การเปรียบเทียบวิธีในการแปลงชุดคำสั่งจากภาษาจาวาเป็นคอทลิน สำหรับการพัฒนาโมบายแอปพลิเคชัน

นฤพล สุวรรณวิจิตร

สาขาวิศวกรรมซอฟต์แวร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏนครปฐม
naruapon@webmail.npru.ac.th

บทคัดย่อ

บทความนี้มีวัตถุประสงค์เพื่อทำการเปรียบเทียบแนวทางในการปรับปรุงชุดคำสั่งของแอปพลิเคชันจากภาษาจาวาเป็นภาษาคอทลิน และทราบถึงผลลัพธ์และปัญหาในแต่ละทางเลือก โดยทำการสร้างแอปพลิเคชัน และทำการเปรียบเทียบวิธีการแปลงชุดคำสั่งของแอปพลิเคชันและทำการเปรียบเทียบผลในแต่ละทางเลือก ผู้วิจัยได้กำหนดวิธีการแปลงชุดคำสั่งไว้ 3 แนวทาง ดังนี้ 1) การใช้เครื่องมือในการแปลงไฟล์ทีละไฟล์ 2)การเขียนชุดคำสั่งใหม่ทีละไฟล์โดยใช้โปรเจ็คเดิม 3) การเขียนชุดคำสั่งใหม่ทีละไฟล์โดยการสร้างโปรเจ็คใหม่ ซึ่งในแต่ละแนวทางเลือกแอปพลิเคชันจะต้องมีฟังก์ชันการทำงานเหมือนเดิม จากผลการวิจัยทำให้ทราบถึง ปัญหาและปัจจัยที่ควรใช้ในการเลือกแนวทางการแปลงชุดคำสั่ง ซึ่งเป็นปัจจัยสำคัญในการเลือกในการปรับเปลี่ยนชุดคำสั่งจากภาษาจาวาไปเป็นภาษาคอทลิน

คำสำคัญ: *การแปลงชุดคำสั่ง ภาษาโปรแกรมคอมพิวเตอร์ การพัฒนาโมบายแอปพลิเคชัน ขนาดของซอตฟ์แวร์*

A Comparative method in moblie application code conversion between Java and Kotlin

NARUAPON SUWANWIJIT
Software Engineeing, Faculty of science and technology
Nakhon Pathom Rajabhat University
naruapon@webmail.npru.ac.th

Abstract

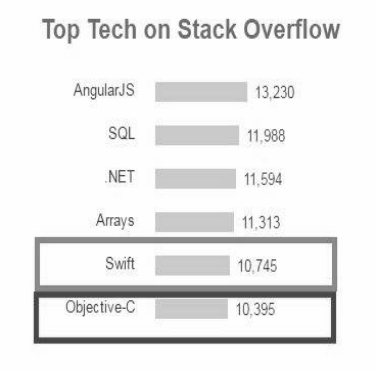
This paper aims to compare methods for the improvement of the application's code conversion from Java to Kotlin, uncovering results and problems regarding each technique. An application had been created. And by applying 3 different ways of converting the Java code conversation to Kotlin, each method had been then compared in terms of its effectiveness. Those 3 ways comprised: 1) conversion with a tool 2) conversion by creating a new file with an old project, and 3) conversion by creating a new file with a new project. However, the functions of the application must remain the same. Findings from this research study revealed problems and factors that can have an impact on the selection of code conversation from Java to Kotlin

Keywords: *code conversion, computer programming, mobile application development, software sizing*

1. บทนำ

การใช้งานอินเตอร์เน็ตผ่านมือถือ ในปัจจุบันพบว่ามีสัดส่วนที่เพิ่มขึ้นเมื่อเทียบกับพีซี เป็นผลมาจากปัจจุบันราคาและความสะดวกต่อการพกพาสมาร์ทโฟน ซึ่งในแต่ละสมาร์ทโฟนจะต้องมีระบบปฏิบัติการอยู่ภายใน จำนวนระบบปฏิบัติการที่ใช้งานเมื่อเทียบกับทั้งหมดส่วนใหญ่เป็นระบบปฏิบัติการแอนดรอยด์ และยังคงเป็นเช่นนี้ต่อไปอีกนาน เพราะว่าแอนดรอยด์เป็นระบบเปิด ที่ค่ายผู้ผลิตมือถือหลายรายเลือกใช้ ต่างจากอุปกรณ์ของบริษัทแอปเปิล International Data Corporation (IDC) ได้คาดการณ์ว่าการเติบโตของระบบปฏิบัติการบนสมาร์ทโฟน 4 ปี ตั้งแต่ปี 2019 – 2022 ที่ใช้ระบบปฏิบัติการแอนดรอยส์เฉลี่ยอยู่ที่ 86.7% และระบบปฏิบัติการ iOS อยู่ที่ 13.2%

เนื่องจากเทคโนโลยีมีการเปลี่ยนแปลงไปไม่ใช่เพียงแค่ในส่วนของฮาร์ดแวร์ และซอฟต์แวร์ ซึ่งในส่วนของภาษาโปรแกรมที่ใช้ในการพัฒนาก็เช่นเดียวกัน เดิมทีในการพัฒนาแอปดรอย์แอปพลิเคชัน นั้นต้องใช้เครื่องมือ Eclipe และ ภาษาจาวา ในการพัฒนาแอปพลิเคชัน ต่อมากูเกิ้ลได้เปิดตัวเครื่องมือ Android Studio 1.0 ในปี 2014 และได้เสนอทางเลือกของภาษาในการพัฒนานอกจากภาษาจาวา อีกหนึ่งภาษาคือคอทลิน ในปี 2017 ถูกรวมเป็นส่วนหนึ่งของ Android Studio 3.0 เหตุการณ์แบบนี้ก็เคยขึ้นกับการพัฒนาแอปพลิเคชันทางฝั่งแอปเปิลที่มีการเพิ่มภาษา Swift เป็นอีกทางเลือกหนึ่งในการพัฒนา ซึ่งมีมาตั้งแต่ Xcode เวอร์ชัน 6 ปัจจุบันผู้ที่พัฒนาแอปพลิเคชันก็ใช้ภาษา Swift ในการสร้างแอปพลิเคชัน พบว่าจำนวนประเด็นคำถามที่มีอยู่ในเว็บไซต์ Stack Overflow ระหว่าง Objective-C กับ Swift ที่ Swift มีมากกว่า ดังภาพที่ 1



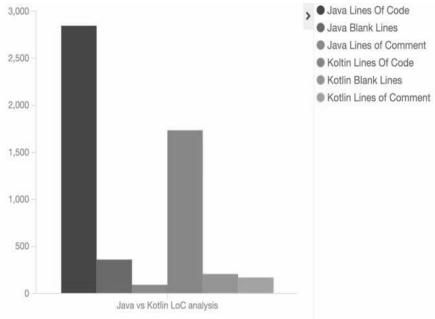
ภาพที่ 1: จำนวนประเด็นคำถามในเว็บ Stack Overflow ในปี 2019

ที่มา: https://hackernoon.com/

มุมมองของนักพัฒนาต่อการใช้ภาษาคอทลินแทนจาวา มีดังนี้

1. ก่อนที่จะปล่อยให้สามารถใช้งานภาษาคอทลินมีหลายขั้นตอนกว่าจะออกมาเป็นเวอร์ชันที่ 1.0 ปัจจุบัน ณ ปี 2020 เวอร์ชันที่ 1.3
2. ทำให้การพัฒนาแอปพลิเคชันแอนดรอยด์ง่ายขึ้น เนื่องจากเป็นภาษาโปรแกรมสมัยใหม่
3. คอทลินมีไวยากรณ์ที่ง่ายกว่าภาษาจาวา ทำให้ช่วยลดข้อผิดพลาดหรือข้อบกพร่องของชุดคำสั่ง
4. ชุดคำสั่งของภาษาคอทลินมีความปลอดภัยมากกว่าภาษาอื่น
5. ชุดคำสั่งของภาษาคอทลินมีจำนวนบรรทัดที่น้อยกว่าในการเขียนฟังก์ชันการทำงานเดียวกัน ทำให้การบำรุงรักษาและการทำความเข้าใจได้ดีกว่า

ในการทำการสร้างแอปพลิเคชันโดยกำหนดให้ฟังก์ชันที่เท่ากัน ซึ่งการพัฒนาระหว่างการใช้ภาษาจาวากับภาษาคอทลินพบว่า ภาษาคอทลินมีจำนวนบรรทัด น้อยกว่าภาษาจาวาอยู่ประมาณ 40 เปอร์เซ็นต์ ดังภาพที่ 2



ภาพที่ 2 : เปรียบเทียบจำนวน LOC ของภาษาจาวาเมื่อเทียบกับภาษา Kotlin

ที่มา: https://eng.uber.com/

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 งานวิจัยที่เกี่ยวข้อง

(Subham Bose et al., 2018: 41-45) ทำการการวิจัยเชิงอธิบาย ในการเปรียบเทียบชุดคำสั่งระหว่างภาษาจาวาและภาษาคอทลิน ในประเด็นต่างๆ ดังนี้ Extension Function, Checked Exception, Constructors, Null Safety, Lazy-Loading และได้ข้อสรุปถึงข้อดีข้อเสียของแต่ละภาษา ในภาษาจาวามีข้อดีในแง่ของแหล่งชุมชนของนักพัฒนาที่มีจำนวนมาก ส่วนภาษาคอทลินมีข้อดีที่เป็นจุดเด่นในแง่ของการเพิ่มผลิตภาพของการเขียนชุดคำสั่งซึ่งได้ทำการเปรียบเทียบในการสร้างคลาสในภาษาจาวาและภาษาคอทลิน พบว่าภาษาคอทลินมีจำนวนบรรทัดที่น้อยกว่า โดยที่การทำงานของคลาสเหมือนกัน

Java

```
public class Product{
  String productID;
  String productName;
  double price;
  public Product(String productID, String productName, double price){
   this.productID = productID;
   this.productName = productName;
   this.price = price;
  }
  public String toString(){
    return productID + " " + productName + " , Price = " + price;
  }
}
```

Kotlin

```
public class Product(val productID:String, val productName:String, val price: Double){
    override fun toString() = "$productID $productName , Price = $price"
}
```

(Dony George et al., 2010: 0975) ได้ทำการแปลงจากภาษา C++ ไปเป็นภาษา Java ได้พบปัญหาและวิธีการแก้ปัญหาต่างๆ ในการแปลงภาษา อาทิ เช่น 1. Pointers 2. Pre-processor 3. Destructors 4. Friend Function 5. Operator Overloading เป็นต้น

(A.A. Terekhov and C. Verhoef , 2000: 111 – 124) ได้ศึกษาทำการแปลงภาษาหนึ่งไปยังอีกภาษาหนึ่ง ดังต่อไปนี้ 1. COBOL to Visual Basic 2. COBOL to C 3. COBOL to Java 4. OS/VS COBOL to VS COBOL II พบว่าการแปลงชุดคำสั่งแบบอัตโนมัติส่วนใหญ่จะล้มเหลว ซึ่งการแปลงเป็นสิ่งที่ยากและยากกว่าที่คิด

ผู้วิจัยได้นำวิธีในการแปลงทั้ง 2 วิธีมาใช้ในการศึกษาดำเนินงาน คือ แบบที่อาศัยเครื่องมือในการแปลงชุดคำสั่งและ การแปลงชุดคำสั่งโดยอาศัยความรู้ของนักพัฒนา โดยใช้ภาษาโปรแกรมจาวาและคอทลินซึ่งเป็นที่นิยมในปัจจุบัน

2.2 Mobile Application

2.2.1 Native application

การพัฒนาแบบเนทีฟแอปพลิเคชันเป็นการพัฒนาโดยใช้ภาษาโปรแกรมคอมพิวเตอร์ที่มีลักษณะเฉพาะเจาะจงสำหรับระบบปฏิบัติการนั้นๆ เช่น การใช้ภาษาจาวาหรือคอทลินในการพัฒนาแอนดรอยส์แอปพลิเคชัน การใช้ภาษา Objective Cหรือ ภาษา Swift ในการพัฒนา iOS แอปพลิเคชัน ซึ่งในการพัฒนาแอปพลิเคชันเพื่อให้ทำงานได้ทั้งสองแพลตฟอร์มจำเป็นต้องสร้างที่เก็บชุดคำสั่งจำนวน 2 ชุด ในแต่ละภาษา การแก้ไขก็ต้องทำทั้งสองชุดเช่นกัน ส่วนประสิทธิภาพนั้นโดยทั่วไป Native จะมีประสิทธิภาพมากกว่า Hybrid เช่น ความเร็ว และการเข้าถึงไลบรารีใหม่ๆ เมื่อแพลตฟอร์มมีการอัพเดท

2.2.2 Hybrid application

ในส่วนของการพัฒนาแบบไฮบริดแอปพลิเคชัน เป็นการพัฒนาที่อาศัยการแสดงผลในส่วนของเว็บวิว เครื่องมือที่ใช้ในการพัฒนาไม่ได้ออกแบบมาโดยตรงจากบริษัทเจ้าของแพลตฟอร์มอย่างกูเกิ้ล และแอปเปิล ต่างจากการพัฒนาแบบเนทีฟแอปพลิเคชันที่มี Android Studio สำหรับการพัฒนาแอนดรอย์แอปพลิเคชัน และ Xcode สำหรับการพัฒนาไอโอเอสแอปพลิเคชัน การพัฒนาไฮบริดแอปพลิเคชัน โดยทั่วไปใช้ภาษา HTML, CSS และ Javascript ซึ่งมีเครื่องมือและเฟรมเวิร์ค จากผู้ผลิตรายอื่นๆ เช่น Apace Cordova, Ionic, React Native เป็นต้น แล้วทำการแปลงให้เป็นเนทีฟแอปพลิเคชัน การพัฒนาแบบนี้ทำได้รวดเร็วเนื่องจากชุดคำสั่งเป็นชุดคำสั่งเดียวกันทั้งสองแพลตฟอร์ม

2.3 การพัฒนาแอนดรอยด์แอปพลิเคชัน

2.3.1 เครื่องมือที่สำหรับการพัฒนาแอปพลิเคชันแบบเนทีฟ

2.3.1.1 Android Studio และ Android SDK

2.3.1.2 Java SE Development Kit (JDK)

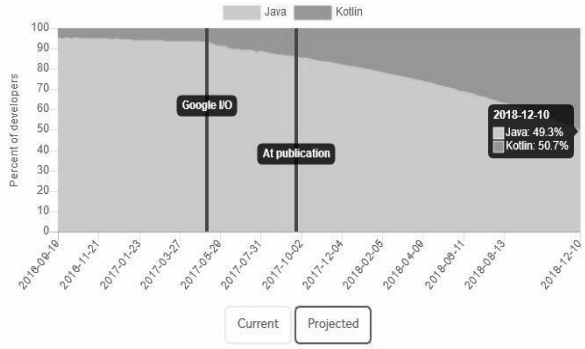
2.3.2 ทักษะทางด้านภาษาโปรแกรมคอมพิวเตอร์

2.3.2.1 Java

2.3.2.2 Kotlin

2.4 การแปลงชุดคำสั่ง

การแปลงชุดคำสั่งเป็นการเปลี่ยนชุดคำสั่งเดิมเป็นชุดคำสั่งใหม่ เพื่อรองรับเทคโนโลยีที่เปลี่ยนแปลงไป โดยผู้วิจัยได้แนวทางในการแปลงชุดคำสั่งไว้ 3 แนวทาง เนื่องจากภาษาคอทลินภาษาใหม่ที่สามารถทำงานร่วมกันกับภาษาเดิมอย่างภาษาจาวาได้ เช่น การเรียกใช้ไลบราลีเดิม และยังสามารถที่จะสร้างไฟล์ทั้งสองภาษาในโปรเจ็คเดียวกันได้ ซึ่งเป็นคุณสมบัติที่เรียกว่า Java Interoperability เหตุที่ต้องทำการแปลงชุดคำสั่ง เนื่องจากนักพัฒนามีแนวโน้มการใช้งานภาษาคอทลินแทนภาษาจาวาที่เพิ่มขึ้น ดังภาพที่ 3 ดังนั้นเวลาที่เกิดปัญหาในการพัฒนาแอปพลิเคชัน การค้นหาวิธีการแก้ปัญหาในอนาคตจะมีที่เป็นภาษาจาวาน้อยลงเมื่อเทียบกับภาษาคอทลิน



ภาพที่ 3: สัดส่วนของนักพัฒนาที่ใช้ภาษาคอทลินในการพัฒนาแอปพลิเคชันในปี 2018

ที่มา: https://www.bleepingcomputer.com/

จากภาพที่ 3 จะเห็นได้ว่าการเติบโตของภาษาคอทลินเพิ่มขึ้นถึง 50.7% ในเดือนธันวาคม ปี 2018 จากเดิมในช่วงปี 2016 จะเห็นว่ามีการพัฒนาแอนดรอยส์โดยใช้ภาษาจาวามากกว่า 90 เปอร์เซ็นต์ และการพัฒนาแอนดรอยส์โดยใช้ภาษาคอทลินไม่ถึง 10 เปอร์เซ็นต์

2.5 ขนาดของซอฟต์แวร์

การวัดขนาดของซอฟต์แวร์ สามารถทำได้ด้วยวิธีการต่างๆ อาทิ เช่น การวัดจากจำนวนบรรทัด การวัดจากฟังก์ชันพอยต์ SLOC (Source Line Of Code) หรือ LOC (Line Of Code) คือ จำนวนบรรทัดของชุดคำสั่ง ซึ่งบ่งบอกถึงขนาดของซอฟต์แวร์ได้ แต่ไม่นับชุดคำสั่งที่ระบบสร้างขึ้นมาให้ และพวกคอมเมนต์ต่างๆ และ FP (Function Point) คือ หน่วยในการวัดจำนวนของฟังก์ชันการทำงานของซอฟต์แวร์ของระบบที่สร้างขึ้นมา สามารถแปลงเป็นจำนวนบรรทัดที่จะต้องใช้โดยประมาณ เพื่อบ่งบอกถึงขนาดของซอฟต์แวร์ได้ เช่น หาค่า FP ได้เท่ากับ 200 เมื่อทำการเขียนโปรแกรมภาษา Java (ค่าเฉลี่ยของชุดคำสั่งภาษา Java ต่อ 1 FP มีค่าเท่ากับ 53) ดังนั้น SLOC เฉลี่ยจะเท่ากับ 200x53 = 10600

3. วิธีการดำเนินการ

3.1 ขั้นตอนการดำเนินการ

ในการดำเนินการ ผู้วิจัยได้ทำการออกแบบการดำเนินการดังต่อไปนี้

1. ทำการออกแบบฟังก์ชันการทำงานของโมบายแอปพลิเคชัน และพัฒนาแอปพลิเคชันด้วยภาษาจาวา โดยมีฟังก์ชันการทำงานตามที่กำหนดเอาไว้ ในการพัฒนาใช้เครื่องมือ Android Studio เวอร์ชัน 3.5.3 และ Api เวอร์ชัน 29 (Android Q)
2. ทำการแปลงชุดคำสั่ง ดำเนินการทดลองกับแอปพลิเคชัน Vocabulary Master Lite ซึ่งผู้วิจัยได้กำหนดแนวทางในการแปลงชุดคำสั่ง 3 แนวทางดังนี้
    1. การแปลงชุดคำสั่งโดยใช้เครื่องมือ – A1
    2. การแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่แต่ใช้โปรเจ็คเดิม – A2
    3. การแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่โดยการสร้างโปรเจ็คใหม่ – A3

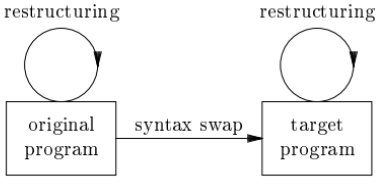
3.2 วิธีในการแปลงชุดคำสั่ง

3.2.1 การแปลงชุดคำสั่งโดยใช้เครื่องมือ – A1

เนื่องจาก Android Studio มีเครื่องมือที่ช่วยในการแปลงจากภาษาจาวาเป็นภาษาคอทลิน เพราะว่าภาษาคอทลินได้รับการสนับสนุนจากกูเกิ้ลอย่างเป็นทางการ แต่ก่อนจะดำเนินการจำเป็นต้องติดตั้งปลั๊กอิน และ ต้องมีการกำหนดเวอร์ชันของภาษาคอทลินที่จะทำการแปลง การแปลงจะแปลงเฉพาะไฟล์ที่มีนามสกุล .java หลังจากแปลงแล้วไฟล์จะถูกเปลี่ยนเป็นนามสกุล .kt

3.2.2 การแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่แต่ใช้โปรเจ็คเดิม – A2

ดำเนินการในลักษณะนี้ โดยการเลือกแปลงชุดคำสั่งขึ้นมาทีละไฟล์ ทำการลบไฟล์ที่จะทำการแปลงและสร้างไฟล์ใหม่ขึ้นมา หลังจากที่สร้างไฟล์และเขียนชุดคำสั่งขึ้นมาใหม่ รันดูผลการเปลี่ยนแปลงทีละไฟล์และทดสอบการทำงานของฟังก์ชันว่ายังทำงานได้ถูกต้องหรือไม่ และทำไปจนครบ นักพัฒนาต้องมีองค์ความรู้ของภาษาจาวาและคอทลิน และมีความรู้ในการพัฒนาแอนดรอยด์แอปพลิเคชันพอสมควร ซึ่งกระบวนการแปลงแสดงในภาพที่ 3



ภาพที่ 3: กระบวนการแปลงจากภาษาหนึ่งไปยังอีกภาษาหนึ่ง

ที่มา: A.A. Terekhov, C. Verhoef. (2000)

3.2.3 การแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่โดยการสร้างโปรเจ็คใหม่ – A3

รูปแบบนี้จะต้องสร้างไฟล์ต่างๆ ขึ้นมาใหม่จำนวนมาก ไม่เพียงแค่ไฟล์ภาษาที่นามสกุล .java เช่น ไฟล์ที่ใช้ในการออกแบบหน้าจอ, ไฟล์ทรัพยากรต่างๆ ของระบบ ผู้พัฒนาต้องมีองค์ความรู้และทักษะของภาษาจากวาและภาษาคอทลิน และกระบวนการสร้างแอนดรอยด์แอปพลิเคชันมากกว่าแนวทาง A2 และ A1

4. ผลการดำเนินงาน

4.1 ปัญหาที่พบ

4.1.1 ปัญหาการแปลงโดยใช้เครื่องมือ เป็นวิธีที่สะดวกที่สุด ในการแปลงชุดคำสั่ง พบว่าในการแปลงจากทั้งหมดจำนวน 6 ไฟล์ มีปัญหาหลังจากแปลงแล้วจำนวน 1 ไฟล์ ซึ่งต้องเข้าไปแก้ไข และหลังจากแก้ไขเพื่อให้แอปพลิเคชันสามารถทำงานได้อย่างถูกต้อง การแก้ไขนักพัฒนาต้องมีความรู้ทางด้านภาษาคอทลินบ้าง เพราะต้องรู้วิธีในการแก้ปัญหาที่เกิดขึ้น

4.1.2 ปัญหาการแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่แต่ใช้โปรเจ็คเดิม พบว่าระยะเวลาที่ใช้ในการพัฒนาจะใช้เวลาปานกลาง และนักพัฒนาจำเป็นต้องมีทักษะทางด้านจาวาและภาษาคอทลิน

4.1.3 ปัญหาการแปลงโดยนักพัฒนาทำการเขียนชุดคำสั่งขึ้นมาใหม่ พบว่าระยะเวลาที่ใช้ในการพัฒนาจะใช้เวลามากที่สุด และนักพัฒนาจำเป็นต้องมีทักษะทางด้านภาษาคอทลินและความสามารถในการพัฒนาแอนดรอยด์แอปพลิเคชัน

4.2 ผลการทดสอบในส่วนของฟังก์ชัน

การแปลงทั้ง 3 แบบ สามารถให้ผลลัพธ์ ได้เช่นเดียวกัน แต่เพื่อให้เกิดความมั่นใจในการทดสอบ ฟังก์ชันการทำงานสามารถนำเอาการทำการทดสอบระดับหน่วย (Unit test) เพื่อทดสอบการทำงานของฟังก์ชันหรือคลาสที่สร้างขึ้น และ การทำการทดสอบระบบ (System test) เพื่อให้เกิดความมั่นใจและคุณภาพของตัวแอปพลิเคชัน

4.3 การเปรียบเทียบในแง่ประสิทธิภาพในแง่ของการพัฒนา

ประสิทธิภาพที่ใช้ในการเปรียบเทียบ โดยเปรียบเทียบโดยขนาดของซอฟต์แวร์ ระยะเวลาในการพัฒนา ได้ผลดังต่อไปนี้ ขนาดของซอฟต์แวร์ เปรียบเทียบโดยใช้ชุดคำสั่ง เนื่องจากระบบมีฟังก์ชันที่แหมือนกัน โดยการเทียบเทียบพบว่ามีจำนวนบรรทัดของชุดคำสั่งที่ลดลง ระยะเวลาในการพัฒนา วิธีการแปลงชุดคำสั่งโดยใช้เครื่องมือ ใช้เวลาน้อยที่สุด ตามด้วยวิธีการแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่แต่ใช้โปรเจ็คเดิม และ วิธีการแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่โดยการสร้างโปรเจ็คใหม่

5. สรุป

ในทุกวิธีในการแปลงผู้ทำการแปลงจะต้องมีความรู้ทั้งภาษาจาวาและคอทลิน ในการเลือกวิธีแปลงชุดคำสั่งหากต้องการความสะดวก นักพัฒนาที่พอมีความรู้ภาษาคอทลินและมีเวลาที่จำกัด ควรใช้วิธีวิธีการแปลงชุดคำสั่งโดยใช้เครื่องมือ ส่วนวิธีการแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่แต่ใช้โปรเจ็คเดิม เหมาะกับนักพัฒนาที่มีความคุ้นเคยกับภาษาคอทลินในระดับที่สูงขึ้น และไม่ต้องการแก้ไขโปรเจ็คทั้งหมด ค่อยๆเปลี่ยนแต่ฟังก์ชันการทำงานยังใช้งานได้เหมือนเดิม และวิธีการแปลงชุดคำสั่งโดยการสร้างชุดคำสั่งขึ้นมาใหม่โดยการสร้างโปรเจ็คใหม่ เหมาะสำหรับการพัฒนาที่มีระยะเวลาในการพัฒนามากกว่า 2 วิธีข้างต้น เหมาะกับนักพัฒนามีความรู้ในภาษาคอทลินหรือ ประสบการณ์ในการเขียนโปรแกรมภาษานี้

ตารางที่ 1 ปัจจัยในการเลือกวิธีที่ใช้ในทางในการแปลงชุดคำสั่ง

แนวทาง	ระยะเวลาที่ใช้	ความรู้ทางด้านภาษาโปรแกรม
A1	น้อย หากไม่พบปัญหา	น้อย
A2	ปานกลาง	ปานกลาง
A3	มาก	มาก

ในแนวทาง A1 กับ A2 สามารถเห็นการทำงานของระบบได้ทันทีหลังจากแปลงชุดคำสั่ง แต่แนวทาง A3 เห็นการทำงานในแต่ละส่วนเรื่อยๆ ตามส่วนที่สร้างขึ้นมา และแนวทาง A1 หากพบปัญหาในการแปลง จะต้องอาศัยความรู้ของนักพัฒนาในการแก้ปัญหาเพื่อให้โปรแกรมทำงานได้ถูกต้อง และในแนวทางที่ A1 กับ A2 หากพบปัญหาเมื่อแปลงแล้วแต่เราแก้ไม่ได้ เราไม่จำเป็นต้องแปลงทั้งหมดเป็นภาษาจาวา ผู้วิจัยเห็นว่าหากต้องการประสิทธิภาพของแอปพลิเคชัน ควรใช้แนวทาง A3 และในการแปลงชุดคำสั่งให้มีขนาดเล็กลงนั้น ขึ้นอยู่กับองค์ความรู้ของนักพัฒนาด้วยเช่นกัน งานวิจัยที่ควรศึกษาต่อไปคือการแปลงชุดคำสั่งของภาษาคอทลินเป็นภาษา Swift เนื่องจากโครงสร้างของทั้งสองภาษามีความใกล้เคียงกัน แม้ว่าไลบรารีจะแตกต่างกันเนื่องจากเป็นภาษาทั้งสองเป็นภาษาที่ใช้ในการพัฒนา ที่ต่างแพลตฟอร์ม และควรทำการเลือกแปลงในส่วนที่เป็น Business logic ก่อนหากพัฒนาแบบ Native application

เอกสารอ้างอิง (References)

A.A. Terekhov, C. Verhoef. (2000). **The Realities of Language Conversion.** IEEE Software, Volume: 17 , Issue: 6, 111 – 124.

Subham Bose, Aditi Kundu, Madhuleena Mukherjee, Madhurima Banerjee.(2018). **A comparative study : Java vs Kotlin programming in android application development**. International Journal of Advanced Research in Computer Science," vol. 9, no. 3, 41-45.

Dony George, Priyanka Girase, Mahesh Gupta, Prachi Gupta, Aakanksha Sharman.(2010). **Programming Language Inter-conversion**. International Journal of Computer Applications, vol. 1, no. 20, (0975-8887).

IDC Corporate USA (2020). **Smartphone Market Share** ค้นเมื่อ 12 เมษายน 2563 จาก https://www.idc.com/promo/smartphone-market-share

Bleeping Computer® (2017). **Kotlin Expected to Surpass Java as Android Default Programming Language for Apps** ค้นเมื่อ 5 มกราคม 2563 จาก https://www.bleepingcomputer.com/news/mobile/kotlin-expected-to-surpass-java-as-android-default-programming-language-for-apps/

Hacker Noon (2019). **The Only Time You Should Use for iOS App Development is Swift—Here's Why Apps** ค้นเมื่อ 5 มกราคม 2563 จาก https://hackernoon.com/the-only-time-you-should-use-for-ios-app-development-is-swift-heres-why-79c837e718f3

Uber Engineering (2019). **Measuring Kotlin Build Performance at Uber** ค้นเมื่อ 5 มกราคม 2563 จาก https://eng.uber.com/measuring-kotlin-build-performance/