสต้นฉบับด้วยความเร็วระดับนั้นมีโอกาสเกิดขึ้นได้เฉพาะในกรณีที่ใช้เครื่องมือทดสอบอัตโนมัติเท่านั้น ส่วนในกรณีทดสอบที่ 1 กรณีทดสอบที่ 3 ถึงกรณีทดสอบที่ 10 ระบบต้นแบบสามารถทำงานได้ถูกต้องทั้งหมด คิดเป็นเปอร์เซ็นต์ความถูกต้อง 100% และเมื่อคิดเป็นเปอร์เซนต์ความถูกต้องรวมของทุกกรณีทดสอบมีเปอร์เซนต์ความถูกต้องสูงถึง 98.18%

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

ในอดีตการพัฒนาซอฟต์แวร์มีคุณภาพค่อนข้างน้อย เนื่องจากในการพัฒนามักจะใช้นักพัฒนาเพียงคนเดียว และหากเป็นการพัฒนาซอฟต์แวร์ที่มีความซับซ้อนด้วยแล้ว โอกาสที่ซอฟต์แวร์จะมีข้อผิดพลาดก็จะสูงขึ้นด้วยเช่นกัน โดยในปัจจุบันมีผู้ต้องการที่จะใช้บริการซอฟต์แวร์ที่มีความซับซ้อนมากขึ้น จึงได้มีการนำเสนอวิธีการพัฒนาซอฟแวร์ร่วมกัน เพื่อเป็นทางเลือกหนึ่งที่ช่วยในการเพิ่มคุณภาพของซอฟต์แวร์ สำหรับวิธีการพัฒนาซอฟต์แวร์ร่วมกันมีการนำเสนอหลายวิธี หนึ่งในนั้นคือการพัฒนาซอฟต์แวร์ร่วมกันผ่านทางระบบเครือข่าย ซึ่งจะทำให้นักพัฒนาสามารถพัฒนาซอฟต์แวร์ได้ตลอดเวลาเมื่อต้องการ และยังสามารถทำงานได้พร้อมกันหลายคน นักพัฒนาเหล่านั้นสามารถแก้ไข เปลี่ยนแปลง และเพิ่มเติมส่วนต่าง ๆ ของรหัสต้นฉบับได้ทันทีที่ต้องการ

ในงานวิจัยนี้ทำการศึกษาและพัฒนาเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ร่วมกันบนเว็บเบราว์เซอร์ โดยมุ่งเน้นที่การศึกษาค้นคว้าซึ่งมีวัตถุประสงค์ของการศึกษาคือ 1) เพื่อศึกษาแนวคิดและวิธีการในการพัฒนาซอฟต์แวร์ร่วมกันบนเว็บเบราว์เซอร์ และการป้องกันความขัดแย้งเชิงความหมายที่อาจเกิดขึ้นจากผู้พัฒนาซอฟต์แวร์ 2) เพื่อศึกษาแนวคิดและวิธีการในการพัฒนาตัวแจงส่วนภาษาจาวา โดยตัวแจงส่วนภาษาจาวานั้นจะต้องอยู่ในรูปแบบของภาษาจาวาสคริปต์ ทั้งนี้เพื่อให้สามารถนำไปใช้บนเว็บเบราว์เซอร์ได้ และ 3) เพื่อศึกษาแนวคิดและวิธีการล็อคพื้นที่การทำงานที่ผู้พัฒนาแต่ละคนกำลังทำงานอยู่ เพื่อป้องกันความขัดแย้งเชิงความหมายที่อาจเกิดขึ้น รวมถึงมีการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ในการทำงานกับตัวแจงส่วน และการทดสอบความถูกต้องของระบบต้นแบบที่ได้ทำการพัฒนาขึ้น

ขั้นตอนในการดำเนินงานวิจัยนี้แบ่งออกเป็น 3 ส่วน โดยส่วนแรกเป็นการศึกษาและพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกันบนเว็บเบราว์เซอร์ ซึ่งประกอบไปด้วยการออกแบบและพัฒนาตัวแจงส่วนภาษาจาวาที่ทำงานบนเว็บเบราว์เซอร์ด้วย PEG.js และต้นไม้สำหรับนำไปใช้ในระบบต้นแบบซึ่งก็คือ 1) ต้นไม้วากยสัมพันธ์เชิงนามธรรม 2) ต้นไม้การพึ่งพาและสถานะล็อค สำหรับส่วนที่สองเป็นการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา โดยการทดสอบบนเว็บเบราว์เวอร์กูเกิลโครม มอซิลลาไฟร์ฟอกซ์ ซาฟารี โอเปร่า และกูเกิลโครมคานารี และส่วนสุดท้ายเป็นการประเมินความถูกต้องของระบบต้นแบบที่ได้

ทำการศึกษาและพัฒนาขึ้นโดยใช้ลักษณะของการจำลองการทำงานแบบวนซ้ำด้วยตัวทดสอบซิลิเนียมเว็บไดร์เวอร์ ซึ่งได้ทำการแบ่งกรณีสำหรับทดสอบระบบต้นแบบออกเป็นทั้งหมด 11 กรณีที่แตกต่างกัน ตามที่ได้นำเสนอไว้ในบทที่ 3 หัวข้อที่ 3.3 เรื่องการออกแบบการประเมินความถูกต้องของระบบต้นแบบ

5.1 สรุปผลการวิจัย

ในการพัฒนาตัวแจงส่วนภาษาจาวาได้พัฒนาขึ้นบนเครื่องมือสร้างตัวแจงส่วน PEG.js โดยจะต้องทำการสร้างส่วนตัววิเคราะห์คำ และส่วนการแจงส่วนของภาษา เพื่อให้ได้ตัวแจงส่วนภาษาจาวาที่อยู่ในรูปแบบของไฟล์จาวาสคริปต์สำหรับการนำตัวแจงส่วนไปใช้งานบนเว็บเบราว์เซอร์ ผลลัพธ์ที่ได้จากตัวแจงส่วนเมื่อทำการแจงส่วนรหัสต้นฉบับภาษาจาวา จะได้ออกมาเป็นต้นไม้วากยสัมพันธ์เชิงนามธรรม โดยต้นไม้วากยสัมพันธ์เชิงนามธรรมนี้จะถูกนำไปปรับปรุงให้เหลือเฉพาะส่วนที่จำเป็นต้องนำไปใช้

การสร้างต้นไม้ที่ใช้ในระบบต้นแบบประกอบไปด้วยต้นไม้ 2 ประเภทด้วยกันคือ 1) ต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ทำการปรับปรุงจากต้นไม้วากยสัมพันธ์เชิงนามธรรมที่ได้จากการแจงส่วนรหัสต้นฉบับภาษาจาวา ผลลัพธ์ที่ได้หลังทำการปรับปรุงต้นไม้วากยสัมพันธ์เชิงนามจะเหลือเฉพาะโหนดที่จำเป็นต้องนำไปใช้ในการสร้างต้นไม้การพึ่งพาและสถานะล็อคต่อไป และ 2) ต้นไม้การพึ่งพาและสถานะล็อค เป็นต้นไม้ที่สร้างขึ้นต่อจากต้นไม้วากยสัมพันธ์เชิงนามธรรม เพื่อใช้เป็นตัวบ่งบอกว่าส่วนใดของรหัสต้นฉบับมีการพึ่งพากัน ทำให้เมื่อมีการล็อคพื้นที่การทำงานต้นไม้นี้จะบอกว่าส่วนใดของรหัสต้นฉบับที่จะถูกล็อคตามไปด้วย และยังสามารถบอกได้ว่า นักพัฒนาคนใดเป็นเจ้าของพื้นที่การทำงานนั้น ๆ

การพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน โดยวิธีการล็อคพื้นที่การทำงานเพื่อป้องกันการเกิดความขัดแย้งเชิงความหมาย ในขั้นตอนของการพัฒนาระบบต้นแบบได้พัฒนาต่อจากซอฟต์แวร์ ACE ซึ่งเป็นซอฟต์แวร์สำหรับการจัดการข้อมูลเบื้องต้น เครื่องมือที่ใช้สำหรับการรันไฟล์จาวาสคริปต์ได้เลือกใช้ Node.js ซึ่งเป็นเครื่องมือเดียวกันกับที่ใช้รันไฟล์จาวาสคริปต์บนกูเกิลโครม และมีการเพิ่มโมดูล socket.io เพื่อใช้สำหรับการเชื่อมต่อกันระหว่างเซิร์ฟเวอร์กับไคลเอนต์ ซึ่งจะช่วยในจัดการกับการส่งผ่านข้อมูล การปรับปรุงรหัสต้นฉบับ และการแจกไอดีให้กับแต่ละไคลเอนต์

หลังจากที่ได้พัฒนาตัวแจงส่วน และซอฟต์แวร์สำหรับการจัดการข้อมูลตามที่ได้อธิบายมาข้างต้นแล้วนำทั้งสองส่วนมารวมกัน เพื่อพัฒนาเป็นระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับภาษาจาวาร่วมกัน จากนั้นได้ทำการพัฒนาฟังก์ชันในการกำหนดเจ้าของพื้นที่การทำงานให้กับ

ต้นไม้การพึ่งพาและสถานะล็อค ฟังก์ชันในการแสดงพื้นที่ที่ถูกล็อคการทำงานบนไคลเอนต์ของตัวเอง ผลลัพธ์ที่ได้จะทำให้ระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน สามารถทำการล็อคพื้นที่การทำงานได้โดยอัตโนมัติ

การทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา ได้ทำการทดสอบกับเว็บเบราว์เซอร์กูเกิลโครม มอซิลลาไฟร์ฟอกซ์ ซาฟารี โอเปร่า และกูเกิลโครมคานารี โดยแบ่งการทดสอบออกเป็น 2 ด้านคือ 1) ด้านของเวลาที่ใช้สำหรับการแจงส่วนข้อมูลประโยคภาษาจาวา จะพบว่าซาฟารีสามารถแจงส่วนข้อมูลประโยคภาษาจาวาได้เร็วที่สุด รองลงมาคือกูเกิลโครม โอเปร่า กูเกิลโครมคานารี และมอซิลลาไฟร์ฟอกซ์ตามลำดับ จากการทดสอบทำให้เห็นได้ว่าเครื่องประมวลผลจาวาสคริปต์ของเว็บเบราว์เซอร์ซาฟารีมีประสิทธิภาพในการแจงส่วนข้อมูลประโยคภาษาจาวาสูงกว่าเว็บเบราว์เซอร์อื่นที่ได้ทำการทดสอบ และ 2) ด้านของหน่วยความจำสูงสุดที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวา จะพบว่าโอเปร่าเป็นเว็บเบราว์เซอร์ที่ใช้หน่วยความจำในการแจงส่วนข้อมูลประโยคภาษาจาวาน้อยที่สุด รองลงมาคือกูเกิลโครม กูเกิลโครมคานารี ซาฟารี และมอซิลลาไฟร์ฟอกซ์ โดยที่โอเปร่า กูเกิลโครม และกูเกิลโครมคานารีใช้หน่วยความจำใกล้เคียงกัน ชี้ให้เห็นว่าเว็บเบราว์เซอร์โอเปร่า กูเกิลโครม และกูเกิลโครมคานารี มีบริหารและจัดการหน่วยความจำได้ดีกว่าเว็บเบราว์เซอร์ซาฟารี และมอซิลลาไฟว์ฟอกซ์

จากการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ทำให้สามารถสรุปได้ว่าเวลาและหน่วยความจำที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวาขึ้นอยู่กับเครื่องประมวลผลจาวาสคริปต์ของเว็บเบราว์เซอร์นั้น ๆ อย่างไรก็ตามในการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวาทำให้เห็นว่า ในปัจจุบันการแจงส่วนข้อมูลประโยคภาษาจาวาจำนวน 5,535 บรรทัด บนเว็บเบราว์เซอร์นั้นใช้เวลาและหน่วยความจำค่อนข้างมาก แต่หากจะพัฒนาโปรแกรมจัดการข้อมูลที่สามารถแจงส่วนข้อมูลประโยคได้ เว็บเบราว์เซอร์ซาฟารีเป็นตัวเลือกที่ดีที่สุดในการนำมาเป็นพื้นฐานในการพัฒนา รวมทั้งเว็บเบราว์เซอร์ยอดนิยมอย่างกูเกิลโครม และเว็บเบราว์เซอร์ที่ใช้เครื่องประมวลผลจาวาสคริปต์ตัวเดียวกันนั้นคือ กูเกิลโครมคานารี และโอเปร่า

การประเมินความถูกต้องของระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับร่วมกัน ได้ทำการทดสอบตามกรณีทดสอบทั้ง 11 กรณี โดยแบ่งเป็นการทำงานบนพื้นที่ที่ไม่มีล็อค และการทำงานบนพื้นที่มีการล็อคเกิดขึ้น ในการทดสอบใช้เครื่องมือซิลิเนียมเว็บไดร์เวอร์ สำหรับจำลองการทำงานบนเว็บเบราว์เซอร์แบบอัตโนมัติ ผลการประเมินความถูกต้องเมื่อคิดเป็นเปอร์เซนต์ความถูกต้องรวมของทุกกรณีทดสอบมีเปอร์เซนต์ความถูกต้องสูงถึง 98.18% และมีความผิดพลาดของของระบบต้นแบบคิดเป็น 1.82% สำหรับข้อผิดพลาดที่เกิดขึ้นนี้อาจเนื่องมาจากระบบต้นแบบจำเป็นต้องใช้เวลาในการตรวจสอบการล็อคพื้นที่การทำงานช่วงระยะเวลาหนึ่ง แต่เมื่อการพิมพ์

รหัสต้นฉบับไปยังพื้นที่การทำงานที่มีการล็อคด้วยความเร็วระดับหนึ่งจะทำให้ระบบต้นแบบทำการตรวจสอบการล็อคพื้นที่การทำงานไม่ทัน ซึ่งในการพิมพ์รหัสต้นฉบับด้วยความเร็วระดับดังกล่าวมีโอกาสเกิดขึ้นได้เฉพาะในกรณีที่ใช้เครื่องมือทดสอบแบบอัตโนมัติเท่านั้น

5.2 ข้อเสนอแนะ

ในการพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับภาษาจาวาร่วมกัน โดยมีการล็อคพื้นที่การทำงานแบบอัตโนมัติเพื่อป้องกันความขัดแย้งเชิงความหมายที่อาจเกิดขึ้นนั้น ยังไม่ครอบคลุมความสามารถของภาษาจาวาทั้งหมด รวมถึงมีข้อจำกัดในการพัฒนา เช่น คลาสแม่จะต้องถูกประกาศอยู่ด้านบนของคลาสลูกเท่านั้น และสามารถทำการแจงส่วนรหัสต้นฉบับได้เฉพาะภาษาจาวารุ่นที่ 1.6 นอกจากนี้ระบบต้นแบบยังใช้เวลาในการแจงส่วนภาษาจาวาเพื่อทำการสร้างต้นไม้วากยสัมพันธ์เชิงนามธรรม ต้นไม้การพึ่งพาและสถานะล็อคค่อนข้างมาก ทำให้อาจเกิดข้อผิดพลาดในการล็อคพื้นที่การทำงาน

สำหรับการศึกษาและพัฒนาระบบต้นแบบสำหรับการพัฒนารหัสต้นฉบับภาษาจาวาร่วมกันนี้ จะเป็นแนวทางในการนำไปศึกษาและพัฒนาต่อเป็นระบบสำหรับการพัฒนารหัสต้นฉบับภาษาจาวาร่วมกัน ให้มีความสามารถในการแจงส่วนข้อมูลประโยคภาษาจาวารุ่นปัจจุบันและครอบคลุมความสามารถทั้งหมดของภาษาจาวา

รายการอ้างอิง

B. Ford, (2014a). **Packrat Parsing: a Practical Linear-Time Algorithm with Backtracking**. Master's thesis, Massachusetts Institute of Technology, Sep 2002.

B. Ford, (2014b). **The Packrat Parsing and Parsing Expression Grammars Page**. Retrieved November 17, 2014, from: http://bford.info/packrat/

D. Majda, (2013a). **PEG.js (Parser Generator for JavaScript)**. Retrieved January 29, 2014, from: http://pegjs.majda.cz

D. Majda, (2013b). **PEG.js (Parser Generator for JavaScript) - Documentation**. Retrieved January 29, 2014, from: http://pegjs.majda.cz/documentation

Google Inc., (2014). **chrome.** Retrieved January 29, 2014, from: https://www.google.com /chrome/browser/features.html

H. Fan, and C. Sun, (2012a). **Dependency-based Automatic Locking for Semantic Conflict Prevention in Real-Time Collaborative Programming.** SAC'12, March 25-29, pp. 737-742, Riva del Garda, Italy.

H. Fan, and C. Sun, (2012b). **Supporting Semantic Conflict Prevention in Real-Time Collaborative Programming Environments.** SAC '12 Proceedings of the 2012 ACM Symposium on Applied Computing, pp. 39-52, VOL. 12, NO. 2.

H. Fan, C. Sun and H. Shen, (2012). **ATCoPE: Any-Time Collaborative Programming Environment for Seamless Integration of Real-Time and Non-Real-Time Teamwork in Software Development.** GROUP'12, October 27–31 , 2012, pp. 107-116, Sanibel Island, Florida, USA.

I. Z. Schlueter, (2014). **socket.io - node.js realtime framework server.** Retrieved February 2, 2014, from: https://www.npmjs.com/package/socket.io

J. A. Preston, X. Hu, and S. K. Prasad, (2007). **Simulation-based architectural design and implementation of a real-time collaborative editing system.** SpringSim '07: Proceedings of the 2007 spring simulation multiconference, pp. 320-327.

J. Coglan, (2014). **The cross-platform JavaScript class library.** Retrieved September 2, 2014, from: http://jsclass.jcoglan.com/

Jmleyba, (2014). **selenium-webdriver.** Retrieved November 5, 2014, from: https:// www.npmjs.com./package/selenium-webdriver

Joyent Inc., (2014). **node.js.** Retrieved April 5, 2014, from: http://nodejs.org/

J. V. Gesser, (2010). **javaparser.** Retrieved November 5, 2013, from: https://code.google.com /p/javaparser/

M. A. Hayat, (2012). **Beginning parsers with PEG.js.** Retrieved November 2, 2014, from: https://coderwall.com/p/316gba

M. Goldman, (2011). **Role-Based Interfaces for Collaborative Software Development.** UIST'11, October 16–19, pp. 23-26. Santa Barbara, CA, USA.

M. Ericsson, (2013). **Exploratory metrics tool.** Retrieved October 21, 2013, from: https:// github.com/morganericsson

N. Nuansri, (2011). **Lex and Yacc.** Retrieved June 25, 2013, from: staff.cs.psu.ac.th/noi/cs323-554/ปโท/g4322015/lex&yacc1.doc?

R. Duque and C. Bravo, (2008). **Analyzing Work Productivity and Program Quality in Collaborative Programming.** ICSEA '08 (The Third International Conference on Software Engineering Advances, 2008), October 26-31, pp. 270-276.

Sun Microsystems Inc., (2005). **The Java Language Specification, Third Edition.** Retrieved January 25, 2014, from: http://docs.oracle.com/javase/specs/jls/se5.0/html/j3TOC.html

The ACE project, (2014). **Ace: About.** Retrieved November 20, 2014, from: http://ace.c9.io/ #nav=about

T. Mueller, (2013). **h2database.** Retrieved November 21, 2013, from: https://code.google.com/p/ h2database/ downloads/list

W3C HTML Working Group, (2014). **HTML5 A vocabulary and associated APIs for HTML and XHTML.** Retrieved November 21, 2014, from: http://www.w3.org/TR/html5/

Z. Carter, (2013). **Jison - Documentation (Using the Parser from the Web) .** Retrieved November 21, 2013 , from: http://zaach.github.io/jison/docs/#using-the-parser-from-the-web

ภาคผนวก ก

บทความทางวิชาการที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างการศึกษา

รายชื่อบทความที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างการศึกษา

ชลวิศว์ เสภาศีราภรณ์ และ พิชโยทัย มหัทธนาภิวัฒน์ (2557). **การทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา**. การประชุมวิชาการระดับชาติด้านเทคโนโลยีสารสนเทศ ครั้งที่ 6 (The 6th National Conference on Information Technology: NCIT 2014)., หน้า 497 - 502

การประชุมวิชาการระดับชาติด้านเทคโนโลยีสารสนเทศ ครั้งที่ 6 (The 6th National Conference on Information Technology: NCIT 2014)

การทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวา

ชลวิตร์ เสภาศีราภรณ์ และ พิชโยทัย มหัทธนาภิวัฒน์

สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี นครราชสีมา
Emails: taro.rs103@gmail.com, pmh@sut.ac.th

บทคัดย่อ

ปัจจุบันการพัฒนาซอฟต์แวร์มีการใช้งานอินเทอร์เน็ตมากขึ้นทั้งด้านการติดต่อกับผู้ที่พัฒนาซอฟต์แวร์ร่วมกันและการเก็บซอร์สโค้ดที่พัฒนาไว้บนกลุ่มเมฆ (Cloud) เพื่อความสะดวก ความรวดเร็ว และความปลอดภัยในการเก็บซอร์สโค้ด ส่วนในด้านของระบบปฏิบัติการปัจจุบันมีการพัฒนาระบบปฏิบัติการที่ให้ความสำคัญกับความเร็วในการเปิด/ปิดเครื่องคอมพิวเตอร์ ความต้องการในการใช้ทรัพยากรของเครื่องคอมพิวเตอร์ที่น้อย และเน้นการใช้งานด้านการค้นหาข้อมูลทางอินเทอร์เน็ต การตรวจสอบอีเมลล์ และการทำงานเกี่ยวกับเอกสาร จากแนวโน้มดังกล่าวการพัฒนาซอฟต์แวร์ในอนาคตนักพัฒนาอาจพัฒนาซอฟต์แวร์บนเว็บเบราว์เซอร์จากระบบปฏิบัติการที่อยู่บนเครื่องคอมพิวเตอร์แบบพกพาที่ราคาไม่สูงมากนัก

ในบทความนี้เป็นการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์โดยการนำตัวแจงส่วน (Parser) ของภาษาจาวา (Java) มาแจงส่วนข้อมูลประโยคภาษาจาวาบนเว็บเบราว์เซอร์ที่ได้รับความนิยมในปัจจุบัน เพื่อทดสอบในด้านของเวลา และหน่วยความจำของเครื่องคอมพิวเตอร์ที่ใช้ในการแจงส่วน ผลลัพธ์ที่ได้จากการทดสอบสามารถแสดงให้เห็นว่าในด้านของเวลาเว็บเบราว์เซอร์ซาฟารี (Safari) ใช้เวลาในการแจงส่วนที่น้อยกว่าเว็บเบราว์เซอร์อื่นที่ทำการทดสอบอย่างชัดเจน ส่วนในด้านของหน่วยความจำสูงสุดที่ใช้ในการแจงส่วนเว็บเบราว์เซอร์กูเกิลโครม (Google Chrome) กูเกิลโครมคานารี (Google Chrome Canary) และโอเปร่า (Opera) ใช้หน่วยความจำสูงสุดที่ใกล้เคียงกันและน้อยกว่าเว็บเบราว์เซอร์อื่นที่ทำการทดสอบ

คำสำคัญ– ตัวแจงส่วน; เว็บเบราว์เซอร์; ประสิทธิภาพ

1. บทนำ

การพัฒนาซอฟต์แวร์ส่วนมากในปัจจุบันเป็นการพัฒนาซอฟต์แวร์ผ่านทางโปรแกรมจัดการข้อมูล (Editor) ซึ่งมีอยู่หลากหลายประเภทและมีคุณลักษณะที่แตกต่างกันออกไป โปรแกรมจัดการข้อมูลส่วนมากในปัจจุบันจะเป็นโปรแกรมประยุกต์ (Application) ที่ติดตั้งลงบนคอมพิวเตอร์ส่วนบุคคล การเลือกว่าจะใช้โปรแกรมจัดการข้อมูลใดนั้นนักพัฒนาซอฟต์แวร์จะเลือกใช้โปรแกรมจัดการข้อมูลตามคุณลักษณะที่พวกเขาต้องการจะใช้ และตามที่คอมพิวเตอร์ส่วนบุคคลของพวกเขาจะสามารถทำงานได้อย่างราบรื่น

ปัจจุบันในด้านของการพัฒนาซอฟต์แวร์ อินเทอร์เน็ตสามารถเข้าถึงได้ทุกที่ทุกเวลาตามที่ต้องการ การฝากซอร์สโค้ดไว้บนกลุ่มเมฆ จึงมีความนิยมเพิ่มมากขึ้น โดยที่กลุ่มเมฆที่กล่าวถึงคือระบบที่มีเซิร์ฟเวอร์ทำงานร่วมกันเป็นจำนวนมาก และเซิร์ฟเวอร์แต่ละตัวทำงานแยกออกจากกันอย่างเป็นระบบ ทำให้เมื่อเกิดข้อผิดพลาดขึ้นจะมีเซิร์ฟเวอร์อื่นมาทำงานแทนในทันที ซึ่งทำให้ผู้ใช้งานกลุ่มเมฆมั่นใจได้ว่าระบบหรือข้อมูลที่อยู่บนกลุ่มเมฆจะไม่เสียหายและสามารถเรียกใช้งานได้ตลอดเวลา ส่วนในด้านการพัฒนาระบบปฏิบัติการ (Operation system) มีการพัฒนาระบบปฏิบัติการให้ใช้เวลาในการเปิด/ปิดเครื่องที่ลดลง ใช้ทรัพยากรของเครื่องคอมพิวเตอร์ไม่มาก และเน้นการใช้งานตามที่ผู้ใช้งานคอมพิวเตอร์ส่วนมากต้องการ เช่น การค้นหาข้อมูลทางอินเทอร์เน็ต การตรวจสอบอีเมลล์ และการทำงานเกี่ยวกับเอกสาร เป็นต้น ความต้องการเหล่านี้ทำให้เกิดระบบปฏิบัติการที่มีความสามารถดังกล่าวเพิ่มมากขึ้นเรื่อย ๆ สิ่งเหล่านี้ทำให้เห็นแนวโน้มว่าในอนาคตอาจมีการพัฒนาซอฟต์แวร์ด้วยคอมพิวเตอร์แบบพกพาที่ราคาไม่สูงมากนัก โดยนักพัฒนาซอฟต์แวร์จะเชื่อมต่อกับอินเทอร์เน็ตเพื่อที่จะเรียกใช้ซอร์สโค้ดที่ฝากไว้บนกลุ่มเมฆ และพัฒนาซอร์สโค้ดผ่านเว็บเบราว์เซอร์ ซึ่งเว็บเบราว์เซอร์นั้นมีอยู่ในทุกระบบปฏิบัติการ ใช้พื้นที่และความต้องการของระบบที่น้อยในการติดตั้งและใช้งาน [1], [2]

โปรแกรมจัดการข้อมูลบนเว็บเบราว์เซอร์ส่วนมากในปัจจุบันเป็นเพียงโปรแกรมจัดการข้อมูลที่ป้อนข้อความใส่ลงไปอาจมีการเน้นสีเพื่อบอกว่าคำไหนเป็นคำเฉพาะของภาษานั้น ๆ โปรแกรมจัดการข้อมูลในอนาคตเพื่อที่จะรองรับการพัฒนาซอฟต์แวร์บนเว็บเบราว์เซอร์จำเป็นต้องแจงส่วนข้อมูลประโยคที่ป้อนเข้ามาด้วยตัวแจงส่วนของภาษานั้น ๆ เพื่อที่จะบอกถึงรูปแบบในการเขียนซอร์สโค้ดที่ผิดพลาด และเพื่อที่จะนำต้นไม้วากยสัมพันธ์เชิงนามธรรม (Abstract syntax tree) ไปใช้งานต่อไป

ในบทความนี้จะเป็นการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ในปัจจุบันโดยการนำตัวแจงส่วนภาษาจาวาที่พัฒนาต่อยอดจาก PAnalyzer ของ Morgan Ericsson [3] โดยตัวแจงส่วนภาษาจาวาของ Morgan Ericsson สามารถทำงานได้เฉพาะบนบรรทัดคำสั่ง

```
CompilationUnit = s:Spacing p:PackageDeclaration? i:ImportDeclaration* t:TypeDeclaration*
  {
    return new nodes.ASTNode('CompilationUnit', pr([ s, p, i, t]), createLog(line, column));
  }

PackageDeclaration = a:Annotation* p:PACKAGE qi:QualifiedIdentifier s:SEMI
  {
    return new nodes.ASTNode('PackageDeclaration', pr([a, p, qi, s]), createLog(line, column));
  }

ImportDeclaration = i:IMPORT s:STATIC? qi:QualifiedIdentifier ds:( DOT STAR )? ss:SEMI
  {
    return new nodes.ASTNode('ImportDeclaration', pr([i, s, qi, ds, ss ]), createLog(line, column), createLog(line, column));
  }

TypeDeclaration = m:Modifier* n:( ClassDeclaration / EnumDeclaration / InterfaceDeclaration / AnnotationTypeDeclaration )
  {
    return new nodes.ASTNode('TypeDeclaration', pr([m, n]), createLog(line, column));
  }

  / s:SEMI

  {
    return s;
  }
```

รูปที่ 1. ตัวอย่างบางส่วนของไวยากรณ์ภาษาจาวาที่ใช้ในการสร้างตัวแจงส่วน

(Command line) เท่านั้น ซึ่งในการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ ตัวแจงส่วนภาษาจาวานั้นต้องทำงานได้บนเว็บเบราว์เซอร์ด้วย ดังนั้นจึงพัฒนาตัวแจงส่วนจากเดิมที่ทำงานได้เฉพาะบนบรรทัดคำสั่งให้สามารถทำงานบนเว็บเบราว์เซอร์ได้ และให้ผลลัพธ์ที่ได้จากตัวแจงส่วนเก็บข้อมูลของบรรทัดและคอลัมน์ของข้อมูลประโยคภาษาจาที่ป้อนเข้ามาเพิ่มเติม ซึ่งจะมีประโยชน์ในการทำงานต่อไปในอนาคต สำหรับการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ทำโดยการป้อนข้อมูลประโยคภาษาจาวาซึ่งเป็นข้อมูลประโยคภาษาจาวาที่ใช้งานจริงให้กับตัวแจงส่วนภาษาจาวาที่อยู่ในรูปแบบของภาษาจาวาสคริปต์ และนำไปดำเนินการบนเว็บเบราว์เซอร์ที่ต้องการทดสอบ เพื่อหาประสิทธิภาพของเว็บเบราว์เซอร์ในการแจงส่วนข้อมูลประโยคภาษาจาวาและอาจเป็นแนวทางในการพัฒนาเว็บเบราว์เซอร์ต่อไป

สำหรับในส่วนที่ 2 ของบทความนี้จะกล่าวถึงงานวิจัยที่เกี่ยวข้องกับการทดสอบประสิทธิภาพและการเพิ่มประสิทธิภาพให้กับเว็บเบราว์เซอร์ในด้านต่าง ๆ ส่วนที่ 3 จะอธิบายถึงรายละเอียดของตัวแจงส่วนภาษาจาวา จากนั้นในส่วนที่ 4 จะกล่าวถึงวิธีการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ ส่วนที่ 5 จะเป็นการวิเคราะห์ผลที่จะได้จากการทดสอบ และสุดท้ายในส่วนที่ 6 จะเป็นการสรุปผลการทดสอบและพูดถึงแนวทางการพัฒนาต่อยอดในอนาคต

2. งานวิจัยที่เกี่ยวข้อง

ในบทความนี้ได้รวบรวมบทความและงานวิจัยที่เกี่ยวข้องกับการพัฒนาและการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ที่ผ่านมาเพื่อที่จะกล่าวถึงปัญหาของเว็บเบราว์เซอร์และแนวทางในการพัฒนาเว็บเบราว์เซอร์ในอนาคต

ในปี 2010 มีการเปรียบเทียบเครื่องมือวัดจาวาสคริปต์ (Javascript benchmark) กับโปรแกรมที่ใช้งานจริงโดย Paruj Ratanaworabhan และคณะได้กล่าวว่าผลของชุดทดสอบในเครื่องมือวัดจาวาสคริปต์ของ SunSpider และ V8 อาจไม่ได้บอกถึงประสิทธิภาพของเว็บเบราว์เซอร์ในการใช้งานกับโปรแกรมที่ใช้งานจริง โดยที่บางหน้าเว็บไม่ได้มีการเรียกใช้งานจาวาสคริปต์อย่างสม่ำเสมออาจทำให้ผลที่ได้จากการวัดผิดพลาดได้ [4] ซึ่งเมื่อเทียบกับโปรแกรมจัดการข้อมูลบนเว็บเบราว์เซอร์ที่มีการแจงส่วนข้อมูลประโยคอย่างสม่ำเสมอทำให้ต้องเรียกใช้ตัวแจงส่วนซึ่งเป็นไฟล์จาวาสคริปต์อย่างสม่ำเสมอด้วย จากเหตุผลที่กล่าวไปทำให้เครื่องมือวัดจาวาสคริปต์สามารถบอกถึงประสิทธิภาพของเว็บเบราว์เซอร์ได้เช่นกัน

ปี 2011 ได้มีการเพิ่มประสิทธิภาพของโปรแกรมภาษาจาวาสคริปต์โดย Mojtaba Mehrara และ Scott Mahlke ได้เสนอการแก้ไขปัญหาคอขวดโดยการจัดการกับ Guard ด้วยตัวจัดการที่ชื่อว่า ParaGuard ทำให้ประสิทธิภาพโดยเฉลี่ยเพิ่มขึ้นถึง 15% [5] จากผลการทดสอบข้างต้นทำให้เห็นว่ามีการพยายามที่จะเพิ่มประสิทธิภาพของโปรแกรมภาษาจาวาสคริปต์ด้วยจากจัดการกับ Guard เพื่อให้เวลาในการประมวลผลของเครื่องประมวลผลจาวาสคริปต์ (Javascript engine) ลดลง

งานวิจัยที่เกี่ยวข้องกับประสิทธิภาพของเว็บเบราว์เซอร์อีกงานวิจัยหนึ่งคือการใช้หลัก Parameter-based มาใช้ในการปรับปรุงคุณภาพของโค้ด โดยเมื่อต้นปี 2013 I. Costa และคณะได้กล่าวว่าเว็บไซต์ยอดนิยม 100 เว็บไซต์มีเกือบ 60% ที่ฟังก์ชันจาวาสคริปต์ถูกเรียกใช้เพียงครั้งเดียว สิ่งเหล่านี้จะทำให้ใช้เวลาในการประมวลผลมากขึ้นรวมถึงขนาดของรหัสเครื่องที่มีจะขนาดใหญ่ขึ้น I. Costa และคณะจึงเสนอการสร้างฟังก์ชันที่ใช้งานจริงจากการดูพารามิเตอร์ โดยทดลองกับเครื่องมือวัดจาวาสคริปต์ยอดนิยมคือ SunSpider, V8 และ Kraken และมีผลการทดลองที่แสดงให้เห็นว่าสามารถเพิ่มความเร็วให้ SunSpider ได้ถึง 5.38% และลดพื้นที่ของรหัสเครื่องลงได้ถึง 16.72% [6]

ในปี 2013 J. K. Martinsen และคณะได้ใช้วิธีการ Thread-level speculation (TLS) ในเครื่องประมวลผลจาวาสคริปต์ Squirrelfish ซึ่งเป็นส่วนหนึ่งของ WebKit บนเว็บเบราว์เซอร์ ซึ่งการทดสอบดังกล่าวเป็นการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์โดยทดสอบการประมวลผลแบบมัลติคอร์ (Multi-core) [7] ซึ่งปัจจุบันเว็บเบราว์เซอร์ทำงานแบบเป็นลำดับทำให้การประมวลผลบางอย่างรวมถึงการประมวลผลการแจงส่วนข้อมูลประโยคใช้เวลาที่มาก

เมื่อกลางปี 2013 ที่ผ่านมามีการพยายามที่จะเพิ่มประสิทธิภาพของโปรแกรมภาษาจาวาสคริปต์สำหรับโทรศัพท์เคลื่อนที่โดย Aditya Srikanth ได้มุ่งเน้นไปที่การจัดสรรเรจิสเตอร์ (Register allocation) ซึ่งเป็นขั้นตอนการทำงานของเครื่องประมวลผลจาวาสคริปต์ที่ใช้เวลามาก และได้กล่าวอีกว่าการเลือกแบบคงที่ที่เหมาะสมกับรูปแบบของการจัดสรรเรจิสเตอร์มีประสิทธิภาพมากกว่าการจัดสรรเรจิสเตอร์แบบดั้งเดิมถึง 9.1% [8] ซึ่งงานวิจัยนี้บอกถึงการพัฒนาเครื่องประมวลผลจาวาสคริปต์ให้มีประสิทธิภาพที่ดีขึ้น โดยการแก้ไขส่วนของการจัดสรรเรจิสเตอร์ซึ่งเป็นส่วนที่ใช้เวลามากที่สุด

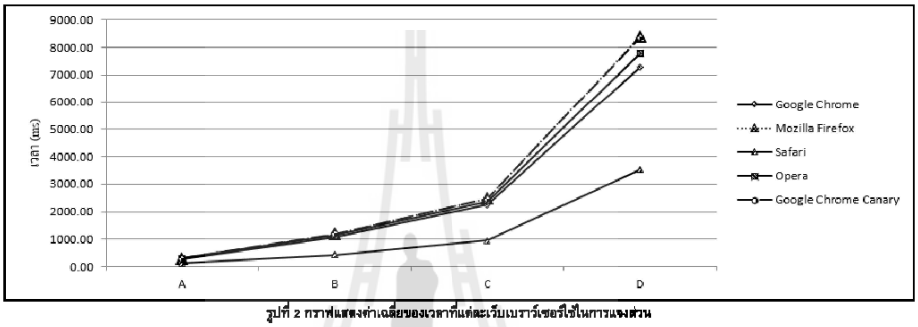
ในบทความนี้จะเสนอการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์โดยใช้ตัวแจงส่วนภาษาจาวาซึ่งจะดำเนินการบนเว็บเบราว์เซอร์เพื่อทดสอบในด้านของระยะเวลาและหน่วยความจำที่เว็บเบราว์เซอร์ใช้ในการแจงส่วนข้อมูลประโยค เพื่อหวังว่าจะเป็นแนวทางในการพัฒนาเว็บเบราว์เซอร์ในอนาคต

3. ตัวแจงส่วนภาษาจาวา (Java parser)

ตัวแจงส่วนภาษาจาวาเป็นตัวแจงส่วนที่ใช้สำหรับการแจงส่วนข้อมูลประโยคของภาษาจาวา ในการที่จะสร้างตัวแจงส่วนภาษาจาวาเองนั้น

ตารางที่ 1. ตารางแสดงค่าเฉลี่ยของเวลาที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วน

เวลา	(A) h2-command-Command.java 344 บรรทัด (ms)	(B) h2-engine-Session.java 1,395 บรรทัด (ms)	(C) h2-engine-Database.java 2,511 บรรทัด (ms)	(D) h2-command-Parser.java 5,535 บรรทัด (ms)
Google Chrome	272.83	1,074.00	2,238.83	7,274.83
Mozilla Firefox	286.67	1,182.67	2,478.33	8,388.33
Safari	129.17	435.33	952.17	3,522.67
Opera	298.50	1,131.33	2,376.00	7,779.50
Google Chrome Canary	308.33	1,184.00	2,465.33	8,352.00



รูปที่ 2 กราฟแสดงค่าเฉลี่ยของเวลาที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วน

ต้องใช้ความสามารถเป็นอย่างมาก ดังนั้นเครื่องมือสำหรับช่วยสร้างตัวแจงส่วนภาษาจาวา (Parser generator) จึงเข้ามาเป็นตัวเลือกในการช่วยลดภาระในการสร้างตัวแจงส่วนภาษาจาวาลง โดยที่ตัวแจงส่วนภาษาจาวาที่ต้องการใช้ในการทดสอบนั้นจำเป็นต้องอยู่ในรูปแบบของภาษาจาวาสคริปต์ เพื่อที่จะนำไปใช้สำหรับการทดสอบประสิทธิภาพบนเว็บเบราว์เซอร์ peg.js [9] ซึ่งเป็นเครื่องมือสำหรับช่วยสร้างตัวแจงส่วนที่เมื่อป้อนไวยากรณ์ของภาษาจาวาให้กับเครื่องมือช่วยสร้างตัวแจงส่วน peg.js แล้วผลลัพธ์ที่ได้จะเป็นตัวแจงส่วนภาษาจาวาที่อยู่ในรูปแบบของภาษาจาวาสคริปต์ ซึ่งจะสามารถนำไปดำเนินการบนเว็บเบราว์เซอร์ได้ทันที peg.js จึงเป็นตัวเลือกที่จะใช้ในการช่วยสร้างตัวแจงส่วนภาษาจาวาในรูปแบบของภาษาจาวาสคริปต์

สำหรับตัวอย่างบางส่วนของไวยากรณ์ภาษาจาวาที่ป้อนให้กับเครื่องมือช่วยสร้างตัวแจงส่วน peg.js จะแสดงอยู่ในรูปที่ 1 ซึ่งเมื่อป้อนไวยากรณ์ภาษาจาดังกล่าวให้กับเครื่องมือช่วยสร้างตัวแจงส่วน peg.js แล้วจะได้ตัวแจงส่วนภาษาจาวาในรูปแบบของภาษาจาวาสคริปต์ออกมา หลังจากนั้นจะนำไปทดสอบกับเว็บเบราว์เซอร์ที่เป็นที่นิยมในปัจจุบันเพื่อเปรียบเทียบประสิทธิภาพของเว็บเบราว์เซอร์แต่ละตัว โดยวิธีการทดสอบจะอธิบายในส่วนที่ 4 ของบทความ

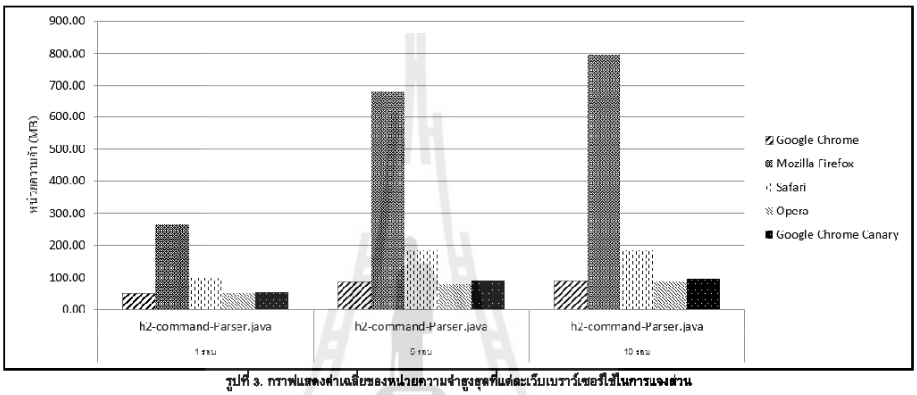
4. วิธีการทดสอบ

การทดสอบประสิทธิภาพของเว็บเบราว์เซอร์จะทดสอบบนเครื่องคอมพิวเตอร์ที่ใช้หน่วยประมวลผลหลัก (CPU) คือ Intel Core i5-2430M หน่วยความจำหลัก (RAM) ทั้งหมด 8 GB ทำงานบนระบบปฏิบัติการ Window 7 64 bit ซึ่งจะทำการทดสอบทั้งหมด 2 ด้าน คือ ด้านที่ 1 เป็นการทดสอบเวลาที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวา และด้านที่ 2 เป็นการทดสอบหน่วยความจำในการแจงส่วนข้อมูลประโยคภาษาจาวา โดยที่ข้อมูลประโยคภาษาจาวาที่ใช้ในการทดสอบเป็นข้อมูลจริงของโปรเจค H2 ซึ่งคือฐานข้อมูลของจาวาเอสคิวแอล (Java SQL database) [10] ที่มีการพัฒนาอย่างต่อเนื่อง โดยเลือกไฟล์ที่มีนามสกุลจาวาออกมาจากโปรเจค H2 ทั้งหมด 4 ไฟล์ สำหรับการเลือกไฟล์นั้นดูจากบรรทัดของข้อมูลภายในไฟล์ ซึ่งทั้ง 4 ไฟล์ที่เลือกมามีบรรทัดทั้งหมด 344, 1,395, 2,511 และ 5,535 บรรทัด ตามลำดับ โดยที่จำนวนบรรทัดที่มากที่สุดของโปรเจค H2 คือ 5,535 บรรทัด ทดสอบกับเว็บเบราว์เซอร์ที่แตกต่างกัน 5 เว็บเบราว์เซอร์ได้แก่ เว็บเบราว์เซอร์กูเกิลโครม เวอร์ชัน 31 มอซิลลาไฟร์ฟอกซ์ (Mozilla Firefox) เวอร์ชัน 25 ซาฟารี เวอร์ชัน 5 โอเปร่า เวอร์ชัน 18 และกูเกิลโครมคานารี เวอร์ชัน 33 โดยการทดสอบจะทดสอบการแจงส่วนข้อมูลประโยคของภาษาจาวาในไฟล์นามสกุลจาวา ด้วยตัวแจงส่วนภาษาจาวาที่อยู่ในรูปแบบของภาษาจาวาสคริปต์เพื่อให้ได้ต้นไม้วากยสัมพันธ์

การทดสอบทั้งหมดจะทดสอบบนเว็บเบราว์เซอร์ที่กล่าวถึงข้างต้น ซึ่งในตารางที่ 1 จะเป็นการแสดงค่าเฉลี่ยที่ได้จากการทดสอบในด้านของเวลาในการแจงส่วน ค่าเฉลี่ยดังกล่าวได้จากการแจงส่วนข้อมูลประโยคภาษาจาวาในไฟล์แต่ละไฟล์ที่นำมาทดสอบไฟล์ละ 10 รอบ และจับเวลาในแต่ละรอบ เมื่อครบทั้ง 10 รอบแล้วจะเรียงเวลาที่ได้จากน้อย

ตารางที่ 2 ตารางแสดงค่าเฉลี่ยของหน่วยความจำสูงสุดที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วน

หน่วยความจำ	h2-command-Parser.java 1 รอบ (MB)	h2-command-Parser.java 5 รอบ (MB)	h2-command-Parser.java 10 รอบ (MB)
Google Chrome	51.79	86.12	89.33
Mozilla Firefox	266.55	681.36	794.24
Safari	98.78	184.38	184.48
Opera	51.79	81.47	87.81
Google Chrome Canary	56.11	89.59	94.38



รูปที่ 3. กราฟแสดงค่าเฉลี่ยของหน่วยความจำสูงสุดที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วน

ไปมากและตัดเวลาที่น้อยที่สุด 2 อันดับและมากที่สุด 2 อันดับออกแล้วจึงนำเวลา 6 อันดับที่เหลือมาหาค่าเฉลี่ย โดยค่าเฉลี่ยของเวลาในการแจงส่วนที่ได้ออกมานั้นจะเห็นได้ว่าในการแจงส่วนไฟล์จำนวน 344, 1,395, 2,511 และ 5,535 บรรทัด เว็บเบราว์เซอร์ซาฟารีใช้เวลาในการแจงส่วนน้อยที่สุดคือ 129.17, 435.33, 952.17 และ 3,522.67 มิลลิวินาทีตามลำดับ โดยที่เว็บเบราว์เซอร์อื่นที่ทำการทดสอบมีค่าเฉลี่ยของเวลาที่ใช้ในการแจงส่วนมากกว่าเว็บเบราว์เซอร์ซาฟารีและมีค่าที่ใกล้เคียงกัน โดยค่าเฉลี่ยของเวลาของเว็บเบราว์เซอร์อื่นที่ทำการทดสอบจะอยู่ที่ประมาณ 296.5, 1,131.33, 2,376 และ 7,779.5 มิลลิวินาทีตามลำดับ เมื่อสร้างเป็นกราฟค่าเฉลี่ยของเวลาที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วนดังรูปที่ 2 ซึ่งจะให้ตัวอักษร A แทนไฟล์ h2-command-Command.java ซึ่งมีจำนวนบรรทัดทั้งหมด 344 บรรทัด ตัวอักษร B แทนไฟล์ h2-engine-Session.java ซึ่งมีจำนวนบรรทัดทั้งหมด 1,395 บรรทัด ตัวอักษร C แทนไฟล์ h2-engine-Database.java ซึ่งมีจำนวนบรรทัดทั้งหมด 2,511 บรรทัด และตัวอักษร D แทนไฟล์ h2-command-Parser.java ซึ่งมีจำนวนบรรทัดทั้งหมด 5,535 บรรทัด จะเห็นว่าเว็บเบราว์เซอร์ซาฟารีมีแนวโน้มในการใช้เวลาในการแจงส่วนที่น้อยกว่าเว็บเบราว์เซอร์อื่นที่ทำการทดสอบอย่างชัดเจนและยังแสดงให้เห็นอีกว่าเวลาที่ใช้ในการแจงส่วนของเว็บเบราว์เซอร์กูเกิลโครม กูเกิลโครมคานารี โอเปร่า และมอซิลลาไฟร์ฟอกซ์มีค่าที่ใกล้เคียงกัน

ในตารางที่ 2 จะเป็นตารางสำหรับแสดงค่าเฉลี่ยของหน่วยความจำสูงสุดที่ใช้ในการแจงส่วนข้อมูลประโยคของแต่ละเว็บเบราว์เซอร์ สำหรับการทดสอบและหาค่าเฉลี่ยของหน่วยความจำสูงสุดที่เว็บเบราว์เซอร์ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวานั้น จะมุ่งเน้นไปที่จำนวนของบรรทัดในไฟล์ที่ทำการทดสอบ โดยจะทดสอบกับไฟล์ที่มีจำนวนบรรทัดมากที่สุดในโปรเจค H2 และทำการวนรอบแจงส่วนข้อมูลประโยคทั้งหมด 3 กรณีคือ จะวนรอบ 1, 5, และ 10 รอบตามลำดับเพื่อหาหน่วยความจำสูงสุดที่เว็บเบราว์เซอร์แต่ละตัวใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวา ซึ่งจะทดสอบทั้งหมดกรณีละ 10 รอบ เมื่อครบทั้ง 10 รอบแล้วจะหาค่าเฉลี่ยของหน่วยความจำสูงสุดที่ใช้โดยการเรียงลำดับข้อมูลที่ได้ และตัดข้อมูลที่มีค่าน้อยที่สุด 2 อันดับและมากที่สุด 2 อันดับแล้วจึงนำ 6 อันดับที่เหลือมาหาค่าเฉลี่ยของหน่วยความจำสูงสุดที่ใช้ในการแจงส่วน โดยค่าเฉลี่ยของหน่วยความจำสูงสุดที่ได้ออกมานั้นจะเห็นว่าเมื่อแจงส่วนไฟล์ที่มีจำนวนบรรทัดมากที่สุดในโปรเจค H2 ติดต่อกัน 1 รอบ 5 รอบและ 10 รอบตามลำดับ แล้วมอซิลลาไฟร์ฟอกซ์ใช้หน่วยความจำในการแจงส่วนมากกว่าเว็บเบราว์เซอร์อื่นที่ทำการทดสอบคือใช้หน่วยความจำสูงสุดเท่ากับ

265.55, 681.36 และ 794.24 เมกะไบต์ตามลำดับ ส่วนเว็บเบราว์เซอร์ซาฟารีใช้หน่วยความจำสูงสุดรองลงมาคือ 98.78, 184.38 และ 184.48 เมกะไบต์ตามลำดับ ส่วนเว็บเบราว์เซอร์ที่เหลือที่ทดสอบคือเว็บ

เบราว์เซอร์กูเกิลโครม กูเกิลโครมคานารี และโอเปร่าใช้หน่วยความจำสูงสุดใกล้เคียงกันคือประมาณ 51.79, 86.12 และ 89.33 เมกะไบต์ ตามลำดับ เมื่อสร้างเป็นกราฟที่แสดงค่าเฉลี่ยของหน่วยความจำสูงสุดที่แต่ละเว็บเบราว์เซอร์ใช้ในการแจงส่วนดังรูปที่ 3 จะเห็นได้อย่างชัดเจนว่าเว็บเบราว์เซอร์มอซิลลาไฟร์ฟอกซ์ใช้หน่วยความจำมากที่สุด รองลงมาคือเว็บเบราว์เซอร์ซาฟารี และเว็บเบราว์เซอร์ที่ใช้หน่วยความจำน้อยที่สุดคือเว็บเบราว์เซอร์กูเกิลโครม กูเกิลโครมคานารี และโอเปร่า ซึ่งใช้หน่วยความจำสูงสุดในการแจงส่วนที่ใกล้เคียงกัน

5. วิเคราะห์ผลการทดสอบ

เมื่อได้ทำการทดสอบกับเว็บเบราว์เซอร์กูเกิลโครม มอซิลลาไฟร์ฟอกซ์ ซาฟารี โอเปร่า และกูเกิลโครมคานารีในแง่ของเวลาสำหรับการแจงส่วนข้อมูลประโยคภาษาจาวา จะเห็นได้ว่าซาฟารีเป็นเว็บเบราว์เซอร์ที่สามารถแจงส่วนข้อมูลประโยคภาษาจาวาได้เร็วที่สุด รองลงมาคือกูเกิลโครม โอเปร่า กูเกิลโครมคานารี และมอซิลลาไฟร์ฟอกซ์ตามลำดับ โดยที่เว็บเบราว์เซอร์กูเกิลโครมคานารีและมอซิลลาไฟร์ฟอกซ์ใช้เวลาในการแจงส่วนข้อมูลประโยคภาษาจาวาใกล้เคียงกัน ทำให้เห็นได้ว่าเครื่องประมวลผลจาวาสคริปต์ ของเว็บเบราว์เซอร์ซาฟารีมีประสิทธิภาพในการแจงส่วนข้อมูลประโยคภาษาจาวาสูงกว่าเว็บเบราว์เซอร์อื่นที่ได้ทำการทดสอบ

เมื่อทดสอบกับเว็บเบราว์เซอร์กูเกิลโครม มอซิลลาไฟร์ฟอกซ์ ซาฟารี โอเปร่า และกูเกิลโครมคานารี ในแง่ของขนาดของหน่วยความจำสูงสุดที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวา จะเห็นว่าโอเปร่าเป็นเว็บเบราว์เซอร์ที่ใช้หน่วยความจำในการแจงส่วนข้อมูลประโยคภาษาจาวาน้อยที่สุด รองลงมาคือกูเกิลโครม กูเกิลโครมคานารี ซาฟารี และมอซิลลาไฟร์ฟอกซ์ โดยที่โอเปร่า กูเกิลโครม และกูเกิลโครมคานารีใช้หน่วยความจำใกล้เคียงกัน ชี้ให้เห็นว่าเว็บเบราว์เซอร์โอเปร่า กูเกิลโครม และกูเกิลโครมคานารีบริหารและจัดการหน่วยความจำได้ดีกว่าเว็บเบราว์เซอร์ซาฟารี และมอซิลลาไฟร์ฟอกซ์

6. สรุปผล

จากผลการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ทำให้เห็นว่าเว็บเบราว์เซอร์ซาฟารีใช้เวลาในการแจงส่วนข้อมูลประโยคภาษาจาวาที่น้อยกว่าเว็บเบราว์เซอร์อื่นที่ทำการทดสอบ นั้นคือเครื่องประมวลผลจาวาสคริปต์ของเว็บเบราว์เซอร์ซาฟารีมีประสิทธิภาพมากกว่าเว็บเบราว์เซอร์อื่นที่ทำการทดสอบ นอกจากนี้เว็บเบราว์เซอร์ที่ใช้เครื่องประมวลผลจาวาสคริปต์เดียวกันได้แก่ กูเกิลโครม โอเปร่า และกูเกิลโครมคานารี [11] ยังมีเวลาในการแจงส่วนข้อมูลประโยคภาษาจาวาที่ใกล้เคียงกัน เช่นเดียวกันกับหน่วยความจำสูงสุดที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวาที่เว็บเบราว์เซอร์ที่ใช้เครื่องประมวลผลจาวาสคริปต์เดียวกันใช้หน่วยความจำสูงสุดใกล้เคียงกัน ดังนั้นจะสามารถสรุปได้ว่าเวลาและหน่วยความจำที่ใช้ในการแจงส่วนข้อมูลประโยคภาษาจาวาขึ้นอยู่กับเครื่องประมวลจาวาสคริปต์ของเว็บเบราว์เซอร์นั้น ๆ

อย่างไรก็ตามในการทดสอบประสิทธิภาพของเว็บเบราว์เซอร์ด้วยตัวแจงส่วนภาษาจาวาทำให้เห็นว่า ในปัจจุบันการแจงส่วนข้อมูลประโยคภาษาจาวาจำนวน 5,535 บรรทัด บนเว็บเบราว์เซอร์นั้นใช้เวลาและหน่วยความจำค่อนข้างมาก แต่หากจะพัฒนาโปรแกรมจัดการข้อมูลที่สามารถแจงส่วนข้อมูลประโยคได้ เว็บเบราว์เซอร์ซาฟารีเป็นตัวเลือกที่ดีที่สุดในการนำมาเป็นพื้นฐานในการพัฒนา รวมทั้งเว็บเบราว์เซอร์ยอดนิยมอย่างกูเกิลโครม และเว็บเบราว์เซอร์ที่ใช้เครื่องประมวลผลจาวาสคริปต์ตัวเดียวกันนั้นคือ กูเกิลโคมคานารีและโอเปร่า ซึ่งเว็บเบราว์เซอร์ที่กล่าวมาเหล่านี้ใช้เวลาและหน่วยความจำในการแจงส่วนที่น้อยกว่าเว็บเบราว์เซอร์มอซิลลาไฟร์ฟอกซ์

งานวิจัยในอนาคตจะทดสอบประสิทธิภาพของเว็บเบราว์เซอร์กับโปรแกรมจัดการข้อมูลที่สามารถแจงส่วนข้อมูลประโยคของภาษาได้จริง เพื่อดูถึงการตอบสนองของเว็บเบราว์เซอร์กับผู้ใช้งานทั้งในด้านของเวลา หน่วยความจำสูงสุดที่ใช้ และความถูกต้องของการแจงส่วน

เอกสารอ้างอิง

[1] Notebookspac, "ทำความรู้จัก Google Chrome OS," 25 พฤศจิกายน 2552. [ออนไลน์] เข้าถึงได้จาก: http://www.notebookspec.com/ทำความรู้จัก-google-chrome-os/13320/. [เข้าถึงเมื่อ 15 พฤศจิกายน 2556]

[2] Wikipedia, "Cloud computing," 14 November 2013. [Online]. Available: http://en.wikipedia.org/ wiki/Cloud_computing. [Accessed 15 November 2013]

[3] M. Ericsson "Exploratory metrics tool," 6 May 2013. [Online]. Available: https://github.com/morganericsson. [Accessed 21 October 2013]

[4] P. Ratanaworabhan, B. Livshits, and B. G. Zorn, "JSMeter: comparing the behavior of Javascript benchmarks with real web applications," USENIX conference on Web application development, CA, USA, 2010.

[5] M. Mehrara and S. Mahlke, "Dynamically accelerating client-side web applications through decoupled execution," 9th Annual IEEE/ACM International Symposium on Code Generaion and Optimization, DC, USA, 2011.

[6] I. Costa, P. Alves, H. N. Santos and F. M. Q. Pereira, "Just-in-time value specialization," Code Generation and Optimization (CGO), Feb. 2013.

[7] J. K. Martinsen, H. Grahn and A. Isberg, "Using speculation to enhance javascript performance in web applications," IEEE Internet Computing, 2013.

[8] A. Srikanth, "Characterization and optimization of JavaScript programs for mobile systems," The University of Texas at Austin, 2013.

[9] D. Majda, "PEG.js," [Online] Available: http://pegjs.majda.cz/. [Accessed 5 May 2013]

[10] T. Mueller "h2database," 28 July 2013 [Online] Available: https://code.google.com/p/h2database/ downloads/list. [Accessed 21 November 2013]

[11] Wikipedia, "Javascript engine," 17 October 2013 [Online] Available: http://en.wikipedia.org/wiki/JavaScript_engine. [Accessed 3 November 2013]

ประวัติผู้เขียน

นายชลวิศว์ เสภาศีราภรณ์ เกิดเมื่อวันที่ 10 กรกฎาคม พ.ศ. 2533 ที่จังหวัดนครราชสีมา สำเร็จการศึกษาระดับประถมศึกษาที่ 6 โรงเรียนมารีย์วิทยา จังหวัดนครราชสีมา จากนั้นสำเร็จการศึกษาระดับชั้นมัธยมศึกษาปีที่ 6 โรงเรียนราชสีมาวิทยาลัย จังหวัดนครราชสีมา และสำเร็จการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ในปีการศึกษา 2554 หลังจากสำเร็จการศึกษาระดับปริญญาตรีแล้ว ได้เข้าศึกษาต่อระดับปริญญาโท สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ในปีการศึกษา 2555 ในระหว่างศึกษาได้มีโอกาสเป็นผู้ช่วยสอนปฏิบัติการรายวิชาเทคโนโลยีเชิงวัตถุ (Object - Oriented Technology) วิชาหัวข้อขั้นสูงในวิศวกรรมคอมพิวเตอร์ 1 (Advanced Topics In Computer Engineering I) วิชาการวิเคราะห์และออกแบบระบบงาน (System Analysis and Design) วิชาวิศวกรรมซอฟต์แวร์ (Software Engineering) วิชาการโปรแกรมโดยยึดเหตุการณ์ (Event - Driven Programming) โดยได้รับทุนผู้ช่วยสอนและผู้ช่วยวิจัยในระหว่างการศึกษา และมีผลงานการประชุมวิชาการระดับชาติ 1 ผลงาน ดังแสดงในภาคผนวก ก