# การเปรียบเทียบความซับซ้อนของโอเพ่นซอสซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

อรสา พัสดุ วิทิดา จงศุภชัยสิทธิ์ และ พรชัย มงคลนาม

คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

Emails: {orasa|vithida|pornchai}@sit.kmutt.ac.th

126 ถ. ประชาอุทิศ เขตทุ่งครุ กรุงเทพฯ 10140

Emails: {orasa|vithida|pornchai}@sit.kmutt.ac.th

## บทคัดย่อ

งานวิจัยนี้มุ่งหวังที่จะศึกษาเปรียบเทียบความซับซ้อนของโอเพ่นซอสซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี (มจธ.) โดยวัดความซับซ้อนของโปรแกรมตัวอย่าง โดยใช้ตัววัดซอฟต์แวร์ (Software Metric) ชนิดต่าง ๆ คือความซับซ้อนไซโคลมาติก (Cyclomatic Complexity) ดับเบิลยูเอ็มซี (Weighted Method per Class: WMC) อาร์เอฟซี (Response for a Class: RFC) ซีบีโอ (Coupling between Object Classes: CBO) แอลซีโอเอ็ม (Lack of Cohesion of Method: LCOM) แฟน-อิน (Fan-in) และแฟน-เอ้าท์ (Fan-out) เพื่อเปรียบเทียบทักษะในการพัฒนาซอฟต์แวร์ของผู้พัฒนาโอเพ่นซอสซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. ว่ามีความแตกต่างกันมากน้อยเพียงใด ในเรื่องของจุดเด่นและจุดด้อยในการพัฒนาซอฟต์แวร์ เพื่อเป็นประโยชน์ต่อนักศึกษาและอาจารย์ผู้สอน ในการนำมาวางแผนในการสอนและแก้ไขปรับปรุงคุณภาพของซอฟต์แวร์ต่อไป ซึ่งโปรแกรมตัวอย่างที่นำมาใช้ในงานวิจัยนี้ เป็นโปรแกรมที่เขียนด้วยภาษาจาวา มีจำนวน 2 กลุ่มตัวอย่าง คือโอเพ่นซอสซอฟต์แวร์ เก็บข้อมูลมาจาก http://java-source.net และกลุ่มตัวอย่างที่สองคือโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. จำนวน 25 โปรแกรม เท่ากัน โดยใช้จำนวนบรรทัดของโปรแกรมเป็นเกณฑ์ในการเลือก และใช้กราฟเส้นเข้ามาช่วยในการเปรียบเทียบ

***คำสำคัญ--*** ความซับซ้อน; ตัววัดซอฟต์แวร์; โอเพ่นซอสซอฟต์แวร์; โปรแกรมภาษาจาวา; จำนวนบรรทัดของโปรแกรม

## 1. บทนำ

ปัจจุบันมีการพัฒนาซอฟต์แวร์กันอย่างแพร่หลาย เรื่องของความซับซ้อนถือว่าเป็นเรื่องที่สำคัญของซอฟต์แวร์ หากซอฟต์แวร์นั้นมีความซับซ้อนมากก็จะทำให้ปรับปรุงแก้ไขยาก เมื่อมีข้อผิดพลาดเกิดขึ้น ทำให้เสียค่าใช้จ่ายในการบำรุงรักษา ส่งผลให้ซอฟต์แวร์นั้นไม่มีประสิทธิภาพเท่าที่ควร ดังนั้นหากซอฟต์แวร์ที่พัฒนาขึ้นมานั้นไม่มีความซับซ้อนหรือมีน้อยก็จะดี

ในการวัดความซับซ้อนนั้น จำนวนบรรทัดของโปรแกรม (Line of Code) ถือว่ามีความสำคัญต่อความซับซ้อนของโปรแกรมเช่นกัน เนื่องจากโปรแกรมที่มีจำนวนบรรทัดของโปรแกรมมากมีผลทำให้เกิดความซับซ้อนเพิ่มมากขึ้น ดังนั้นเราต้องมีการนับจำนวนบรรทัดของโปรแกรม เพื่อดูขนาดของโปรแกรมว่ามีขนาดเล็กหรือขนาดใหญ่ ซึ่งสามารถวัดได้จากโปรแกรม Code Counter Pro [1]

หลังจากที่ได้กลุ่มตัวอย่างและทราบขนาดของจำนวนบรรทัดของโปรแกรมตัวอย่างแล้ว เราสามารถวัดความซับซ้อนของโปรแกรมภาษาจาวาโดยใช้ตัววัดซอฟต์แวร์ชนิดต่าง ๆ ซึ่งตัววัดซอฟต์แวร์ที่นำมาใช้ในการวัดความซับซ้อนนั้น มีทั้งหมด 7 ตัว คือ ความซับซ้อนไซโคลมาติก, ดับเบิลยูเอ็มซี อาร์เอฟซี ซีบีโอ แอลซีโอเอ็ม แฟน-อิน และ แฟน-เอ้าท์ ซึ่งสามารถใช้ได้กับโปรแกรม Jhawk [2] และตัววัดซอฟต์แวร์แต่ละตัวนั้นมีคุณสมบัติที่แตกต่างกัน สามารถใช้วัดความซับซ้อนได้หลายด้าน เช่นวัดความซับซ้อนในระดับเมท็อด (Method) วัดความซับซ้อนในระดับคลาส (Class) วัดความซับซ้อนในเรื่องของคลุปปลิ้ง (Coupling) และ โคฮีชั่น (Cohesion) เป็นต้น

Chidamber และ Kemerer [3] ทำการศึกษาเปรียบเทียบจุดเด่นและจุดด้อย รวมทั้งความแตกต่างของตัววัดซอฟต์แวร์แต่ละประเภท โดยมีวัตถุประสงค์เพื่อศึกษาความแตกต่างของตัววัดซอฟต์แวร์ที่มีอยู่ในปัจจุบันและนำจุดด้อยของตัววัดซอฟต์แวร์แต่ละชนิดมาพัฒนาให้ดีขึ้น

Jung et al. [4] ได้ศึกษาการนำตัววัดซอฟต์แวร์เข้ามาวัดความซับซ้อนของโปรแกรมที่เขียนด้วยภาษา COBOL โดยกำหนดให้จำนวนบรรทัดของโปรแกรมเป็นตัวแปรอิสระ และตัววัดซอฟต์แวร์ต่าง ๆ เป็นตัวแปรตาม พบว่าจำนวนบรรทัดของโปรแกรมสามารถนำมาพยากรณ์ความซับซ้อนได้ และความซับซ้อนไซโคลมาติกสามารถนำมาพยากรณ์ Halstead's Software Science Metric ได้

ในการวิจัยนี้จะศึกษาความซับซ้อนของโปรแกรมที่เขียนด้วยภาษาจาวา เพื่อทำการเปรียบเทียบค่าความซับซ้อนของกลุ่มตัวอย่าง 2 กลุ่มคือ โอเพ่นซอสซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี (มจธ.) ว่ามีความแตกต่างกันมากน้อยเพียงใด ในเรื่องของความซับซ้อน ซึ่งกลุ่มตัวอย่างที่นำมาใช้ในงานวิจัยครั้งนี้ เป็นโปรแกรมที่เขียนด้วยภาษาจาวา มี

จำนวนโปรแกรมตัวอย่างกลุ่มละ 25 โปรแกรมเท่ากัน โดยใช้จำนวนบรรทัดของโปรแกรมเป็นเกณฑ์ในการเลือก และลักษณะของโปรแกรมตัวอย่างของทั้ง 2 กลุ่มนั้น ส่วนใหญ่จะเป็นโปรแกรมแบบเว็บแอปพลิเคชั่น (Web Application) ซึ่งโอเพ่นซอสซอฟต์แวร์ที่นำมาใช้ในการเปรียบเทียบกับโปรแกรมโครงงานของนักศึกษานั้นจะเป็นโปรแกรมที่พัฒนามาจากนักวิจัยและผู้ที่มีความสนใจในเรื่องเดียวกันทำให้มีความเชี่ยวชาญและมีประสบการณ์มากกว่าโปรแกรมที่นักศึกษาพัฒนาขึ้นมา หลังจากนั้นนำโปรแกรมตัวอย่างมาวัดความซับซ้อนโดยใช้ตัววัดซอฟต์แวร์ชนิดต่าง ๆ เมื่อได้ค่าความซับซ้อนของตัววัดซอฟต์แวร์แต่ละตัวแล้ว ก็จะนำค่าความซับซ้อนที่ได้มาเปรียบเทียบเพื่อดูความแตกต่างของกลุ่มตัวอย่างทั้ง 2 กลุ่ม ว่ามีความซับซ้อนมากน้อยเพียงใด มีจุดเด่นและจุดด้อยในด้านใดบ้าง เพื่อนำข้อมูลที่ได้ไปวางแผนในการพัฒนาซอฟต์แวร์ต่อไป และเป็นประโยชน์ต่อนักศึกษาและอาจารย์ผู้สอน

## 2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

### 2.1. ตัววัดซอฟต์แวร์ที่นำมาใช้ในการวัดความซับซ้อน

ตัววัดซอฟต์แวร์ที่นำมาใช้ในงานวิจัยครั้งนี้ แบ่งออกได้เป็น 2 ประเภทคือ 1. ตัววัดซอฟต์แวร์ที่นำมาใช้ในการวัดความซับซ้อนของเมท็อด 2. ตัววัดซอฟต์แวร์ที่นำมาใช้ในการวัดความซับซ้อนของคลาส สามารถอธิบายรายละเอียดได้ดังนี้

#### *2.1.1. ตัววัดซอฟต์แวร์ที่ใช้ในการวัดความซับซ้อนของเมท็อด*

ความซับซ้อนไซโคลมาติก [5] เป็นตัววัดซอฟต์แวร์ที่ใช้ในการคำนวณค่าความซับซ้อนของแต่ละเมท็อด สามารถคำนวณได้ดังนี้

- คำนวณจากเส้นทางการเชื่อมต่อในรูปของกราฟควบคุมกระแส (Flow Graph) ซึ่งคิดจากจำนวนของ Edge และ จำนวน Node ดังสูตรต่อไปนี้

$$v(G) = e - n + 2 \quad (1)$$

เมื่อ $v(G)$= ค่าความซับซ้อนไซโคลมาติก

$e$ = จำนวนของ Edge

$n$ = จำนวนของ Node ซึ่งเป็นทั้งแบบ Sequential Statement และ Predicate

- คำนวณจากจำนวน Predicate ซึ่ง Predicate คือจำนวนของ Condition Statement ดังสูตรต่อไปนี้

$$v(G) = P + 1 \quad (2)$$

เมื่อ $P$ = จำนวนของ Predicate Node ที่อยู่ในกราฟควบคุมกระแส

#### *2.1.2. ตัววัดซอฟต์แวร์ที่ใช้ในการวัดความซับซ้อนของคลาส*

1. ดับเบิลยูเอ็มซี [6] คือตัววัดซอฟต์แวร์ที่ใช้ในการวัดความซับซ้อนของคลาส ซึ่งเกิดจากผลรวมของค่าความซับซ้อนไซโคลมาติกของแต่ละเมท็อดภายในคลาสนั้น ดังสูตรต่อไปนี้

$$\text{WMC} = \sum_{i=0}^{n} c_i \quad (3)$$

เมื่อ $C_i$ = ค่าความซับซ้อนไซโคลมาติกของแต่ละเมท็อด

2. อาร์เอฟซี [6] คือจำนวนเมท็อดที่มีการสร้างไว้ใช้งานภายในคลาสรวมกับจำนวนเมท็อด ของคลาสอื่นที่ถูกเรียกใช้โดยเมท็อดต่าง ๆ ในคลาสนั้น ๆ

3. ซีบีโอ [6] คือ ผลรวมของจำนวนคลาสอื่นที่เข้ามาเรียกใช้คลาสที่กำลังพิจารณารวมกับจำนวนคลาสอื่นที่ถูกเรียกใช้โดยคลาสที่กำลังพิจารณา (นั้นคือเป็นผลรวมของค่าแฟน-อินและแฟน-เอ้าท์)

4. แอลซีโอเอ็ม [6] คือตัววัดซอฟต์แวร์ที่ใช้ ในการวัดจำนวนของเมท็อดที่มีการเรียกใช้ตัวแปร ที่มีการประกาศไว้ภายในคลาส ดังสูตรต่อไปนี้

$$LCOM\,HS = \frac{M - \left(\frac{Sum(MF)}{F}\right)}{M - 1} \quad (4)$$

เมื่อ M = จำนวนของเมท็อดภายในคลาส

F = จำนวนตัวแปรที่ประกาศไว้ภายในคลาส

MF = ตัวแปรที่ประกาศไว้ภายในคลาสที่ถูกเรียกใช้ในเมท็อดต่าง ๆ

$Sum(MF)$ = ผลรวมของตัวแปรที่ประกาศไว้ภายในคลาสที่ถูกเรียกใช้ในเมท็อดต่าง ๆ ของคลาสนั้น ๆ

5. แฟน-อิน [7] คือจำนวนของคลาสอื่นที่เข้ามาเรียกใช้คลาสที่กำลังพิจารณา

6. แฟน-เอ้าท์ [7] คือจำนวนของคลาสอื่นที่ถูกเรียกใช้โดยคลาสที่กำลังพิจารณา

### 2.2. เครื่องมือที่ใช้ในงานวิจัย

งานวิจัยครั้งนี้มีการนับจำนวนบรรทัดของโปรแกรม ซึ่งเครื่องมือที่ใช้คือ Code Counter Pro ซึ่งสามารถนับจำนวนบรรทัดของโปรแกรมได้หลายภาษา เช่น C, C++, Java, Delphi, Pascal, COBOL, VB, PHP, ASP, XML และ FORTRAN นอกจากนี้ Code Counter Pro ยังสามารถนับจำนวนบรรทัดของโปรแกรมแบบลอจิคอล (Logical Line of Code) คือนับเฉพาะจำนวนบรรทัดของโปรแกรมจริง ๆ เท่านั้น จะไม่นับบรรทัดคอมเมนต์และบรรทัดว่าง และแบบกายภาพ (Physical Line of Code) คือนับจำนวนบรรทัดทั้งหมดรวมทั้งบรรทัดคอมเมนต์ และบรรทัดว่าง ซึ่งงานวิจัยนี้จะใช้การนับจำนวนบรรทัดของโปรแกรมแบบลอจิคอล และเครื่องมือที่ใช้วัดความซับซ้อนของโปรแกรมคือ Jhawk ใช้ในการวัดความซับซ้อนของโปรแกรมเชิงออบเจ็กต์ ซึ่ง Jhawk จะวัดความซับซ้อนในระดับคลาส แต่ถ้าต้องการวางแผนการทดสอบในระดับโปรเจค ยังไม่มีเครื่องมือที่ใช้วัดความซับซ้อนของโปรเจคโดยตรง

## 3. วิธีการดำเนินงานวิจัย

ขั้นตอนการวิจัยมีดังต่อไปนี้

**ขั้นตอนที่ 1** การศึกษาครั้งนี้ใช้กลุ่มตัวอย่างที่เขียนด้วยภาษาจาวา จำนวน 2 กลุ่ม เพื่อเปรียบเทียบค่าความซับซ้อนของกลุ่มตัวอย่างทั้ง 2 กลุ่ม ซึ่งกลุ่มตัวอย่างกลุ่มแรก คือโอเพ่นซอสซอฟต์แวร์ เป็นกลุ่มตัวอย่างที่เก็บมาจาก http://java-source.net โดยโปรแกรมที่เก็บมานั้น ส่วนใหญ่จะเป็นโปรแกรมที่พัฒนามาจากนักวิจัย รองลงมาจะมาจากกลุ่มที่มีความสนใจในเรื่องเดียวกันและต้องการซอฟต์แวร์ออกมาใช้งาน โดยไม่ต้องการเสียค่าใช้จ่าย จึงร่วมมือกันพัฒนาซอฟต์แวร์นั้นขึ้นมา และกลุ่มตัวอย่างที่ 2 คือโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. เป็นโปรแกรมที่นักศึกษา ได้จัดทำขึ้นเพื่อเป็นโปรเจคจบก่อนสำเร็จการศึกษาในชั้นปีที่ 4 โดยนักศึกษานั้นจะเคยเรียนวิชาการเขียนโปรแกรมมาก่อนจำนวน 2 วิชา คือการเขียนโปรแกรมภาษาจาวา I และการเขียนโปรแกรมภาษาจาวา II ซึ่งจำนวน

ของโปรแกรมตัวอย่างของทั้ง 2 กลุ่มนั้นมีจำนวน 25 โปรแกรมเท่ากัน โดยใช้จำนวนบรรทัดของโปรแกรมเป็นเกณฑ์ในการเลือก เพื่อที่จะไม่ให้ขนาดของจำนวนบรรทัดของโปรแกรมของกลุ่มตัวอย่างนั้น มีความแตกต่างกันทำให้ง่ายต่อการเปรียบเทียบความซับซ้อน

**ขั้นตอนที่ 2** นำกลุ่มตัวอย่างทั้ง 2 กลุ่ม มาหาค่าเฉลี่ย ค่าสูงสุด ค่าต่ำสุด ค่ามัธยฐาน และส่วนเบี่ยงเบนมาตราฐานของตัววัดซอฟต์แวร์แต่ละตัวได้ผลดังต่อไปนี้ กลุ่มตัวอย่างที่ 1 พบว่าจำนวนบรรทัดของโปรแกรมของโอเพ่นซอร์สซอฟต์แวร์นั้น มีจำนวนบรรทัดของโปรแกรมสูงสุด 16,376 บรรทัด ต่ำสุด 833 บรรทัด และเฉลี่ยประมาณ 4,725 บรรทัด จำนวนของเมท็อดสูงสุด 1,800 เมท็อด ต่ำสุด 30 เมท็อด และเฉลี่ย 500 เมท็อด จำนวนของคลาสสูงสุด 227 คลาส ต่ำสุด 13 คลาส และเฉลี่ย 67 คลาส ดังตารางต่อไปนี้

ตาราง 1. รายละเอียดของกลุ่มตัวอย่างที่ 1

| Open Source Software | Mean | Max | Min | S.D. |
|---|---|---|---|---|
| LOC | 4,725 | 16,376 | 833 | 3,714 |
| No. of Method | 500 | 1,800 | 30 | 430 |
| No. of Class | 67 | 227 | 13 | 55 |

และกลุ่มตัวอย่างที่ 2 พบว่าจำนวนบรรทัดของโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ.นั้น มีจำนวนบรรทัดของโปรแกรมสูงสุด 17,659 บรรทัด ต่ำสุด 816 บรรทัด และเฉลี่ยประมาณ 4,793 บรรทัด จำนวนของเมท็อดสูงสุด 1,572 เมท็อด ต่ำสุด 12 เมท็อด และเฉลี่ย 258 เมท็อด จำนวนของคลาสสูงสุด 105 คลาส ต่ำสุด 2 คลาส และเฉลี่ย 30 คลาส ดังตารางต่อไปนี้

ตาราง 2. รายละเอียดของกลุ่มตัวอย่างที่ 2

| IT Students' Projects | Mean | Max | Min | S.D. |
|---|---|---|---|---|
| LOC | 4,793 | 17,659 | 816 | 3,876 |
| No. of Method | 258 | 1,572 | 12 | 331 |
| No. of Class | 30 | 105 | 2 | 28 |

**ขั้นตอนที่ 3** นำโปรแกรมตัวอย่างมาวัดความซับซ้อน โดยใช้ตัววัดซอฟต์แวร์ชนิดต่าง ๆ คือ ความซับซ้อนไซโคลมาติก ดับเบิลยูเอ็มซี อาร์เอฟซี ซีบีโอ แอลซีโอเอ็ม แฟน-อิน และ แฟน-เอ้าท์

**ขั้นตอนที่ 4** เปรียบเทียบความซับซ้อนของโอเพ่นซอร์สซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี โดยใช้กราฟเส้น มีการจับคู่กันเพื่อเปรียบเทียบความซับซ้อนของซอฟต์แวร์ โดยจะมองจากโปรแกรมตัวอย่างทั้ง 2 กลุ่มกับตัววัดซอฟต์แวร์แต่ละตัว ว่าโปรแกรมตัวอย่างไหนจะมีความซับซ้อนมากกว่ากัน จากความซับซ้อนที่วัดได้ ซึ่งตัววัดซอฟต์แวร์แต่ละตัวนั้นมีค่าเฉลี่ยที่วัดได้ของโปรแกรมตัวอย่างทั้ง 2 กลุ่ม ดังตารางต่อไปนี้

ตาราง 3. ค่าเฉลี่ยของตัววัดซอฟต์แวร์แต่ละตัวของกลุ่มตัวอย่างที่ 1

| Open Source Software | WMC | RFC | CBO | LCOM | Fan-in | Fan-out |
|---|---|---|---|---|---|---|
| Mean | 2.07 | 19.85 | 1.84 | 1.29 | 0.98 | 0.98 |

ตาราง 4. ค่าเฉลี่ยของตัววัดซอฟต์แวร์แต่ละตัวของกลุ่มตัวอย่างที่ 2

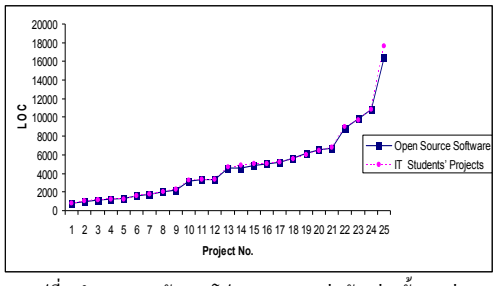
| IT Students' Projects | WMC | RFC | CBO | LCOM | Fan-in | Fan-out |
|---|---|---|---|---|---|---|
| Mean | 3.15 | 27.61 | 1.27 | 0.74 | 0.68 | 0.68 |

**ขั้นตอนที่ 5** สรุปผลที่ได้จากการเปรียบเทียบความซับซ้อนของโอเพ่นซอร์สซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

## 4. ผลการวิจัย

โอเพ่นซอสซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี (มจธ.) ถูกนำมาเปรียบเทียบกันในแง่ต่าง ๆ คือจำนวนบรรทัดของโปรแกรม จำนวนของเมท็อด จำนวนของคลาส และจากค่าตัววัดซอฟต์แวร์ต่าง ๆ ที่วัดได้ดังต่อไปนี้

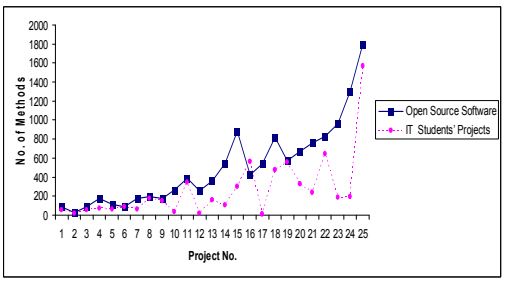
**จำนวนบรรทัดของโปรแกรม**




รูปที่ 1 จำนวนบรรทัดของโปรแกรมของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 1 พบว่าขนาดของจำนวนบรรทัดของโปรแกรมของกลุ่มโปรแกรมตัวอย่างทั้ง 2 กลุ่มนั้น มีขนาดใกล้เคียงกัน เนื่องจากต้องการให้มีจำนวนบรรทัดของโปรแกรมเท่ากัน เพื่อให้ง่ายต่อการเปรียบเทียบความซับซ้อนของซอฟต์แวร์

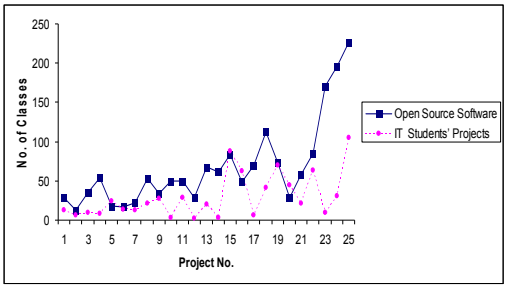
**จำนวนของเมท็อด**




รูปที่ 2 จำนวนเมท็อดของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 2 พบว่าจำนวนเมท็อดของโอเพ่นซอสซอฟต์แวร์ มีมากกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. และเมื่อขนาดโปรแกรมเพิ่มมากขึ้น จำนวนเมท็อดของโอเพ่นซอสซอฟต์แวร์ก็จะยิ่งเพิ่มมากขึ้น แสดงให้เห็นว่ายิ่งโปรแกรมขนาดใหญ่ ยิ่งต้องมีการเพิ่มจำนวนของเมท็อดให้มากขึ้น เพื่อจะทำให้โปรแกรมไม่ซับซ้อน เพราะมีการแบ่งการทำงานของโปรแกรมออกเป็นส่วน ๆ ทำให้เวลาที่ต้องการเรียกใช้โปรแกรมหรือต้องการหาข้อผิดพลาดและแก้ไขทำได้ง่ายขึ้น เนื่องจากไม่ต้องคอยตรวจสอบโปรแกรมทีละบรรทัด แต่สามารถเข้าไปแก้ไขโปรแกรมจากแต่ละเมท็อดที่มีการแบ่งลักษณะการทำงานต่าง ๆ ไว้ได้เลย

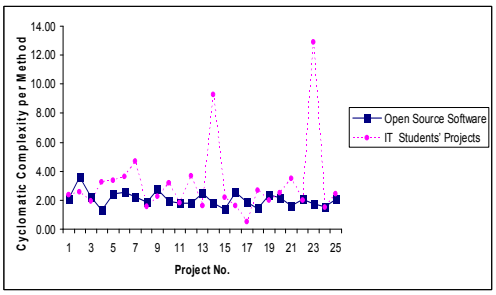
**จำนวนของคลาส**




รูปที่ 3 จำนวนคลาสของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 3 พบว่าจำนวนคลาสของโอเพ่นซอสซอฟต์แวร์มีมากกว่าจำนวนคลาสของโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. เมื่อขนาดของจำนวนบรรทัดของโปรแกรมเพิ่มมากขึ้น จำนวนคลาสก็จะเพิ่มขึ้นตามไปด้วย จึงพบว่ายิ่งมีจำนวนคลาสมากเท่าไหร่ยิ่งช่วยลดความซับซ้อนของโปรแกรมและสามารถทำการทดสอบและนำกลับมาใช้ใหม่ได้ง่ายขึ้น เนื่องจากมีการแบ่งการทำงานออกเป็นคลาส โดยมีลักษณะการทำงานที่เฉพาะเจาะจงกันไป ไม่ได้มีการเขียนคลาสยาว ๆ ลงมาเพียงคลาสเดียวแบบการเขียนโปรแกรมเชิงโครงสร้าง (Structured Programming) เพราะจะทำให้ยากต่อการทดสอบเมื่อเกิดปัญหา จึงพบว่าโปรแกรมของนักศึกษานั้นยังไม่เป็นโปรแกรมเชิงอ็อบเจกต์มากนักเนื่องจากมีจำนวนคลาสที่น้อย แต่จะถนัดการการเขียนโปรแกรมเชิงโครงสร้างมากกว่า

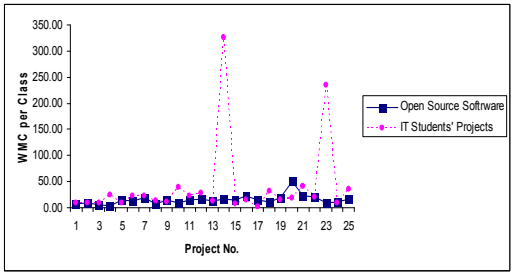
**ความซับซ้อนไซโคลมาติก**




รูปที่ 4 การเปรียบเทียบค่าความซับซ้อนไซโคลมาติกของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 4 พบว่าค่าความซับซ้อนไซโคลมาติกต่อเมท็อดของโอเพ่นซอสซอฟต์แวร์มีค่าน้อยกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. แสดงว่าโอเพ่นซอสซอฟต์แวร์นั้น มีความซับซ้อนน้อยกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี เนื่องจากจำนวนเมท็อดของโอเพ่นซอสซอฟต์แวร์มีมากกว่าทำให้มีการเขียนโปรแกรมในแต่ละเมท็อดไม่มาก เมื่อเทียบกับโปรแกรมของนักศึกษาที่มีจำนวนเมท็อดไม่มาก จึงมีการเขียนโปรแกรมทุกอย่างรวมไว้ภายในเมท็อดเดียวกัน เมื่อนำมาวัดความซับซ้อนจึงทำให้โปรแกรมของนักศึกษาส่วนใหญ่มีความซับซ้อนมากกว่าของโอเพ่นซอสซอฟต์แวร์ที่วัดกันต่อเมท็อด

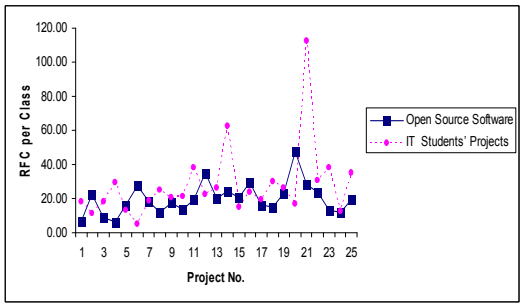
**ดับเบิลยูเอ็มซี**




รูปที่ 5 การเปรียบเทียบค่าดับเบิลยูเอ็มซีของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 5 พบว่าค่าดับเบิลยูเอ็มซีต่อคลาสของโอเพ่นซอสซอฟต์แวร์มีค่าน้อยกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. แสดงว่าโอเพ่นซอสซอฟต์แวร์นั้น มีความซับซ้อนน้อยกว่าซอฟต์แวร์โครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. เนื่องจากจำนวนคลาสของโอเพ่นซอสซอฟต์แวร์มีมากกว่าทำให้มีการเขียนโปรแกรมในแต่ละคลาสไม่มาก เมื่อเทียบกับโปรแกรมของนักศึกษาที่มีจำนวนคลาสไม่มาก จึงมีการเขียนโปรแกรมทุกอย่างรวมไว้ภายในคลาสเดียวกัน เมื่อนำมาวัดความซับซ้อนจึงทำให้โปรแกรมของนักศึกษาส่วนใหญ่มีความซับซ้อนมากกว่าของโอเพ่นซอสซอฟต์แวร์ที่วัดกันต่อคลาส

**อาร์เอฟซี**

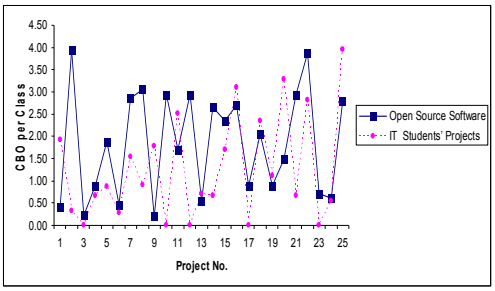



รูปที่ 6 การเปรียบเทียบค่าอาร์เอฟซีของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 6 พบว่าค่าอาร์เอฟซีต่อคลาสของโอเพ่นซอสซอฟต์แวร์มีค่าน้อยกว่าซอฟต์แวร์โครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. ซึ่งอาร์เอฟซีเป็นการวัดความซับซ้อนของซอฟต์แวร์ ที่มองในเรื่องของการเรียกใช้กันระหว่างคลาส เกิดจากจำนวนเมท็อดที่มีการสร้างไว้ใช้งานภายในคลาสรวมกับจำนวนเมท็อดของคลาสอื่นที่ถูกเรียกใช้โดยเมท็อดต่าง ๆ

ในคลาสนั้น ๆ แสดงว่าค่าอาร์เอฟซีต้องมีค่าน้อย เพื่อที่จะมีประสิทธิภาพในการตอบสนองต่อการร้องขอ จากรูปพบว่าโอเพ่นซอสซอฟต์แวร์มีค่าอาร์เอฟซีต่อคลาสน้อยกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. ดังนั้นการตอบสนองต่อการเรียกใช้จะดีกว่าโปรแกรมของนักศึกษา

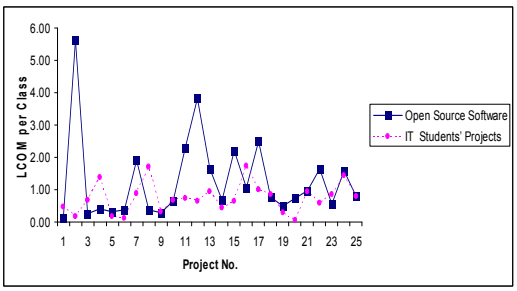
**ซีบีโอ**




รูปที่ 7 การเปรียบเทียบค่าซีบีโอของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 7 พบว่าค่าซีบีโอต่อคลาสของโอเพ่นซอสซอฟต์แวร์ มีค่ามากกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. ซึ่งซีบีโอเป็นการวัดความซับซ้อนของซอฟต์แวร์ โดยมองจากการคุปปลิ้ง (Coupling) กันระหว่างคลาส คือเป็นการพึ่งพาคลาสอื่น หากซีบีโอมีค่าสูงจะไม่ดีเพราะว่าต้องไปเกี่ยวข้องกับคลาสอื่น ๆ สรุปว่าโปรแกรมของนักศึกษานั้นมีการเขียนโปรแกรมมีการคุปปลิ้งน้อยกว่าของโอเพ่นซอสซอฟต์แวร์ ทำให้ได้ค่าซีบีโอที่ต่ำ ซึ่งตรงกับหลักการในการเขียนโปรแกรมเชิงอ็อบเจกต์ที่ว่าต้องมีโคฮีชั่นสูง (Strong Cohesion) คือให้สัมพันธ์ภายในของคลาสหรือโมดูลต่าง ๆ เป็นไปอย่างแนบแน่นและคุปปลิ้งต่ำ (Loosely Couple) คือให้ความสัมพันธ์ระหว่างคลาสหรือโมดูลต่าง ๆ เป็นไปอย่างหลวม ๆ ทำให้เวลาต้องการแก้ไขหรือปรับปรุงคลาสหรือโมดูลใด ๆ แล้ว สามารถทำได้เลย จะไม่กระทบกับคลาสหรือโมดูลอื่น ๆ ซึ่งทำให้ไม่ต้องตามไปแก้ไข ช่วยให้ประหยัดเวลา ลดภาระในการเขียนโปรแกรม การทดสอบการบำรุงรักษาทำได้ง่ายและสามารถนำส่วนประกอบต่าง ๆ เช่นเมท็อดและคลาสต่าง ๆ นำกลับมาใช้ใหม่ได้

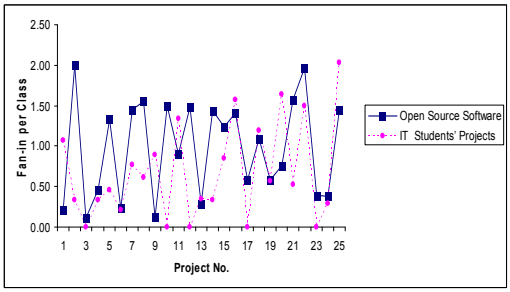
**แอลซีโอเอ็ม**




รูปที่ 8 การเปรียบเทียบค่าแอลซีโอเอ็มของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 8 พบว่าค่าแอลซีโอเอ็มต่อคลาสของโอเพ่นซอสซอฟต์แวร์มีค่ามากกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. ซึ่งแอลซีโอเอ็มเป็นการวัดความซับซ้อน โดยมองจากความสัมพันธ์ระหว่างเมท็อดและจำนวนตัวแปรที่มีการประกาศไว้ภายในคลาส หากค่าแอลซีโอเอ็มมีค่าสูงจะไม่ดี เนื่องจากมีการประกาศตัวแปรไว้ภายในคลาสแต่ไม่มีการเรียกใช้ จึงทำให้ค่าแอลซีโอเอ็มมีค่าสูง ซึ่งเป็นแค่คลาสที่สร้างขึ้นมาเพื่อเก็บ Data Structure เท่านั้น หากค่าแอลซีโอเอ็มต่ำแสดงว่าเมท็อดนั้นมีการไปเรียกใช้ตัวแปรที่ประกาศไว้ภายในคลาส ปกติค่าแอลซีโอเอ็มจะมีค่าอยู่ระหว่าง 0 ถึง 2 หากค่าแอลซีโอเอ็มมีค่าเท่ากับ 0 แสดงว่าทุกเมท็อดมีการเรียกใช้ทุกตัวแปร และหากค่าแอลซีโอเอ็มมีค่าเท่ากับ 2 แสดงว่ามีจำนวนเมท็อด 2 เมท็อด มีจำนวนตัวแปร 1 ตัวขึ้นไป แต่ไม่มีการเรียกใช้ตัวแปร หากค่าแอลซีโอเอ็มมีค่าอยู่ระหว่าง 0 และ 2 แสดงว่ามีจำนวนเมท็อดตั้งแต่ 2 ขึ้นไป มีจำนวนตัวแปรตั้งแต่ 1 ตัวขึ้นไป และจากรูปพบว่าค่าแอลซีโอเอ็มต่อคลาสของโอเพ่นซอสซอฟต์แวร์บางโปรแกรมมีค่ามากกว่า 2 เนื่องจากเหตุผลดังนี้ คือมีจำนวนเมท็อด 1 เมท็อด มีจำนวนตัวแปรตั้งแต่ 1 ตัวขึ้นไป ไม่มีการเรียกใช้ตัวแปร ค่าแอลซีโอเอ็มจะมีค่าเท่ากับจำนวนตัวแปรที่มีการประกาศไว้ภายในคลาส หรือหากมีจำนวนเมท็อด 1 เมท็อด มีจำนวนตัวแปรตั้งแต่ 1 ตัวขึ้น มีการเรียกใช้ตัวแปร ค่าแอลซีโอเอ็มจะมีค่าเท่ากับจำนวนตัวแปรที่มีการประกาศไว้ภายในคลาสลบด้วยตัวแปรที่ประกาศไว้ภายในคลาสที่ถูกเรียกใช้ในเมท็อดต่าง ๆ และในกรณีที่ไม่มีเมท็อด ไม่มีตัวแปรที่ประกาศไว้ภายในคลาส เป็นคลาสว่าง แสดงว่าไม่มีค่าแอลซีโอเอ็ม หรือในกรณีที่ไม่มีเมท็อด แต่มีตัวแปรที่ประกาศไว้ภายในคลาสตั้งแต่ 1 ตัวขึ้นไป ค่าแอลซีโอเอ็มจะมีค่าเท่ากับจำนวนตัวแปรที่มีการประกาศไว้ภายในคลาส จึงเป็นเหตุผลที่ทำให้ค่าแอลซีโอเอ็มต่อคลาสมีค่าเกิน 2 ได้ จากการเปรียบเทียบพบว่าค่าแอลซีโอเอ็มต่อคลาสของโอเพ่นซอสซอฟต์แวร์มีค่ามากกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. เนื่องจากมีจำนวนคลาสที่มีขนาดเล็กและเป็นคลาสที่สร้างขึ้นมาเพื่อเก็บ Data Structure ทำให้มีค่าแอลซีโอเอ็มสูงเมื่อเปรียบเทียบกันต่อคลาสกับโปรแกรมของนักศึกษาที่มีคลาสขนาดที่ใหญ่กว่า ทำให้มีการเรียกใช้ตัวแปรมากกว่า จึงทำให้มีค่าแอลซีโอเอ็มของโปรแกรมของนักศึกษาต่ำกว่าของโอเพ่นซอสซอฟต์แวร์ ทำให้มีความซับซ้อนมากกว่าของโอเพ่นซอสซอฟต์แวร์

**แฟน-อิน**

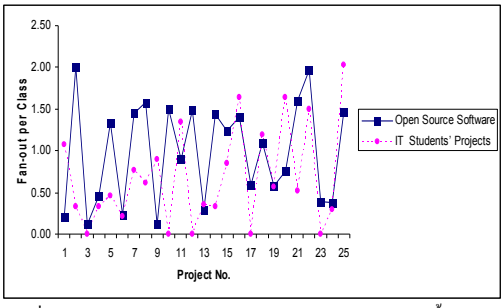



รูปที่ 9 การปรียบเทียบค่าของแฟน-อิน ของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 9 พบว่าค่าแฟน-อินต่อคลาส ของโอเพ่นซอสซอฟต์แวร์มีค่ามากกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. แสดงว่าโอเพ่นซอสซอฟต์แวร์มีการออกแบบคลาสที่ดี เนื่องจากสามารถให้

คลาสอื่นเข้ามาเรียกใช้คลาสได้อีก จะเป็นประโยชน์ในแง่ของการนำกลับมาใช้ใหม่

**แฟน-เอ้าท์**




รูปที่ 10 การเปรียบเทียบค่าของ แฟน-เอ้าท์ ของกลุ่มตัวอย่างทั้ง 2 กลุ่ม

จากรูปที่ 10 พบว่า ค่าแฟน-เอ้าท์ต่อคลาส ของโอเพ่นซอสซอฟต์แวร์ มีค่ามากกว่าโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มจธ. แสดงว่าโอเพ่นซอสซอฟต์แวร์นั้น มีการเขียนโปรแกรมที่ต้องพึ่งพาคลาสภายนอกเพื่อให้งานสมบูรณ์ แต่ต่างจากโปรแกรมนักศึกษาที่พยายามเขียนโปรแกรมให้จบภายในคลาสของตัวเอง ไม่เน้นการพึ่งพาคลาสอื่น สังเกตุได้จากจำนวนเมท็อดและจำนวนคลาสของโปรแกรมที่มีจำนวนไม่มากทั้ง ๆ ที่จำนวนบรรทัดของโปรแกรมมีจำนวนใกล้เคียงกับของโอเพ่นซอสซอฟต์แวร์ แต่โอเพ่นซอสซอฟต์แวร์กลับมีจำนวนเมท็อดและคลาสมากกว่า

## 5. สรุปผลการวิจัย

จากการศึกษาความซับซ้อนของโปรแกรมภาษาจาวา โดยใช้ตัววัดซอฟต์แวร์ชนิดต่าง ๆ เพื่อเปรียบเทียบความซับซ้อนของกลุ่มตัวอย่าง 2 กลุ่ม คือโอเพ่นซอสซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี จำนวน 25 โปรแกรม เท่ากัน โดยใช้จำนวนบรรทัดของโปรแกรมเป็นเกณฑ์ในการเลือก โดยใช้ตัววัดซอฟต์แวร์ทั้งหมด 7 ตัว คือความซับซ้อนไซโคลมาติก ดับเบิลยูเอ็มซี อาร์เอฟซี ซีบีโอ แอลซีโอเอ็ม แฟน-อิน และ แฟน-เอ้าท์ จากผลลัพธ์ของค่าความซับซ้อนที่วัดได้ นำมาเปรียบเทียบความซับซ้อนของโอเพ่นซอสซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ทำให้พบจุดด้อยของโปรแกรมของนักศึกษาในด้านต่าง ๆ เช่น การแบ่งเมท็อดและคลาสของโปรแกรมของนักศึกษายังมีไม่มากนัก และนิยมการเขียนโปรแกรมเชิงโครงสร้างอยู่ ส่วนเรื่องของความซับซ้อนนั้น โปรแกรมของนักศึกษาจะมีความซับซ้อนมากกว่าของโอเพ่นซอสซอฟต์แวร์ ทั้งระดับของเมท็อดและระดับคลาส เนื่องจากจำนวนของเมท็อดและจำนวนของคลาสของโปรแกรมของนักศึกษานั้นมีจำนวนไม่มาก จึงทำให้เนื้อหาต่าง ๆ ของโปรแกรมมารวมกันอยู่ในเมท็อดและคลาส เพียงไม่กี่คลาส ดังนั้นเมื่อนำมาเปรียบเทียบกับของโอเพ่นซอสซอฟต์แวร์ที่มีจำนวนเมท็อดและคลาสมากกว่า เนื้อหาของโปรแกรมกระจายไปตามคลาสต่าง ๆ ทำให้ความซับซ้อนที่วัดได้ของแต่ละเมท็อดและคลาส ของโอเพ่นซอสซอฟต์แวร์มีความซับซ้อนน้อยกว่าโปรแกรมของนักศึกษาเมื่อเทียบกันต่อเมท็อดและคลาส ส่วนจุดเด่นของโปรแกรมของนักศึกษา คือสามารถเขียนโปรแกรมที่มีโคฮีชั่นสูงและมีและคุปปลิ้งต่ำ ทำให้ไม่ต้องมีการพึ่งพาคลาสอื่น มีความสมบูรณ์ของโปรแกรมอยู่ภายในตัวเอง ทำให้เวลาต้องการแก้ไขหรือปรับปรุงคลาสหรือโมดูลใด ๆ แล้ว สามารถทำได้เลย จะไม่กระทบกับคลาสหรือโมดูลอื่น ๆ ซึ่งทำให้ไม่ต้องตามไปแก้ไข ช่วยให้ประหยัดเวลา ลดภาระในการเขียนโปรแกรม การทดสอบการบำรุงรักษาทำได้ง่ายขึ้น

จากผลการวิจัยทำให้ทราบว่าโปรแกรมของนักศึกษานั้นมีจุดด้อยและจุดเด่นในด้านใดบ้าง ซึ่งจะเป็นประโยชน์ในด้านของการบำรุงรักษาและการนำโปรแกรมไปพัฒนาต่อ เนื่องจากเราสามารถทราบว่าโปรแกรมที่พัฒนาขึ้นมานั้นมีความซับซ้อนในด้านใดบ้างและสามารถทราบข้อด้อยของผู้พัฒนาซอฟต์แวร์เองว่ามีข้อด้อยในการพัฒนาตรงจุดใดบ้าง เพื่อปรับปรุงแก้ไขให้ดีขึ้น ผู้วิจัยหวังเป็นอย่างยิ่งว่าผลที่ได้จากการวัดความซับซ้อนของโปรแกรมและการเปรียบเทียบความซับซ้อนของโอเพ่นซอสซอฟต์แวร์กับโปรแกรมโครงงานของนักศึกษาชั้นปีที่ 4 คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรีนั้น จะมีประโยชน์ต่อนักศึกษาและอาจารย์ผู้สอนในการนำข้อมูลมาใช้เพื่อวางแผนในการสอนวิชาการเขียนโปรแกรมภาษาจาวาต่อไป

## เอกสารอ้างอิง


[1] CodeCounterPro. [Online]. เข้าถึงได้จาก: *http://www.sharewareconnection.com/code-counter-pro.htm*, ค้นเมื่อ 6 กุมภาพันธ์ 2552.

[2] VirtualMachinery. [Online]. เข้าถึงได้จาก: *http://www.virtualmachinery.com/jhawkprod.htm*, ค้นเมื่อ 13 ธันวาคม 2551.

[3] Chidamber, S.R. and Kemerer, C.F., "A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering,* Vol. 20, No. 6, Cambridge, USA, 1994.

[4] Jung, H.W., Pivka, M. and Kim, J.Y., "An Empirical Study of Complexity Metrics in COBOL Programs", *The Journal of Systems and Software*, Vol.51, Issue 2, 2000.

[5] Thomas J. McCabe, "A Complexity Measure", *IEEE Transaction on Software Engineering*, Vol. SE-2, No. 4, December, 1976.

[6] Doake, J. and Duncan, I., "Amber Metrics for the Testing & Maintenance of Object-Oriented Designs", *IEEE Computer Society Technical Council on Software Engineering (TCSE)*, Florence, Italy, 1998.

[7] Linda H. Rosenberg and Lawrence E. Hyatt, "Software Quality Metrics for Object – Oriented Environments", Goddard Space Flight Center, Greenbelt, USA, 1997.