1. ส่วนตัวคิดว่า agile กับ waterfall ดีคนละแบบ แต่agileจะสะดวกมากกว่า เพราะสามารถเปลี่ยนแปลงได้ตลอด ทำงานตามความถนัดของคนทำ แก้ไขได้ง่าย waterfallจะมีระเบียบ ขั้นตอนแบบแผนที่เยอะกว่า

2. อาจจะไม่ขนาดนั้น แต่gitและgithub ใช้งานง่ายกว่า CVSหรือSVN

3. git branch -m old\_branch new\_branch

git push origin new\_branch:refs/heads/new\_branch

git checkout --orphan new\_empty\_branch

git reset –hard

4. ก็อาจจะเป็นไปได้ แต่ มีวิธีแก้ไขคือ ทำการ edit แล้ว commit ไปใหม่

5. abcdesh-4.3$

6. ก็อาจจะเป็นความคิดส่วนบุคคล แต่web applicationใช้งานได้ง่ายและสะดวกกว่า

7. **Rails** เป็น Web Framework มีลักษณะเป็น MVC (Model-View-Controller) พัฒนาโดย 37signals

Rails ถูกออกแบบมาให้มีการใช้งานที่ง่ายและรวดเร็ว ลดปัญหางานทีี่ต้องทำซ้ำๆ ทำให้ได้ productivity

ที่สูงขึ้น สอดคล้องกับ Methodology แบบ Ajile และมีแนวความคิดพื้นฐานคือ Don't Repeat Yourself (DRY) อะไรที่อยู่แล้ว ก็ไม่มีความจำเป็นต้องทำสิ่งนั้นซ้ำ Reuseable นั้นเอง

Convention over configuration คือข้อตกลงที่ Rails สร้างขึ้นเป็นแบบแผน ที่กำหนดมา

ขอเพียงเราทำตามรูปแบบเหล่านั้นผลลัพธ์จะออกมาตามที่เราต้องการ โดยที่ไม่จำเป็นต้อง

ใช้การ configuration สำหรับคนที่ใช้ Convention ได้อย่างคล่องแคล่วและถูกต้องแล้วจะทำให้

การพัฒนาระบบได้เร็วขึ้นเพราะไม่ต้องมาทำการ configuration ให้ยืดยาว แต่มีข้อดีก็ต้องมีข้อเสียครับ เพราะ Convention ถ้าเป็นมือใหม่หัดเขียนคงต้องศึกษาและทำความเข้าใจกันมากซักหน่อย และมันอาจสร้างความประหลาดใจให้คุณได้

**MVC (Model-View-Controller)** ก่อกำเนิดขึ้นมาตั้งแ่ต่ปี 1979 นานโขแล้วเลยทีเดียว คิดขึ้นโดย

Trygve Reenskaug ใช้กับ Smalltalk แนวคิดเป็นที่ยอมรับในวงกว้างมี Framework ในยุคหลังๆ หยิบยืมแนวความคิดนี้ไปใช้เช่น Struts, Spring, Tapestry, CakePHP และอีกมากมาย รวมถึง Rails

เพราะ Web Application Framework ตระกูลหลักๆ ในตอนนี้ยืดแนวคิดนี้เป็นพื้นฐานทั้งนั้น

MVC เป็นลักษณะสถาปัตยกรรมที่มีการแบ่งการทำงานออกเป็น 3 Layer คือ

**Model** เป็นภาชนะเก็บข้อมูลหรือสถานะการทำงาน แต่ Model เป็นหม้อที่วิเศษกว่านั้น เพราะนอกจากจะเก็บข้อมูลยังสามารถทำการตรวจสอบความถูกต้องของข้อมูลได้อีกด้วย

**View** เป็นส่วนการแสดงผลเป็น User Interface ที่ User จะได้พบเจอ โดยอาจจะนำ Model มาแสดงผลในส่วนนี้ และ Userแต่่ล่ะคนอาจที่จะใช้ข้อมูลกลุ่มเดียว แต่การแสดงผลอาจจะต่างออกไป เช่น 1 Model มี 2 View

**Controller** เป็นส่วนการเชื่อมโยงการทำงาน เป็นส่วนที่เลือกว่าจะใช้ Model อันไหน หรือ View มาทำงาน เพื่อตอบสนองการทำงานได้ถูกต้องตาม User ต้องการ

8. **ข้อดีของ Ruby On Rails**  
- Code เขียนง่าย ไม่ซับซ้อน เมื่อเทียบกับ java  
- ถ้า Code สั้น ดังนั้นโอกาสในการเกิด Bug จึงน้อย  
- เป็น Open source  
- มี Library ให้นำไปใช้เป็นจำนวนมาก  
- ง่ายต่อการนำไปขยายและพัฒนาต่อ (extend)  
- เป็น pure Object Oriented้

**ข้อเสียของ Ruby On Rails**- ผู้พัฒนาต้องการเรียนรู้ตามโครงสร้างของ Rails

**ข้อดีของFramework**

1. มีธรรมเนียมการเขียนที่ชัดเจน โปรแกรมเมอร์คนเก่าลาออก คนใหม่เข้ามารู้ Framework ตัวนั้นเขียนต่อได้เลย
2. ลดเวลาในการสร้าง Create, Read, Update และ Delete หรือ CRUD เพราะ Framework ส่วนใหญ่มีระบบ Generator ให้ พูดง่ายๆ ว่า Gen Code Insert Update Delete ให้ได้เลย ลดเวลาในการเขียนส่วนนี้ไปอีก
3. มีระบบในการเชื่อมต่อไปยังฐานข้อมูลได้มากกว่า 1 ชนิด เช่น เชื่อมต่อไปยัง MySQL, MS SQL, ProgreSQL, SQLite และอื่นๆ ได้หลากหลาย (ทั้งนี้ขึ้นกับความสามารถของแต่ละ Framework)

**ข้อเสียของ Framework**

1. มี Class จำนวนมากมาย เพื่อให้การทำงานอย่างเป็นระบบ ทำให้ต้องเขียน Class ที่เป็นตัวช่วยให้ผู้พัฒนาใช้ แต่บาง Class อาจไม่จำเป็น
2. ใช้เวลาในการเรียนรู้และศึกษาธรรมเนียมของการเขียนโปรแกรมใน Framework นั้นๆ (แต่ระยะหลังนี้มีการใช้มาตรฐาน PSR ในการพัฒนา ต่อไปเรียนรู้ Framework ตัวเดียวแต่สามารถเขียนได้ทุก Framework)

9. Heroku คือ Platform as a Service (Paas) ที่ให้เราใช้งานได้ฟรี (มีแบบเสียเงินด้วย) โดยรองรับภาษาโปรแกรมที่หลากหลาย เช่น Ruby, PHP, Node.js, Python, Java, Clojure, Scala และยังสามารถสร้าง buildpack สำหรับภาษาอื่นๆได้ เช่น Lua ที่รันอยู่บน OpenResty ได้อีกด้วย

มีบทบาทกับการพัฒนา web application อย่างไร Heroku เหมาะกับใครบ้าง ? จริงๆแล้ว มันเหมาะกับทุกคนแหละ เช่น นักศึกษาอยากลองเขียนเว็บด้วย PHP แต่ไม่ได้เช่า Hosting ก็สามารถใช้ Heroku ได้ หรือแม้แต่บริษัท Start up ที่ไม่อยากวางเครื่องเอง คอนฟิกเอง ก็ใช้ได้ เพราะมันสามารถ scale ให้รองรับผู้ใช้เยอะๆได้โดยง่าย

10. เพราะว่าทางคณะวิทยาการสารสนเทศต้องการให้นิสิตสาขาเทคโนโลยีสารสนเทศมีความรู้เบื้องต้นเกี่ยวกับวิศวกรรมซอฟต์แวร์ กระบวนการพัฒนาซอฟต์แวร์ การบริหารโครงการซอฟต์แวร์กระบวนการวิศวกรรมความต้องการ แบบจำลองระบบ การออกแบบ การสร้างซอฟต์แวร์ การทดสอบ การตรวจสอบความถูกต้อง ตัวชี้วัดซอฟต์แวร์ การประกันคุณภาพซอฟต์แวร์ การจัดการและควบคุมการเปลี่ยนแปลงในการพัฒนางานด้านซอฟต์แวร์ การบำรุงรักษาซอฟต์แวร์