

## บทที่ 3

### วิธีการดำเนินงานวิจัย

#### 3.1 การพัฒนาการคอมไพเลอร์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศ

ในการจัดทำคอมไพเลอร์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศให้เป็นแบบอัตโนมัติขึ้น เพื่อลดการเสียเวลาในการทำงาน และทำให้สามารถดำเนินงานเป็นไปได้อย่างราบรื่น และสามารถได้פקเกจที่สามารถใช้งานได้ทันที

สำหรับวิธีการดำเนินงานการคอมไพเลอร์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศแบบอัตโนมัติ แบ่งขั้นตอนออกเป็น 5 ขั้นตอนดังนี้

- 3.1.1 ขั้นตอนการวางแผนและการเตรียมการ
- 3.1.2 ขั้นตอนการวิเคราะห์ระบบ
- 3.1.3 ขั้นตอนการออกแบบระบบ
- 3.1.4 ขั้นตอนการพัฒนา
- 3.1.5 ขั้นตอนการทดสอบระบบ

##### 3.1.1 ขั้นตอนการวางแผนและการเตรียมการ

ในการดำเนินการจัดทำโครงการคอมไพเลอร์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศนี้ ได้มีการวางแผนและเตรียมการดังนี้

1. ศึกษาความเป็นไปได้และเก็บรวบรวมข้อมูลเกี่ยวกับการจัดทำโครงการในครั้งนี้
2. ศึกษาข้อมูลเกี่ยวกับภาษา shell script ในการจัดทำโครงการนี้ ที่เลือกภาษานี้มาใช้งานเพราะว่า มีความเห็นว่าจะต้องทำงานบนระบบปฏิบัติการลินุกซ์
3. ศึกษาขั้นตอนการทำงานของระบบเดิมของการคอมไพเลอร์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศ เพื่อนำไปพัฒนาระบบงานใหม่

##### 3.1.2 ขั้นตอนการวิเคราะห์ระบบ

- วิเคราะห์ระบบงานเดิม

การคอมไพเลอร์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศนั้น หากจะมีการเริ่มคอมไพเลอร์ระบบ ต้องใช้ทรัพยากรมนุษย์ในการทำงานเสมอ โดยมีขั้นตอนในการทำงานดังนี้

- 1) ต้องมีการกำหนดว่าในแต่ละอาทิตย์ จะต้อง Build\* วันไหน และใครต้องเป็นคนสั่งให้คอมไพล์
- 2) คนที่จะ Build ต้องเข้าสู่ระบบด้วยชื่อผู้ใช้งาน “retbuild” ที่เครื่องเซิร์ฟเวอร์ทั้งสองเครื่อง คือ เครื่อง ncq-retbuild01 เป็นเครื่องสถาปัตยกรรมแบบ x86 ระบบปฏิบัติการโซลาริส 10 (Solaris Operating System) และเครื่อง ncq-retbuild02 สถาปัตยกรรมแบบ x86 ระบบปฏิบัติการแบบลินุกซ์ด้วย โดยเครื่องเซิร์ฟเวอร์ทั้งสองเครื่องถูกติดตั้งอยู่ที่ประเทศอังกฤษ เมืองนอตทิงแฮม
- 3) ต้องทำการหาที่อยู่ที่จะทำการ Build ด้วยการใช้ชุดคำสั่งโดยการใช้โปรแกรม SuperPuTTY ในการเข้าถึงเครื่องเซิร์ฟเวอร์
- 4) ต้องพิมพ์คำสั่งที่ใช้ในการ Build ในเครื่อง ncq-retbuild01 ก่อน
- 5) จากนั้นเข้าไปที่เครื่อง ncq-retbuild02 เพื่อทำการเตรียม Build แต่การที่จะ Build ได้ต้องรอเครื่อง ncq-retbuild01 ทำงานถึงขั้นตอนที่สามารถ Build เครื่อง ncq-retbuild02 ได้
- 6) เมื่อถึงขั้นตอนที่สามารถรันเครื่อง ncq-retbuild02 ได้ ให้พิมพ์ชุดคำสั่งเพื่อทำการ Build

จะเห็นได้ว่าการทำงานที่ใช้การทรัพยากรมนุษย์ในการทำงานเป็นอย่างมาก ทำให้เกิดปัญหาในการทำงาน ดังนี้

- 1) ผู้พัฒนาไม่สามารถที่จะจำคำสั่งในการ Build ได้ ทำให้ต้องเสียเวลาในการหา และต้องจดเก็บเอาไว้
- 2) หากไม่กำหนดว่าจะ Build ทุกๆ วันอะไรของสัปดาห์ และไม่กำหนดว่าใครจะเป็นคน Build อาจจะทำให้เกิดการชนกันของ package ได้
- 3) ในการที่จะ Build ต้องมีการลงชื่อเข้าใช้ระบบ บางครั้งไม่สามารถลงชื่อเข้าใช้ระบบได้
- 4) การที่จะ Build เครื่อง ncq-retbuild02 ได้นั้นจะต้อง Build เครื่อง ncq-retbuild01 ให้ถึงขั้นตอนที่จะทำการ Build เครื่อง ncq-retbuild02 ได้ก่อน ทำให้ทางผู้พัฒนาเสียเวลาในกานที่ต้องคอยดูหน้าจอรว่า Build ถึงขั้นตอนไหนแล้ว

---

\*Build คือการที่เอาชุดคำสั่งของโปรแกรมและทรัพยากรต่างๆ ที่เขียนมาคอมไพล์ (Compile) และมาห่อรวมกัน (package) ให้ได้ผลลัพธ์สุดท้าย

- 5) การตั้งค่าช่วงเวลาที่จะทำการ Build ในตอนกลางคืนนั้นทำค่อนข้างยาก เพราะต้องไปตั้งค่าในครอนแท็บ (Crontab)
- 6) หากทำการ Build อยู่ ต้องเปิดเครื่องทิ้งเอาไว้ หากปิดเครื่องการ Build จะหยุดการทำงานทันที

- วิเคราะห์ระบบงานใหม่

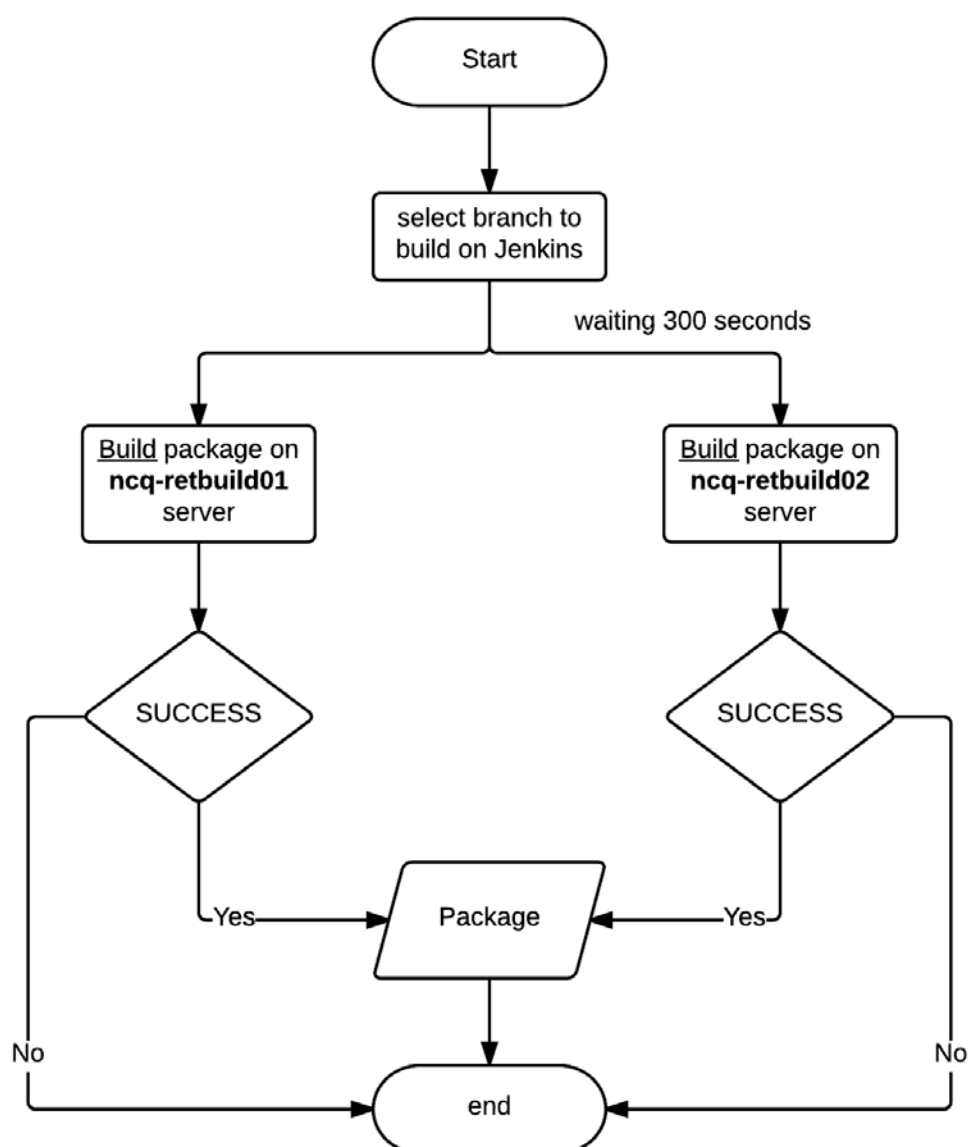
เป็นการทำงานที่เอาระบบการทำงานแบบอัตโนมัติช่วยในการทำงาน โดยทางผู้จัดทำได้อาวิธีการ และขั้นตอนต่างๆ มาเรียบเรียง และได้เขียนเป็นแผนภาพ (Flow chart) เพื่อออกแบบระบบงานใหม่

ในการวิเคราะห์ระบบงานใหม่ ได้มีการเอาแนวทางวิธีการทำงานแบบอัตโนมัติมาช่วยในการวิเคราะห์ระบบงาน เพื่อให้ได้ระบบงานที่มีความเสถียรมากยิ่งขึ้น โดยได้ปรับปรุงขั้นตอนให้มีการใช้งานที่สะดวกมากยิ่งขึ้น ดังนี้

- 1) เข้าหน้าเว็บไซต์เจเนกิน
- 2) เลือกงาน (Jobs) ที่ต้องการจะ Build
- 3) รอผลการ Build ผ่านทางจดหมายอิเล็กทรอนิกส์

จะเห็นได้ว่า ระบบงานใหม่ที่ทำนั้นมีขั้นตอนการทำงานที่สะดวกและรวดเร็วกว่าการทำงานแบบระบบเก่า ประโยชน์ของการนำระบบอัตโนมัติมาใช้

- 1) ลดเวลารวมในการ Build แต่ละครั้ง
- 2) เพิ่มประสิทธิภาพในการทำงาน และช่วยลดเวลาในการทำงานของผู้พัฒนาระบบ
- 3) เพิ่มความสะดวกในการใช้งาน ทำให้มีความรวดเร็วในการทำงานมากยิ่งขึ้น
- 4) ลดความผิดพลาด และเพิ่มความถูกต้องให้กับงานมากยิ่งขึ้น



รูปที่ 3.1 การทำงานของการคอมไพล์ระบบจำลองการแลกเปลี่ยนเงินตราระหว่างประเทศ ระบบงานใหม่

### 3.1.3 ขั้นตอนการออกแบบระบบ

#### 1) แผนภาพบริบท (Context diagram)

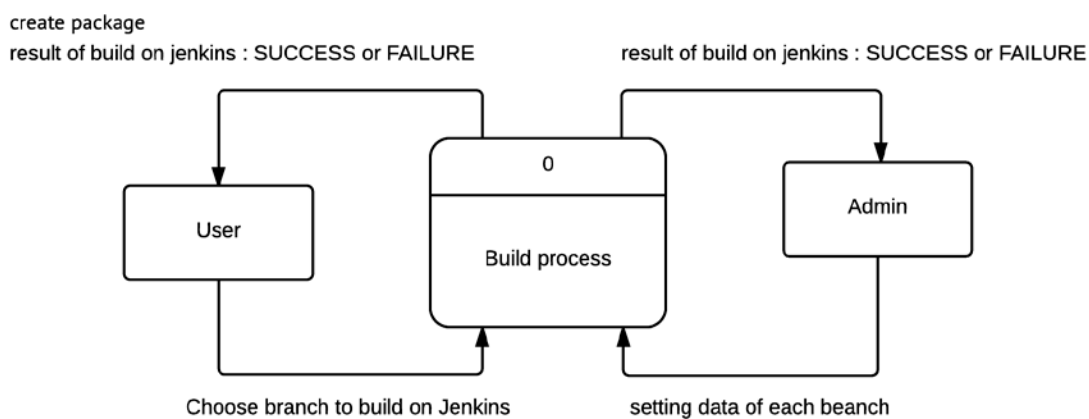
แผนภาพบริบท ( Context diagram) คือ แผนภาพแสดงกระแสการไหลของข้อมูลระดับบนสุด แสดงภาพรวมการทำงานของระบบที่มีความสัมพันธ์กับระบบภายนอก และแสดงถึงขอบเขตของระบบที่ทำการศึกษาและพัฒนา

โดยผู้ที่เกี่ยวข้องกับระบบ แบ่งเป็น 2 ส่วนคือ ผู้ใช้งาน และ ผู้ดูแลระบบ

- ผู้ใช้งาน ผู้ใช้งานต้องลงชื่อเข้าใช้ระบบก่อน ถึงจะมีสิทธิในการทำงานได้

- ผู้ดูแลระบบ ต้องลงชื่อเข้าใช้งาน และมีสิทธิในการแก้ไข และจัดการข้อมูลได้

ดังรูปที่ 3.2

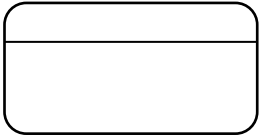




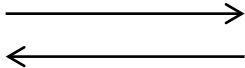
รูปที่ 3.2 แผนภาพบริบทของการคอมพิวเตอร์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศ

## 2) แผนภาพกระแสข้อมูล (Data flow diagram)

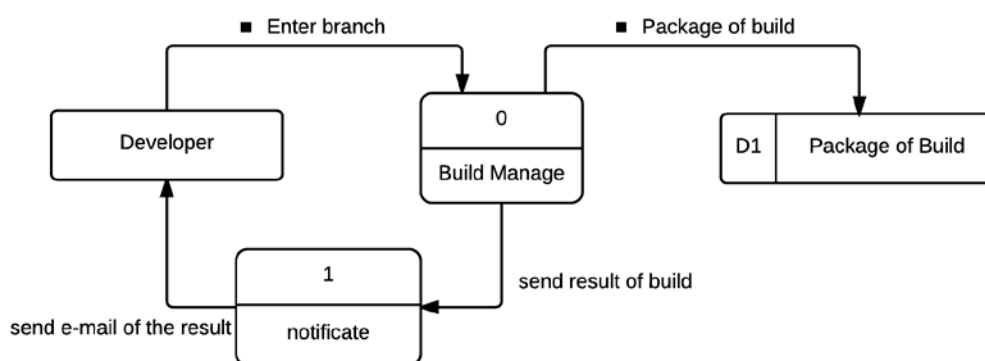
แผนภาพกระแสข้อมูล คือ เป็นแบบจำลองการทำงานของระบบ เพื่ออธิบายขั้นตอนการทำงานของระบบ แผนภาพจะแสดงทิศทางการไหลของข้อมูลและอธิบายความสัมพันธ์ของการดำเนินงานของระบบ โดยการทำงานของระบบการคอมพิวเตอร์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศนั้น มีแผนภาพกระแสข้อมูล 2 ระดับ คือ level 0 และ level 1

ตารางที่ 3.1 แสดงสัญลักษณ์ของแผนภาพกระแสข้อมูล

สัญลักษณ์	ความหมาย
	การประมวลผล (Process)
	ที่เก็บข้อมูล (Data store)

	ปัจจัย หรือ สิ่งที่อยู่ภายนอก (External Entity)
	เส้นทางการไหลของข้อมูล (Data flow)

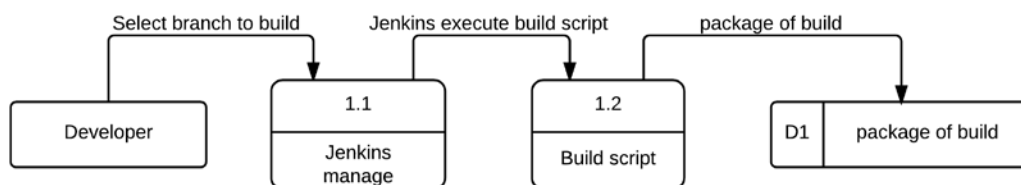
- แผนภาพแสดงกระแสของข้อมูล (Data Flow Diagram : Level 0)



รูปที่ 3.3 แผนภาพกระแสข้อมูลของการคอมไพล์ระบบจำลองการแลกเปลี่ยนเงินตราระหว่างประเทศ ระดับ 0

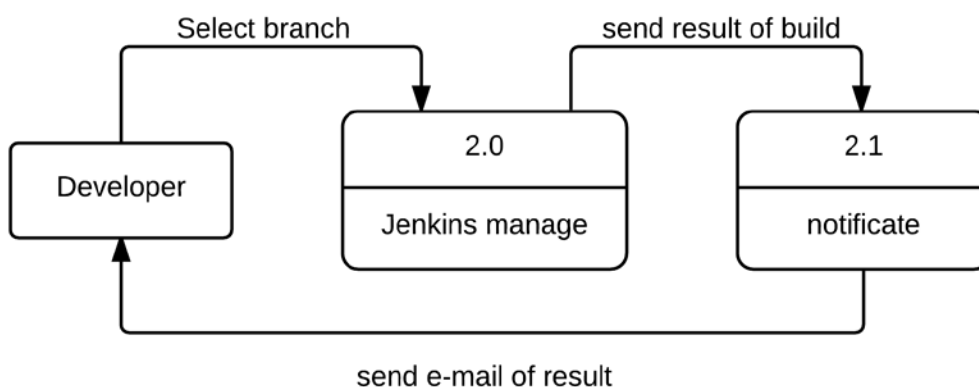
จากรูปที่ 3.3 จะเห็นได้ว่ามีกระแสข้อมูล ระดับที่ 0 คือมีกระบวนการในการทำงานระบบนี้ อยู่ 2 กระบวนการ คือ Build manage คือ กระบวนการการ Build ของ ซอร์สโค้ดเพื่อให้ได้ package และ อีกกระบวนการคือ Notificate เป็นการแจ้งผลลัพธ์ของการ Build ว่าสำเร็จหรือไม่ ทางจดหมายอิเล็กทรอนิกส์ ซึ่งจะอธิบายกระบวนการทั้งสองด้วยแผนภาพกระแสข้อมูล ระดับ 1 ดังนี้

- แผนภาพแสดงกระแสของข้อมูล (Data Flow Diagram : Level 1)



รูปที่ 3.4 แผนภาพแสดงกระแสข้อมูล ระดับที่ 1 ของ Build manage

จากรูปที่ 3.4 เห็นได้ว่ากระบวนการ Build manage สามารถแยกได้เป็น 2 กระบวนการ ดังภาพด้านบน เป็นขั้นตอนการใช้งานเจนนิงส์ โดยจะเห็นได้ว่า กระบวนการ Jenkins manage นั้นมีการส่งต่อข้อมูลจากผู้ใช้ไปยังกระบวนการ Build script เพื่อทำการ Build package เพื่อให้ได้ package เพื่อมาใช้งาน



รูปที่ 3.5 แผนภาพแสดงกระแสข้อมูล ระดับที่ 1 ของ Notificate

จากรูปข้างต้น กระบวนการ Notificate นั้นต้องผ่านการประมวลผลจากกระบวนการ Build manage ก่อนเพื่อที่จะได้แสดงผลลัพธ์ของการ Build ให้ผู้พัฒนาทางจดหมายอิเล็กทรอนิกส์ได้

### 3.1.4 ขั้นตอนการพัฒนาระบบ

จากกระบวนการข้างต้นที่กล่าวมาทำให้เห็นถึง ขั้นตอนต่างๆ ในการทำวิจัยระบบการคอมไพล์ระบบจำลองการแลกเปลี่ยนอัตราเงินระหว่างประเทศ ซึ่งการคอมไพล์ระบบนั้นต้องใช้เครื่องมือในของกระบวนการบูรณาการอย่างต่อเนื่อง คือ Jenkins และต้องใช้ภาษา Shell Script ในการเขียน Script เพื่อเรียกใช้คำสั่งในการ Build อีกด้วย

### 3.1.5 ขั้นตอนการทดสอบระบบ

ในขั้นตอนการทดสอบระบบนั้น ทางผู้จัดทำได้ทดสอบระบบดังนี้

- 1) ทดสอบเข้าหน้าเว็บไซต์ของ Jenkins
- 2) ทดสอบเมนูการเลือก Branch\*\* เพื่อการไม่เกิดความผิดพลาดในการเลือก
- 3) ทดสอบการ Build เพื่อทำการคอมไพล์ซอสโค้ดแล้วรวมแพ็คเกจเพื่อนำไปใช้งาน
- 4) ทดสอบระบบการส่งจดหมายอิเล็กทรอนิกส์

## 3.2 การพัฒนาการอัปเดตระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศ

การอัปเดตระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศให้เป็นระบบอัตโนมัตินั้น เพื่อความสะดวกของผู้พัฒนาระบบในการอัปเดตระบบ อีกทั้งยังช่วยลดเวลาในการทำงาน และสามารถใช้งานระบบได้ทันที

สำหรับวิธีดำเนินการพัฒนาระบบอัปเดตระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศให้เป็นระบบอัตโนมัติ สามารถแบ่งขั้นตอนออกเป็น 5 ขั้นตอนได้ดังนี้

- 3.2.1 ขั้นตอนการวางแผนและการเตรียมการ
- 3.2.2 ขั้นตอนการวิเคราะห์ระบบ
- 3.2.3 ขั้นตอนการออกแบบระบบ
- 3.2.4 ขั้นตอนการพัฒนา
- 3.2.5 ขั้นตอนการทดสอบระบบ

### 3.2.1 ขั้นตอนการวางแผนและการเตรียมการ

ในการดำเนินการจัดทำโครงการการคอมไพล์ระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศนี้ ได้มีการวางแผนและเตรียมการดังนี้



Branch\*\* หมายถึง การแยกตัวออกไปจากโปรแกรมที่ทำอยู่ ไปทำโปรแกรมอีกอันหนึ่ง เช่น แยกออกจากโปรแกรมหลัก (main program) ไปทำงานที่โปรแกรมน้อย (subprogram)

- 1) ศึกษาความเป็นไปได้และเก็บรวบรวมข้อมูลเกี่ยวกับการจัดทำโครงการในครั้งนี้
- 2) ศึกษาข้อมูลเกี่ยวกับภาษา shell script ในการจัดทำโครงการนี้ ที่เลือกภาษานี้มาใช้งาน เพราะว่า มีความเห็นว่าจะต้องทำงานบนระบบปฏิบัติการลินุกซ์
- 3) ศึกษาขั้นตอนการทำงานของระบบเดิมของการอัพเกรดระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศ เพื่อนำไปพัฒนาระบบงานใหม่

### 3.2.2 ขั้นตอนการวิเคราะห์ระบบ

- วิเคราะห์ระบบงานเดิม

ขั้นตอนการอัพเกรดระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศนั้นมีขั้นตอนที่มากมาย และยุ่งยาก ดังนี้

- 1) ต้องใช้โปรแกรม SuperPuTTY เพื่อเข้าไปที่เครื่อง ncq-retbuild01
- 2) ต้องใช้ชุดคำสั่งคอมมานด์ไลน์ในการหาที่อยู่ของระบบที่จะอัพเกรด
- 3) เข้าไปตรวจสอบว่ามีแพ็คเกจของระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศหรือไม่
- 4) หากมี ต้องตรวจสอบว่าระบบจำลองการแลกเปลี่ยนเงินตรานั้นเป็นระบบล่าสุดแล้วหรือยัง
- 5) หากเป็นระบบล่าสุดแล้ว ให้คัดลอกระบบมาไว้ที่เครื่องของผู้พัฒนา

จะเห็นได้ว่าการอัพเกรดระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศนั้นมีขั้นตอนที่ยุ่งยากซับซ้อน จึงเกิดปัญหาดังนี้

- 1) ต้องพิมพ์ชุดคำสั่งในการอัพเกรดระบบทุกขั้นตอน โดยการพิมพ์ชุดคำสั่งนั้นต้องทำโดยผู้พัฒนาระบบ ทำให้มีโอกาสในการผิดพลาดสูง เช่นการพิมพ์ชุดคำสั่ง การดูแลรุ่นของระบบจำลองผิด
- 2) ผู้พัฒนาระบบไม่ยากที่จะอัพเกรดระบบ เพราะว่า มีกระบวนการในการอัพเกรดระบบที่ยุ่งยาก
- 3) ถ้าหากอยู่ในระหว่างการอัพเกรดระบบอยู่ แล้วมีการปิดเครื่องคอมพิวเตอร์นั้น การอัพเกรดระบบก็จะสิ้นสุดลง

- วิเคราะห์ระบบงานใหม่

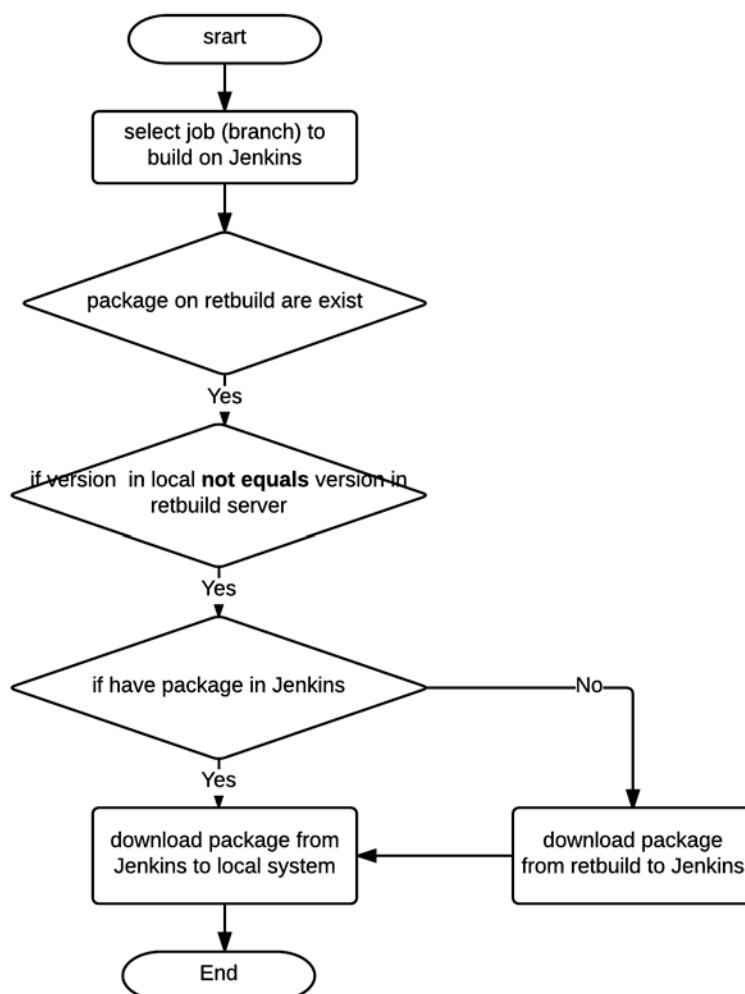
การพัฒนาระบบการอัพเกรดระบบจำลองการแลกเปลี่ยนเงินตราระหว่างประเทศให้เป็นระบบอัตโนมัติ นั้นทางผู้จัดทำได้เขียนขั้นตอนต่างๆ ในการพัฒนาระบบใหม่เป็น Flow Chart

ในการวิเคราะห์ระบบงานใหม่ ได้เอาแนวคิดของการบูรณาการอย่างต่อเนื่องมาเป็นตัวช่วยในพัฒนาระบบ โดยระบบใหม่สามารถทำงานตามขั้นตอนดังนี้

- 1) เข้าเว็บไซต์เงินกินส์
- 2) เลือกงาน (Jobs) ที่ต้องการจะอัพเกรด
- 3) รวบรวมการอัพเกรดทางจดหมายอิเล็กทรอนิกส์

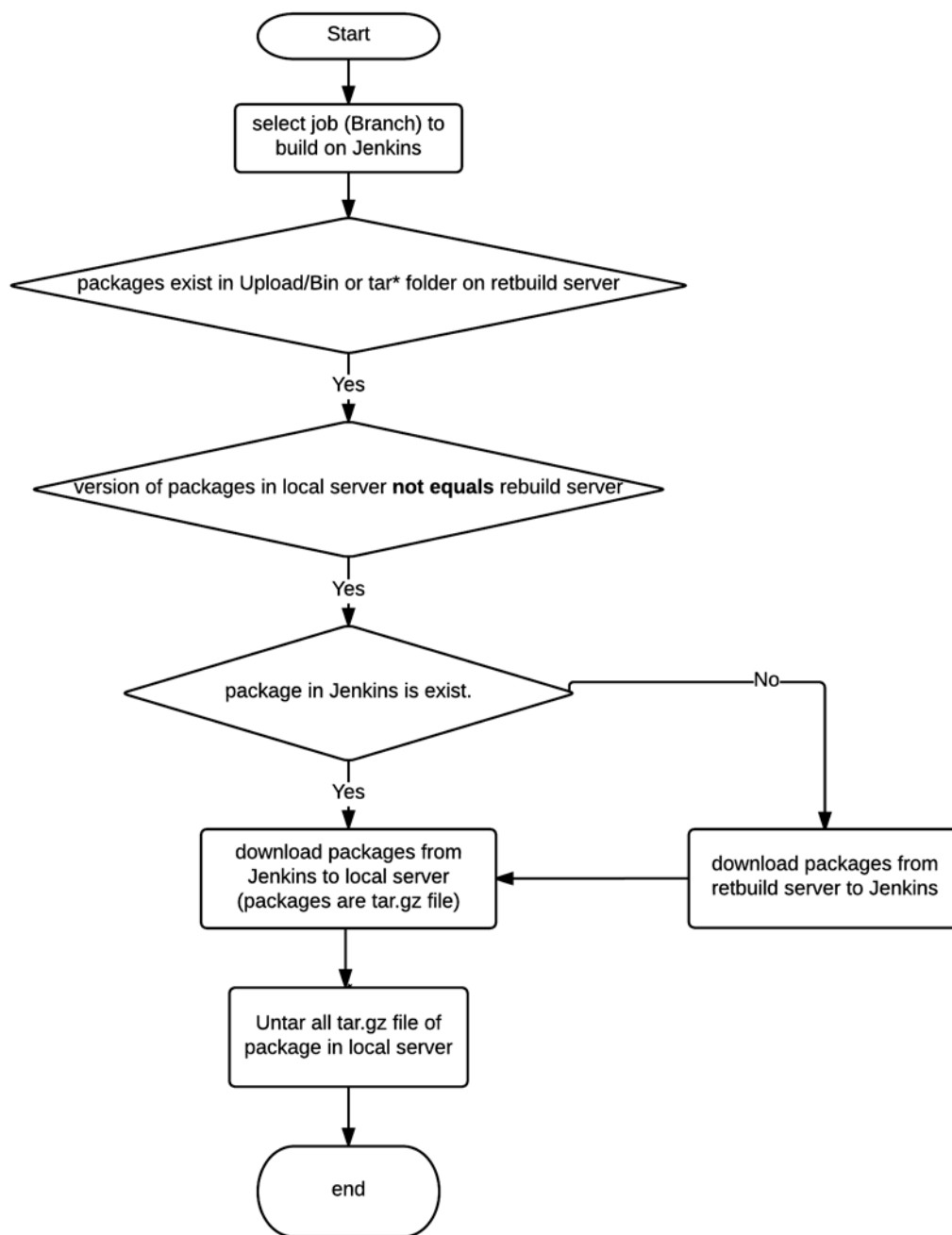
จะเห็นได้ว่า ระบบงานใหม่ที่ทำนั้นมีขั้นตอนการทำงานที่สะดวกและรวดเร็ว กว่า การทำงานแบบระบบเก่า ประโยชน์ของการนำระบบอัตโนมัติมาใช้

- 1) เพิ่มประสิทธิภาพในการทำงาน และช่วยลดเวลาในการทำงานของผู้พัฒนาระบบ
- 2) เพื่อเพิ่มความสะดวกในการทำงาน และช่วยเพิ่มความรวดเร็วในการทำงานให้มากยิ่งขึ้น
- 3) ลดความผิดพลาดในการทำงาน



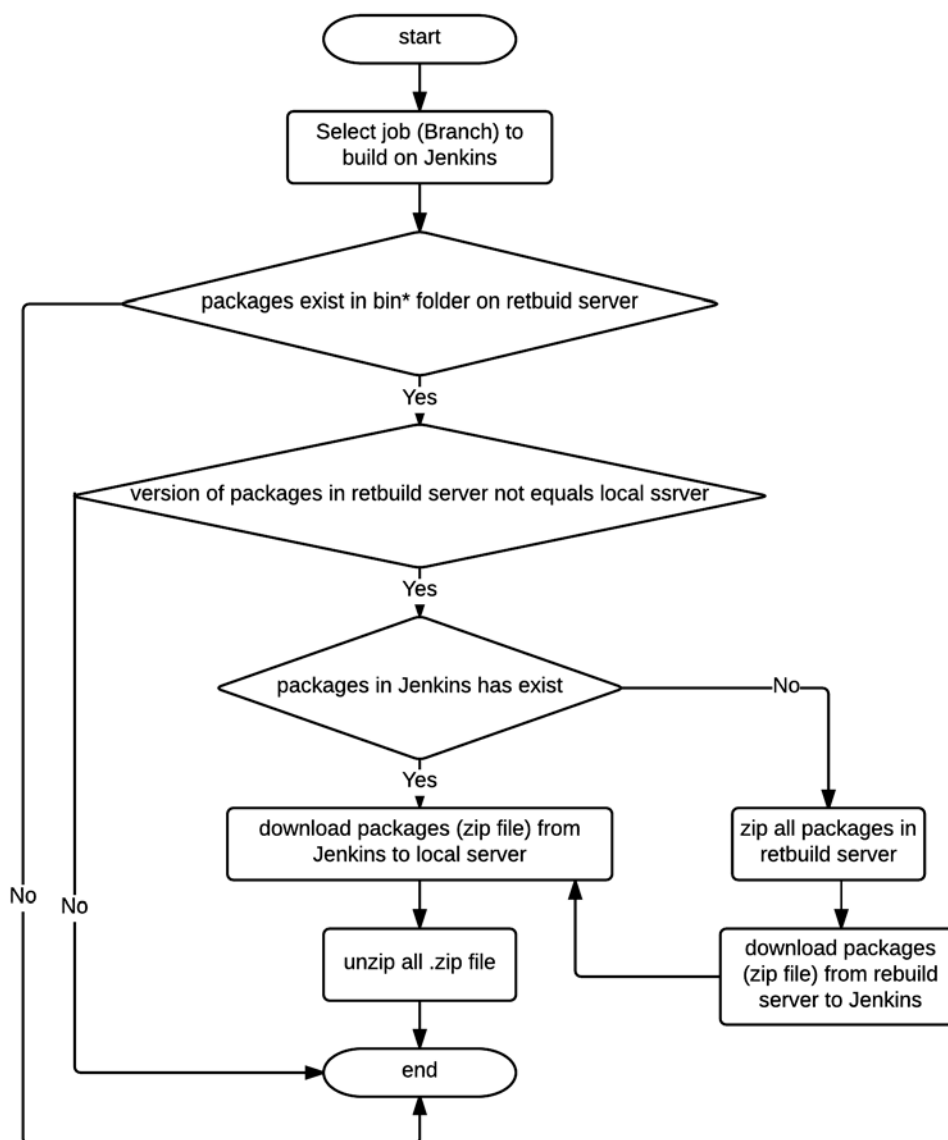
รูปที่ 3.6 แผนภาพ (Flow Chart) แสดงภาพรวมของระบบการทำงานแบบใหม่

จากรูปที่ 3.6 จะเห็นได้ถึงการทำงานระบบใหม่ โดยจะมีการตรวจสอบว่ามีแพ็คเกจของระบบจำลองอยู่ที่ไหน จึงสามารถแยกการทำงานได้ 2 กระบวนการ ดังรูปที่ 3.7 และ รูปที่ 3.8



รูปที่ 3.7 แผนภาพ (Flow Chart) แสดงถึงขั้นตอนการทำงานของการทำงานของการค้นหาที่อยู่ของแพ็คเกจ(1)

จากรูปที่ 3.7 แสดงถึงการค้นหาที่อยู่ของแพ็คเกจ ในโฟลเดอร์ Upload/Bin และโฟลเดอร์ tar\* และยังมีการค้นหาในโฟลเดอร์ bin\* อีกด้วย ดังรูปที่ 3.8



รูปที่ 3.8 แผนภาพ (Flow Chart) แสดงถึงขั้นตอนการทำงานของการทำงานของการค้นหาที่อยู่ของแพ็คเกจ(2)

จากรูปที่ 3.8 จะเห็นได้ว่าการค้นหาในไฟล์เดอร์ bin\* แล้วถ้าเจอว่ามีแพ็คเกจอยู่ก็จะทำการตรวจสอบรุ่นว่าเป็นเวอร์ชันล่าสุดหรือไม่ โดยการไปเปรียบเทียบกับเวอร์ชันที่อยู่ในเครื่องของผู้พัฒนา หากเวอร์ชันเหมือนกัน หรือเวอร์ชันนั้นต่ำกว่า ก็จะไม่ทำการดาวน์โหลดแพ็คเกจมาไว้ที่เครื่องของผู้พัฒนา และจะออกจากโปรแกรมทันที แต่หากเวอร์ชันนั้นไม่ตรงกันและเป็นเวอร์ชันที่สูงกว่าเครื่องของผู้พัฒนา แสดงว่าเป็นเวอร์ชันใหม่ ให้ทำการตรวจสอบต่อว่าในเซิร์ฟเวอร์เครื่องของ เจนกินส์นั้นมี เวอร์ชันนี้อยู่หรือไม่ หากไม่มี จะทำการดาวน์โหลดมาไว้ที่เครื่องเจนกินส์เพื่อเป็นการสำรองข้อมูล จากนั้นให้ดาวน์โหลดจากเครื่องเจนกินส์มาไว้ที่เครื่องของผู้พัฒนา หากเครื่องเจนกินส์นั้นมีเป็นเวอร์ชันใหม่อยู่แล้ว ก็จะไม่ทำการดาวน์โหลดจากเครื่องเจนกินส์มาไว้ที่เครื่องพัฒนา

โดยตรง และทำการออกจากโปรแกรม สิ้นสุดการอัปเดตระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศ

สิ่งที่แตกต่างระหว่าง การค้นหาในโฟลเดอร์ bin\* และ Upload/Bin หรือ tar\* นั้นคือ ไฟล์ของแพ็คเกจ หากแพ็คเกจนั้นอยู่ที่โฟลเดอร์ bin\* นั้น จะเป็นตัวแพ็คเกจของระบบ ทำให้ต้องบีบอัดไฟล์เพื่อลดขนาดของไฟล์ให้มีความเล็กลง เพราะต้องดาวน์โหลดไฟล์จากเครื่องเซิร์ฟเวอร์ที่อยู่ประเทศอังกฤษ เมืองนอตติงแฮม เพราะว่าเครื่องเซิร์ฟเวอร์ทั้งที่ประเทศไทยและเมืองนอตติงแฮมนี้มีระยะทางที่ห่างกันเป็นอย่างมาก ทำให้อาจเกิดการดาวน์โหลดที่ช้าหากไม่มีการบีบอัดไฟล์ โดยจะบีบอัดไฟล์เป็นนามสกุล .zip ส่วนการค้นหาไฟล์ในโฟลเดอร์ Upload/Bin และ tar\* นั้นมีความเหมือนกันคือ แพ็คเกจจะถูกบีบอัดเป็นนามสกุล tar.gz อยู่แล้ว ทำให้อาจดาวน์โหลดมาได้เลย โดยไม่ต้องบีบอัดไฟล์อีกครั้ง

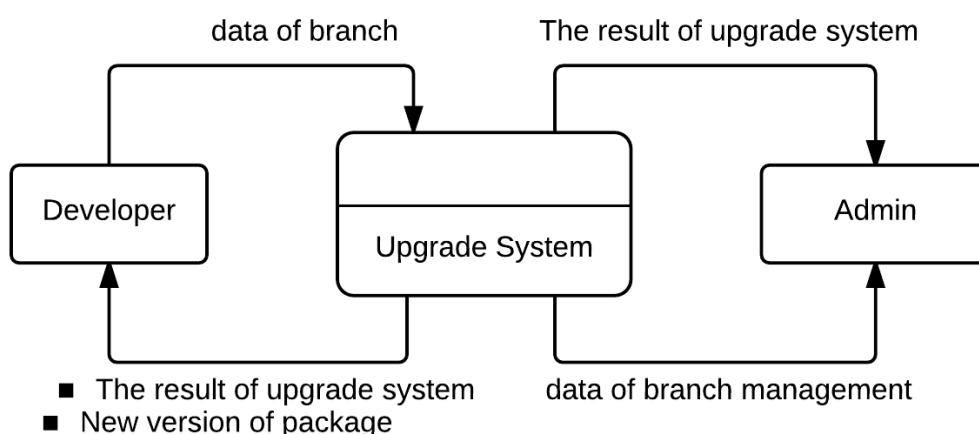
### 3.2.3 ขั้นตอนการวิเคราะห์ระบบ

#### 1) แผนภาพบริบท (Context diagram)

แผนภาพบริบท (Context diagram) คือ แผนภาพแสดงกระแสการไหลของข้อมูลระดับบนสุด แสดงภาพรวมการทำงานของระบบที่มีความสัมพันธ์กับระบบภายนอก และแสดงถึงขอบเขตของระบบที่ทำการศึกษาและพัฒนา

โดยผู้ที่เกี่ยวข้องกับระบบ แบ่งเป็น 2 ส่วนคือ ผู้ใช้งาน และ ผู้ดูแลระบบ

- ผู้ใช้งาน ผู้ใช้งานต้องลงชื่อเข้าใช้ระบบก่อน ถึงจะมีสิทธิในการทำงานได้
- ผู้ดูแลระบบ ต้องลงชื่อเจ้าใช้งาน และมีสิทธิในการแก้ไข และจัดการข้อมูลได้ดังรูปที่ 3.9

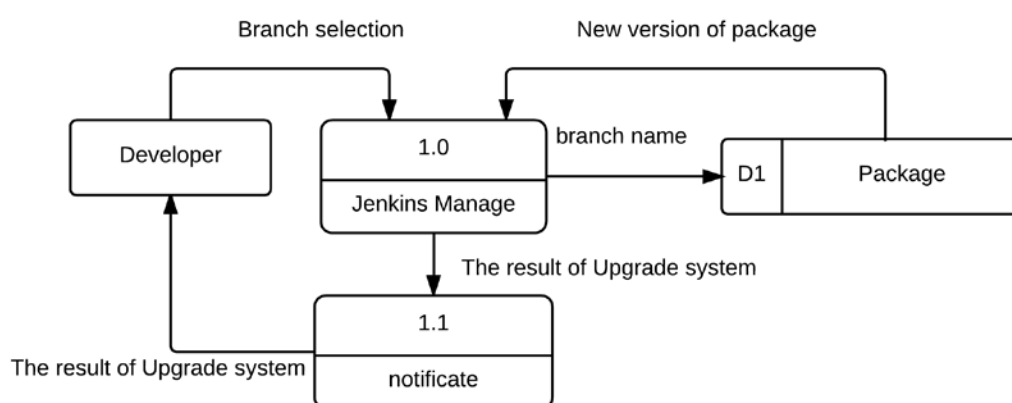


รูปที่ 3.9 แผนภาพบริบทของการอัปเดตระบบจำลองการแลกเปลี่ยนเงินตราต่างประเทศ

## 2) แผนภาพกระแสข้อมูล (Data flow diagram)

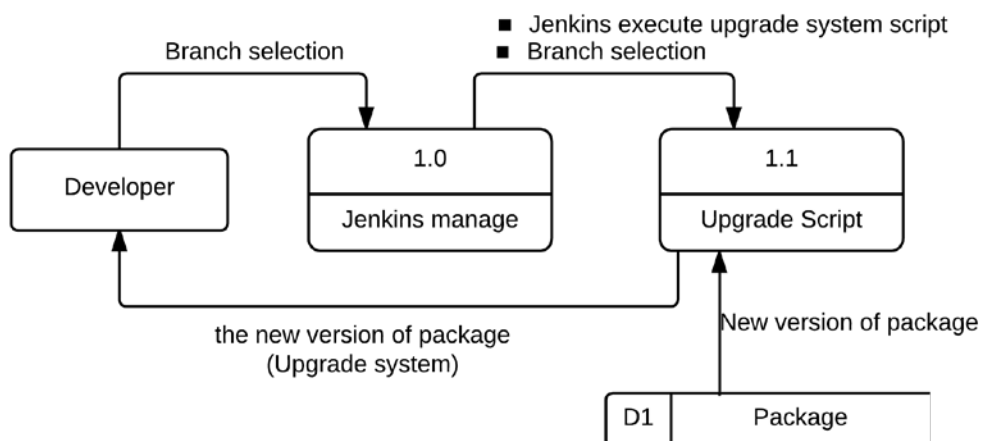
แผนภาพกระแสข้อมูล คือ เป็นแบบจำลองการทำงานของระบบ เพื่ออธิบายขั้นตอนการทำงานของระบบ แผนภาพจะแสดงทิศทางการไหลของข้อมูลและอธิบายความสัมพันธ์ของการดำเนินงานของระบบ โดยการทำงานของระบบการคอมพิวเตอร์ระบบจำลองการแลกเปลี่ยนเงินตราระหว่างประเทศนั้น มีแผนภาพกระแสข้อมูล 2 ระดับ คือ level 0 และ level 1 มีสัญลักษณ์ในการทำงานดัง ตารางที่ 3.1

- แผนภาพแสดงกระแสของข้อมูล (Data Flow Diagram : Level 0)



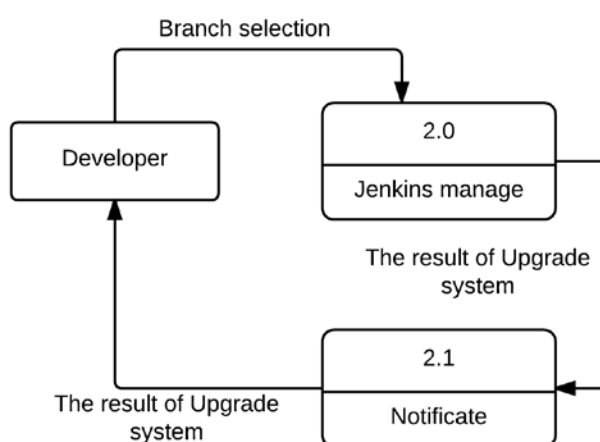
รูปที่ 3.10 แผนภาพกระแสข้อมูลของการอัพเกรดระบบจำลองการแลกเปลี่ยนเงินตราระหว่างประเทศ ระดับ 0

- แผนภาพแสดงกระแสของข้อมูล (Data Flow Diagram : Level 1)



รูปที่ 3.11 แผนภาพแสดงกระแสข้อมูล ระดับที่ 1 ของ Jenkins manage process

จากรูปที่ 3.11 จะเห็นได้ว่าแผนภาพกระแสข้อมูลระดับที่ 1 ของ Jenkins manage process นั้น มีการส่งส่งข้อมูลชื่อ Branch จากผู้พัฒนาระบบไปที่กระบวนการของการจัดการเงินกู้ยืม และมีการส่งต่อไปยังสคริปต์ที่ทำการประมวลผล โดยมีการเรียกเอา package ของระบบจำลองการแลกเปลี่ยนเงินตราระหว่างประเทศมาเพื่อทำการประมวลผลอีกด้วย โดยการประมวลผลนั้นจะเป็นไปตามขั้นตอนของรูปที่ 3.7 และรูปที่ 3.8 จากนั้นเมื่อทำการประมวลผลเสร็จสิ้น จะส่งผลลัพธ์ของการประมวลผลไปที่ผู้พัฒนา



รูปที่ 3.12 แผนภาพแสดงกระแสข้อมูล ระดับที่ 1 ของ notificate process



จากรูปที่ 3.12 เป็นแผนภาพที่แสดงการแจ้งผลลัพธ์ของการทำงานของระบบว่าสำเร็จหรือไม่ โดยทางผู้พัฒนาจะทำการส่งข้อมูลไปยังกระบวนการ Jenkins manage แล้วจะประมวลผลออกมา เมื่อมีการประมวลผลเสร็จแล้ว จะส่งผลลัพธ์มาที่กระบวนการ Notificate ซึ่งเป็นปลั๊กอินของเครื่องมือเจกินส์ ก็จะทำให้การส่งจดหมายอิเล็กทรอนิกส์ไปยังผู้พัฒนาระบบเพื่อให้ทราบผลลัพธ์ว่าการอัปเดตระบบนั้นสำเร็จหรือไม่

### 3.2.4 ขั้นตอนการพัฒนา

จากกระบวนการการทำงานระบบงานข้างต้นที่กล่าวมาทำให้เห็นถึง ขั้นตอนต่างๆ ในการทำวิจัยระบบการอัปเดตระบบจำลองการแลกเปลี่ยนอัตราเงินระหว่างประเทศ ซึ่งการอัปเดตระบบนั้นต้องใช้เครื่องมือในของกระบวนการบูรณาการอย่างต่อเนื่อง คือ Jenkins เพื่อในการช่วยให้การทำงานเป็นระบบอัตโนมัติ เพื่อช่วยลดความซับซ้อนและเวลาในการทำงาน เนื่องจากระบบเครื่องเซิร์ฟเวอร์นั้นเป็นระบบปฏิบัติการโซลาริส (Solaris) ทำให้ต้องเขียนชุดคำสั่งในภาษา Shell script ในการทำตามขั้นตอนต่างๆ

### 3.2.5 ขั้นตอนการทดสอบระบบ

ในขั้นตอนการทดสอบระบบนั้น ทางผู้จัดทำได้ทดสอบระบบดังนี้

- 1) ทดสอบเข้าหน้าเว็บไซต์ Jenkins ของบริษัทรอยเตอร์ส ซอฟต์แวร์ (ประเทศไทย) จำกัด (มหาชน) ในประเทศไทย
- 2) ทดสอบเมนูการเลือก Branch เพื่อการไม่เกิดความผิดพลาดในการเลือก
- 3) ทดสอบการอัปเดต โดยการกดปุ่ม “Build Now” เพื่อทำการอัปเดตระบบการแลกเปลี่ยนเงินตราต่างประเทศ
- 4) ทดสอบระบบการส่งจดหมายอิเล็กทรอนิกส์

## 3.3 การพัฒนาการทดสอบประสิทธิภาพของซอฟต์แวร์

การทดสอบประสิทธิภาพของซอฟต์แวร์ (Performance testing) เป็นการทดสอบประสิทธิภาพของซอฟต์แวร์ที่ถูกพัฒนาขึ้นมา เช่น ทดสอบว่า ระบบที่ถูกพัฒนาขึ้นมานั้นสามารถรองรับการทำงานหนักได้ดีมากน้อยเท่าใด เมื่อมีผู้ใช้งานจำนวนมาก ซอฟต์แวร์นั้นมีการตอบสนองเป็นอย่างไร

### 3.1.1 ขั้นตอนการวางแผนและการเตรียมการ

ในการดำเนินการจัดทำแผนการพัฒนาการทดสอบประสิทธิภาพของซอฟต์แวร์ ได้มีการวางแผนและเตรียมการดังนี้

1. ศึกษาความเป็นไปได้และเก็บรวบรวมข้อมูลที่เกี่ยวข้องกับการจัดทำโครงการในครั้งนี้
2. ศึกษาข้อมูลเกี่ยวกับภาษา shell script ในการจัดทำโครงการนี้ ที่เลือกภาษานี้มาใช้งานเพราะว่า มีความเห็นว่าจะต้องทำงานบนระบบปฏิบัติการลินุกซ์
3. ศึกษา The Sun Studio performance tools เป็นเครื่องมือชุดคำสั่งในการทดสอบประสิทธิภาพของซอฟต์แวร์
4. ศึกษาขั้นตอนการทำงานของระบบเดิมของการทดสอบประสิทธิภาพของซอฟต์แวร์ เพื่อนำไปพัฒนาระบบงานใหม่

### 3.1.2 ขั้นตอนการวิเคราะห์ระบบ

- วิเคราะห์ระบบเดิม

ทุกๆ ครั้งที่ผู้พัฒนาระบบหรือโปรแกรมเมอร์นั้นได้มีการเขียนโปรแกรมให้ได้ออฟต์แวร์ตามที่คาดหวังไว้เสร็จแล้ว ก็จะส่งซอฟต์แวร์ไปให้ฝ่ายทดสอบระบบ (Tester) เพื่อทำการทดสอบประสิทธิภาพของซอฟต์แวร์ โดยทางผู้ทดสอบระบบจะทำการทดสอบซอฟต์แวร์ทุกๆ 2 เดือน และข้อมูลที่เป็นผลลัพธ์จากการทดสอบนั้นเป็นข้อมูลโดยภาพรวมของซอฟต์แวร์ (Black box) โดยไม่ได้แยกการทดสอบออกเป็นแต่ละฟังก์ชัน ทำให้ทางผู้พัฒนาระบบหรือโปรแกรมเมอร์นั้นได้ข้อมูลผลการทดสอบที่ล่าช้า และยังได้ข้อมูลโดยรวมซึ่งทำให้เสียเวลาในการปัญหาของซอฟต์แวร์นั้นๆ

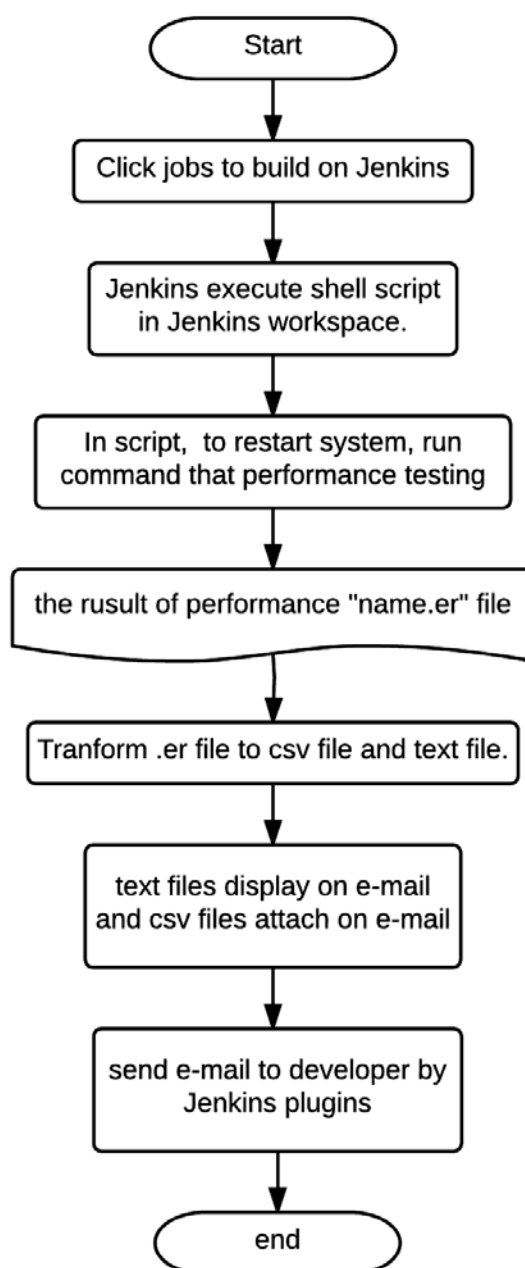
- วิเคราะห์ระบบใหม่

โดยจากการศึกษาระบบเดิมนั้น ทำให้เกิดแนวทางการแก้ปัญหาโดยการที่ทุกๆ ครั้งที่ผู้พัฒนาได้เขียนโปรแกรมเสร็จและได้มีการ commit โค้ดไปที่ Version control จะมีการตรวจสอบว่าโค้ดที่ระบบซอสโค้ดกลางมีการเปลี่ยนแปลงหรือไม่ หากมีการเปลี่ยนแปลง จะเริ่มการทดสอบซอฟต์แวร์นั้นโดยทันที และจะแจ้งผลการทดสอบผ่านทางจดหมายอิเล็กทรอนิกส์ โดยผลการทดสอบนั้นจะแยกไปตามฟังก์ชัน เพื่อให้ทางผู้พัฒนาได้มองเห็นถึงปัญหาของซอฟต์แวร์อย่างตรงจุด และสามารถแก้ไขปัญหาดังกล่าวได้อย่างรวดเร็ว

ประโยชน์ของการพัฒนาการทดสอบประสิทธิภาพของซอฟต์แวร์

- 1) เพิ่มความรวดเร็วในการทราบถึงผลการทดสอบซอฟต์แวร์ ทำให้การแก้ไขปัญหาของซอฟต์แวร์เป็นไปได้อย่างมีประสิทธิภาพ
- 2) ผลการทดสอบเป็นผลการทดสอบที่แยกตามฟังก์ชันการทำงาน ทำให้สามารถแก้ไขปัญหาดังกล่าวได้ตรงจุด

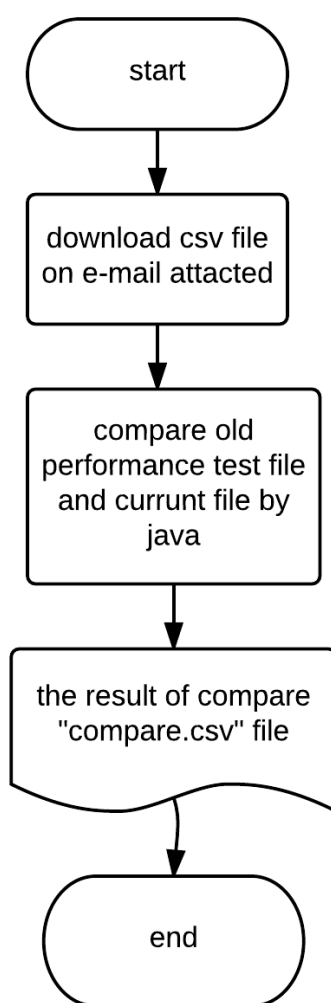
- 3) เพิ่มความสะดวกในการทำงานของทางผู้พัฒนา แลพเพิ่มความถูกต้องให้กับชิ้นงาน



รูปที่ 3.13 แผนภาพแสดงการทำงานของงานของการพัฒนาการทดสอบประสิทธิภาพของซอฟต์แวร์

จากรูปที่ 3.13 การทำงานการทดสอบระบบใช้แนวคิดของการบูรณาการอย่างต่อเนื่องมาเป็นแนวทางในการพัฒนา โดยใช้เครื่องมือคือเจนกินส์ โดยให้เจนกินส์เริ่มการทำงานโดยการเฝ้าขอสโตร์กลางของซอฟต์แวร์ หากมีการเปลี่ยนแปลง จะเริ่มกระบวนการทันที หรือ สามารถที่จะคลิกเลือก

ว่าจะทดสอบตอนไหนก็ได้ เริ่มกระบวนการโดยเจเนกิ้นส์จะไปเรียกให้เซลล์สคริปต์ที่เขียนไว้ทำงาน โดยภายในจะมีการให้ เตรียมระบบของซอฟต์แวร์ก่อนโดยการ Stop และ Start ระบบ จากนั้นก็ใช้ ชุดคำสั่งของเครื่องมือ The Sun Studio performance tools เพื่อเป็นการประมวลผลของแต่ละ ฟังก์ชัน เมื่อได้ผลลัพธ์การประมวลผลแล้ว จะทำการแปลงไฟล์ข้อมูลให้อยู่ในรูปแบบไฟล์ข้อความ ให้ สามารถเปิดอ่านได้จากเครื่องคอมพิวเตอร์ และจะส่งผลลัพธ์ไปยังผู้พัฒนาด้วยจดหมาย อิเล็กทรอนิกส์



**รูปที่ 3.14** แผนภาพแสดงการเปรียบเทียบผลลัพธ์ของการทดสอบประสิทธิภาพของ ซอฟต์แวร์ระหว่างรุ่นเก่าและรุ่นใหม่

จากรูปที่ 3.14 เมื่อมีการทราบผลของการทดสอบประสิทธิภาพของซอฟต์แวร์แล้ว หากต้อง เพิ่มความถูกต้องให้กับระบบอีก ต้องทำการเปรียบเทียบผลลัพธ์ระหว่างการทดสอบครั้งที่แล้วกับการ ทดสอบปัจจุบัน เพื่อจะได้ทราบว่าฟังก์ชันที่เราแก้ไขไปนั้นดีขึ้นหรือว่าแย่ลง

โดยการพัฒนาการทดสอบประสิทธิภาพของซอฟต์แวร์นั้นอยู่ในการวิจัยและพัฒนา (The research and development) คือ เป็นการศึกษาและวิจัยระบบของการทดสอบประสิทธิภาพของซอฟต์แวร์อยู่ เนื่องจากซอฟต์แวร์ที่นำไปใช้ในการทดสอบประสิทธิภาพนั้นเกิดปัญหาค้าง ซึ่งทางพีทีดูแล ยังไม่สามารถแก้ไขได้ ฉะนั้นการพัฒนาการทดสอบประสิทธิภาพของซอฟต์แวร์จึงจำเป็นต้องหยุดพัก การพัฒนาไว้เท่านี้ หากเมื่อทางพีทีดูแลสามารถแก้ไขปัญหาของซอฟต์แวร์ได้ งานวิจัยชิ้นนี้จะ เป็นประโยชน์ต่อทางทีมพัฒนาเป็นอย่างมาก