

Final_part_1

1. ก็เห็นด้วยทั้ง 2 ทาง หลักการทำ แบบ **agile** คือ ทำงานย่อยๆ ส่งให้ลูกค้าได้ดูงานของเราได้เรื่อยๆ แต่ก็มีเรื่องข้อจำกัดเข้ามา แบบ **waterfall** ก็ดีทำงานตามลำดับไปเรื่อยๆ มีแบบแผน แต่เสียเรื่อง ลูกค้าไม่สามารถดูงานของเราได้บ่อยๆ ต้องรอให้ **software** เสร็จและทดสอบ ทีเดียวจึงจะส่งมอบให้ลูกค้าไป

2. **centralized version system(cvs)** ยังไม่สูญหายไปหรอก จะใช้วิธีให้วิธีให้คนทำงานเสร็จแล้วจะส่งข้อมูลไปเก็บไว้ที่ **Server** กลางซึ่ง **server** กลางตัวนี้ก็จะสำรองข้อมูลไว้ด้วย ถ้าใครอยากได้ข้อมูลรุ่นไหนก็สามารถมาดึงข้อมูลออกไปได้จาก **server server** กลางโดยทั่วไปแล้วจะเป็นรุ่นใหม่ล่าสุด และทดสอบว่าใช้งานได้แล้ว ระบบแบบนี้มีข้อดีคือ การทำงานเก็บไว้ที่ศูนย์กลางเดียวกัน คนที่เป็น **project manager** ก็ดูได้ง่ายว่าทำงานกันไปถึงไหนแล้ว (ความคืบหน้า) ไฟล์ข้อมูลก็ใหม่อยู่เสมอ และทุกคนสามารถมีไฟล์ข้อมูลเหมือนกันได้ง่าย เพียงแค่ดึงไฟล์จาก **server** ไว้ในเครื่องตัวเอง

3. คำสั่ง

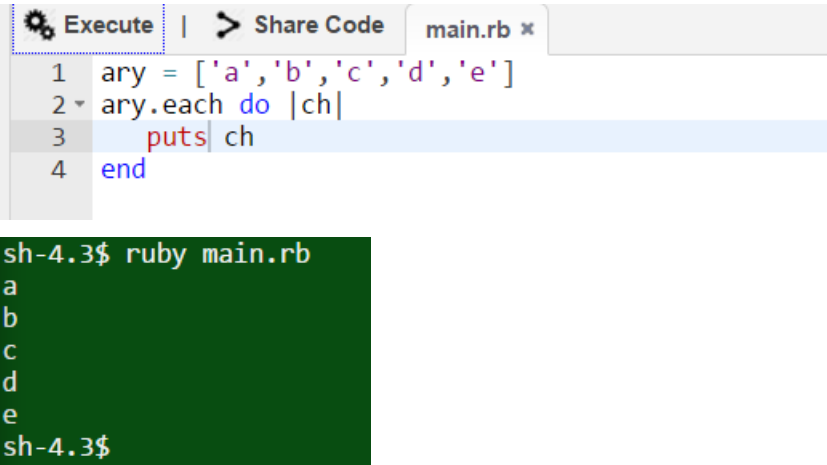
3.1 git add -A

3.2 git commit -m " feature1 "

3.3 git checkout -b 'feature1'

3.4 git push origin master

4. ทีมหรือหัวหน้าจะต้องตั้งกฎว่า ใครจะ merge ขึ้นไปในเวลาไหน เพื่อที่จะไม่ให้เกิดเหตุการณ์ **conflict** เกิดขึ้นจากเหตุการณ์ในข้อนี้แสดงว่า ทีมของเขานั้นไม่ได้ตั้งกฎกันเอาไว้ ทำให้เจอ **conflict** บ่อย

5. The image shows a screenshot of a code editor with a tab labeled 'main.rb'. The code is as follows:

```
1 ary = ['a', 'b', 'c', 'd', 'e']
2 ary.each do |ch|
3   puts ch
4 end
```

Below the code editor, there is a terminal window showing the command 'sh-4.3\$ ruby main.rb' and its output, which is the characters 'a', 'b', 'c', 'd', and 'e' on separate lines, followed by the prompt 'sh-4.3\$'.

6. คนที่ทำ web application ไม่ได้ตามแฟชั่น เหตุที่ทำไมเพราะในยุคปัจจุบัน internet สามารถเข้าถึงได้ทุกวัย และทุกคนก็มี smart phone ที่มี internet ทำให้ programmer หันมาทำ web application กันมากขึ้น เพราะทุกคนเข้าถึงมันได้ ส่วน นำ software มาใส่ไว้ใน cd ฟังดูเหมือนจะติดอยู่ในยุคเก่า เพราะยุคนั้น internet ยังไม่แพร่หลายมากกว่าในปัจจุบัน เหตุที่ทำไม software ใส่ cd นั้นก็มีข้อดีอยู่เหมือนกัน programmer อาจจะทำ software ที่ไม่ต้องการใช้ internet ก็ดีกันได้ เพราะกลัว hacker ล้วงข้อมูลเอาไปได้ และอาจจะเป็นเพราะ software ของเขานั้นสำคัญมาก ๆ จึงไม่สามารถเอา software ติดต่อผ่าน internet

7. User คลิกรายการหนึ่ง หรือ user เปิด browser เข้าเว็บครั้งแรก จากนั้น mvc ใน rails ก็ไปติดต่อกับ rails router rails router ก็ไปเรียก ฟังก์ชัน index ใน controller ถ้าในฟังก์ชัน index ใน controller มีข้อมูลใน db controller ก็ไปเรียก model ฟังก์ชันใน model ก็ไปติดต่อกับ databases แล้ว return หรือ ส่งข้อมูลกลับมาให้ controller controller ก็ไปเรียก view view ก็ส่งผลลัพธ์มาให้ controller controller ก็ส่งผลลัพธ์ ไปที่ browser ของ user html

8. ส่วนตัวผม ได้ลองใช้ CI หรือ codeigniter framework ถ้าเทียบกับ rails framework ส่วนตัวผมคิดว่า ci ค่อนข้างใช้ง่ายกว่า rails framework เพราะ ใช้หลัก mvc ตรงๆ ส่วน rails framework อาจยุ่งยากสักเล็กน้อย

ข้อดี

1. Ci เขียนง่าย ใช้ภาษาค่อนข้างนิยม ส่วน rails framework มาใหม่ ภาษาใหม่ ต้องเรียนรู้สักกระยะหนึ่งแล้วจะดีเอง
2. ส่วนตัวคิดว่า ci ทำงานได้รวดเร็วกว่า rails framework เพราะว่า rails framework ใช้คำสั่งใน command คือ rails server ซึ่งกว่าจะ Run ได้ ก็นาน

ข้อเสีย

1. ไม่มี คำสั่ง หรือ รูปแบบคล้ายๆ rails test ใน rails framework ทำให้ประสิทธิภาพในเรื่องการทดสอบ ci ดีกว่า
 2. Rails framework เป็นภาษาใหม่ ค่อนข้างเติบโตและค่อนข้างใช้งานง่ายพอๆกับ ci
 3. rails framework มีเครื่องมือช่วยเยอะและดี เช่น heroku และ github ทำให้ในเรื่องการทำงานเป็นทีม ค่อนข้างมีประสิทธิภาพดีกว่าเมื่อเทียบกับ ci framework
 4. rails framework สามารถใช้ heroku ได้เป็น server ขั้นตอนการทำความง่าย ส่วน ci ต้องมี server เป็นของตัวเองหรือจำลองserver ขึ้นมาเอง ทำให้ rails framework ดีกว่า
 9. heroku คือ platform as a service (Paas) ที่ให้เราใช้งานได้ฟรีหรือเสียตัง
- บทบาทกับการพัฒนา web application คือ heruko จะมี platform ไม่ว่าจะจะเป็น windows osx linux ทำให้การทำงานจะมีประสิทธิภาพดีและสามารถใช้ node.js ได้
10. เพราะว่าคณะเห็นความสำคัญของวิชา 887342 Software Development Process ว่าเมื่อเด็กนิสิตจบไป ในเรื่องการทำงาน programmer จะต้องทำงานเป็นทีมและงานที่ทำอาจจะเป็นงานที่ใหญ่ จึงจัดวิชานี้ให้กับเด็ก ในหลักสูตร คำอธิบายในหลักสูตรก็ชัดเจนแล้วว่า กระบวนการพัฒนา software จะต้องทำยังไง อะไรบ้าง

คำอธิบายจากหลักสูตร

ความรู้เบื้องต้นเกี่ยวกับวิศวกรรมซอฟต์แวร์ กระบวนการพัฒนาซอฟต์แวร์ การบริหารโครงการซอฟต์แวร์ กระบวนการวิศวกรรมความต้องการ แบบจำลองระบบ การออกแบบ การสร้างซอฟต์แวร์ การทดสอบ การตรวจสอบความถูกต้อง ตัววัดซอฟต์แวร์ การประกันคุณภาพซอฟต์แวร์ การจัดการและควบคุมการเปลี่ยนแปลงใน การพัฒนางานด้านซอฟต์แวร์ การบำรุงรักษาซอฟต์แวร์

Software engineering overview, software processes, software project management, software requirements and specification, software design, software testing and validation, software metrics, software quality assurance, software configuration management, and software maintenance