1.ตอบ อาจกล่าวได้ว่า Agile Method ไม่มีอะไรที่เหมือนกับ Waterfall Model เลย แต่ Waterfall Model ก็ยังเป็นวิธีการพัฒนาระบบที่ใช้กันอยู่ค่อนข้างแพร่หลาย ซึ่งวิธีนี้จะมีลำดับขั้นตอนที่ตายตัว เริ่มตั้งแต่รวบรวมข้อมูล กำหนดความต้องการของผู้ใช้ วิเคราะห์ทางเลือก ออกแบบ เขียนโปรแกรม ทดสอบระบบ และสุดท้ายทำการติดตั้งระบบ โดยแต่ละส่วนของขั้นตอนดังกล่าวจะถือเป็นตัววัดความก้าวหน้าของงาน ปัญหาสำคัญของ Waterfall Model คือขั้นตอนของการพัฒนาที่ไม่ยืดหยุ่น เพราะตัวงานจะแบ่งเป็นช่วงๆแบบตายตัว ทำให้มีข้อผูกมัดตั้งแต่เริ่มโครงงานและไม่สามารปรับเปลี่ยนความต้องการผู้ ใช้ได้ หมายความว่าการพัฒนาโดยใช้ Waterfall Model นั้น ไม่เหมาะกับงานที่ความต้องการของผู้ใช้เข้าใจยาก และมีแนวโน้มว่าจะเปลี่ยนแปลงตลอดเวลา

ในทางกลับกัน Agile Method จะแบ่งงานออกเป็นส่วนย่อยๆ แล้วค่อยๆ ทดสอบไปเรื่อยๆทุกสัปดาห์หรือทุกสองสัปดาห์ ทั้งนี้จะเน้นสร้างส่วนย่อยที่สุดของงานทั้งหมดที่สามารถใช้งานได้ทีละชิ้น เพื่อให้ส่งมอบได้รวดเร็วและทำการปรับปรุงเพิ่มเติมอย่างต่อเนื่องตลอดช่วง เวลาของโครงงาน

2.ตอบ มีข้อดีข้อเสียแตกต่างกันออกไป Git คือ Version Control ตัวหนึ่ง ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ในโปรเจ็คเรา มีการ backup code ให้เรา สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชั่นต่างๆของโปรเจ็คที่ใด เวลาใดก็ได้ หรือแม้แต่ดูว่าไฟล์นั้นๆใครเป็นคนเพิ่มหรือแก้ไข หรือว่าจะดูว่าไฟล์นั้นๆถูกเขียนโดยใครบ้างก็สามารถทำได้ ฉะนั้น Version Control ก็เหมาะอย่างยิ่งสำหรับนักพัฒนาไม่ว่าจะเป็นคนเดียวโดยเฉพาะอย่างยิ่งจะมีประสิทธิภาพมากหากเป็นการพัฒนาเป็นทีม

การทำงานแบบcvsนี้มีประโยชน์เหนือ local VCS ในหลายด้าน เช่น ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ

แต่ระบบแบบนี้ก็มีจุดอ่อนเหมือนกัน ตรงที่การรวมศูนย์ทำให้มันเป็นจุดอ่อนจุดเดียวที่จะล่มได้เหมือนกันเพราะทุกอย่างรวมกันอยู่ที่เซิร์ฟเวอร์ที่เดียว ถ้าเซิร์ฟเวอร์นั้นล่มซักชั่วโมงนึง หมายความว่าในชั่วโมงนั้นไม่มีใครสามารถทำงานร่วมกันหรือบันทึกการเปลี่ยนแปลงงานที่กำลังทำอยู่ไปที่เซิร์ฟเวอร์ได้เลย หรือถ้าฮาร์ดดิสก์ของเซิร์ฟเวอร์เกิดเสียขึ้นมาและไม่มีการสำรองข้อมูลเอาไว้ คุณก็จะสูญเสียข้อมูลประวัติและทุกอย่างที่มี จะเหลือก็แค่ก๊อปปี้ของงานบนเครื่องแต่ละเครื่องเท่านั้นเอง

3.ตอบ

git branch feature1

git push -u origin feature1

git checkout master

git pull origin master

4.ตอบ

พบว่ามีฟ้อง conflict และส่งผลให้ merge ไม่ผ่าน ตอนนี้ถ้าดูสถานะจะพบว่า

$ git status

# On branch master

# Unmerged paths:

# (use "git add/rm <file>..." as appropriate to mark resolution)

#

# both modified: hello.py

#

no changes added to commit (use "git add" and/or "git commit -a")

หมายเหตุว่า Git จะไม่ยอมให้ย้าย branch ถ้าหากยังมี conflict เช่นนี้อยู่ ดังนั้นมาเก็บข้อผิดพลาดนี้กัน โดยเปิดไฟล์ hello.py ขึ้นมา จะเห็นดังนี้

import sys

names = sys.argv[1:]

if '-h' in names:

exit('usage: python hello.py [-h] [NAME [NAME ...]]')

<<<<<<< .merge\_file\_trt1cP

elif '-v' in names:

exit('advance hello beta')

=======

elif '-l' in names:

exit('license under WTFPL v2.0')

>>>>>>> .merge\_file\_XPZY4N

if not names:

print('Hello, world!')

if '-s' in names:

names.remove('-s')

names.sort()

for name in names:

print('Hi {}.'.format(name))

ส่วนที่เกิด conflict จะถูกคั่นด้วยบรรทัดที่ขึ้นต้นด้วยเครื่องหมาย <<<<<<<, =======, >>>>>>> ในตัวอย่างนี้ จัดการลบ 3 บรรทดนั้นทิ้งไปก็พอครับ (สำหรับงานจริง เปิดอ่านโปรแกรมแล้วจัดการเรียบเรียง code ใหม่ให้ทำงานถูกต้องนะครับ)

ตอนนี้อาจจะทดสอบเพิ่มกันอีกเล็กน้อย เมื่อมั่นใจว่าโปรแกรมทำงานถูกต้องแล้ว ก็สั่ง commit ความเปลี่ยนแปลงนี้ครับ (ไม่ใช่สั่ง merge ไปอีกรอบนะครับ -- ตัว Git จะรู้ได้เองว่านี่เป็นการแก้ conflict ที่เกิดจากการ merge)

$ git commit -am 'manual merge show-version and show-license'

[master 388a013] manual merge show-version and show-license

ถึงตอนนี้ถ้ายังไม่มั่นใจว่าทั้ง 2 branch นั้นถูก merge แล้วจริงหรือเปล่า สามารถดูด้วย branch --no-merged ซึ่งควรจะไม่แสดงผลลัพท์ออกมา (เพราะถูก merge เรียบร้อยแล้ว) และถ้าสั่ง branch --merged ก็ควรจะเห็นครบ

5. ตอบ 6 7 8

6.ตอบโดยส่วนตัวแล้วชอบ web app มากกว่า เอาแผ่นcdมาลงเพราะมันทำให้เสียเวลาและค่าใช้จ่าย แต่ข้อดีของcdก้มีคือไม่ติดไวรัสแต่ถ้าโหลดจากแอฟแล้วล่ะก็ต้องมีไวรัสมาอย่างแน่นอน

7. ตอบ

Rails framework

- Form data validation (การตรวจสอบข้อมูลใน form)

- Application template (การสร้าง template ด้วย CSS?)

- Email sending and receiving (การจัดการ รับ-ส่ง email)

- Date and time formatting and manipulation (รูปแบบของวัน – เวลา)

- Session and cookie management (การจัดการ session , cookie )

- สนับสนุนการทำงาน Web 2.0 ด้วย AJax

• Model

ใน object oriented การใช้เว็บ database-driven จะเป็นแบบ MVC ซึ่ง Model จะประกอบด้วย class ที่เชื่อมต่อกับ RDBMS ใน Ruby On Rails class model จะถูกจัดการผ่านทาง Active Record (เป็นตัวเข้าถึงข้อมูลในฐานข้อมูล) ซึ่งโปรแกรมเมอร์ทุกคนควร ต้องทำเป็น subclass คือ ActiveRecord?::Base class และโปรแกรมจะเข้าใจอัตโนมัติว่าจะใช้ตาราง RDBMS อันไหน และเรียกคอลัมภ์ต่างๆในตารางเอง

ใน Model มีการติดต่อกับ Active Record เพื่อช่วยจัดการงานด้าน Database เช่น

ดูแลในเรื่องของการติดต่อสื่อสารระหว่าง Object และ Database โดยที่ผู้พัฒนาไม่ต้องยุ่งยากกับการใช้ SQL command

เป็นงานด้านการตรวจสอบความสัมพันธ์ของข้อมูล มีผลกับฐานข้อมูล

Handles validation(ตรวจสอบความถูกต้อง), association(ความสัมพันธ์ระหว่างฐานข้อมูล), transactions, and more…

• View

- เป็นส่วนที่ต้องแสดงผลผ่าน web browser

- เขียนด้วยพื้นฐานของ HTML(.rhtml), แทรกด้วย script ของ ruby คล้าย PHP,JSP,ASP

- การทำงานสัมพันธ์อยู่กับ controller

- นำ component มาใช้ใหม่ได้ (Reusable)

- สนับสนุน Ajax

- View เป็นการแสดงผลทาง logic หรือ การทำอย่างไรให้ข้อมูลจาก Controller class ถูกแสดงผล วิธีการใน Rails จะใช้ Embedded Ruby (ไฟล์นามสกุล .rhtml) ซึ่งก็เป็นพื้นฐานจาก HTML และด้วยไวยากรณ์ (syntax) ที่คล้าย JSP นอกจากนี้ด้วยยังสนับสนุนการใช้ HTML และ XML

- สำหรับ method ที่อยู่ใน class ของ controller หากต้องการที่จะแสดงผลแก่ผู้ใช้ จึงจำเป็นต้องเขียน code ย่อยขึ้นมา และเก็บในโฟลเดอร์ของ view นี้เอง โดยจะต้องตั้งชื่อไฟล์นี้ เป็นชื่อเดียวกันกับ method ใน controller ที่ต้องการให้มี output ในการแสดงผล เช่น

- ใน controller mysite มีการกำหนด method ที่มีชื่อว่า

index,home,contact เป็นต้น โดยทั้ง 3 method ต้องมีการแสดงผลต่างกัน ดังนั้น

โปรแกรมเมอร์ต้องเขียนไฟล์ในการแสดงผลใน โฟลเดอร์ view 3 ไฟล์ ได้แก่

index.rhtml , home.rhtml , contact.rhtml เป็นต้น

- นอกจากนี้ยังสามารถกำหนด stysheet และ template เพื่อให้งานเว็บแอพลิเคชั่นนั้นมี

มาตรฐานเดียวกันทั้งหมด ในกรณี template จะสร้างไฟล์ ที่เป็นตัวกำหนด header,

content, footer ไว้ที่โฟลเดอร์ layout ภายใต้โฟลเดอร์ view ซึ่งลักษณะการทำ

งานของไฟล์นี้ จะถูกเรียกใช้ในการแสดงผลทุกครั้ง เป็นต้น

• Controller

- เป็นส่วนที่ทำงานเป็นอันดับแรกเมื่อมีโปรแกรมถูกเรียก จาก Web browser

- เป็นส่วนที่ติดต่อการทำงานระหว่างผู้ใช้และโปรแกรม

- มีการติดต่อกับ Database(ฐานข้อมูล) ด้วย Model และแสดงผลข้อมูลผ่านทาง View

- เป็นส่วนที่มีการประมวลผลหลัก ของโปรแกรม

• การทำงานของ Ruby On Rails

1.Client หรือผู้ใช้ร้องขอข้อมูลทาง Web site

2.Web server รับการร้องขอในรูปแบบต่างๆ ผ่านทาง HTTP,RSS,ATOM หรือ SOAP

3.Web server ส่งต่อไปยังไฟล์ที่ชื่อว่า dispatcher ที่อยู่ใน Rails ซึ่งเป็นไฟล์สำหรับโหลด Controller ให้ทำงาน

4.ไฟล์ Dispatcher ทำงาน โดยจะโหลดตัว Controller ที่อยู่ใน Rails ขึ้นมา

5.Controller เป็นส่วนแรกของการทำงานแบบ MVC ติดต่อกับ Model และ View หน้าที่คือการประมวลผลหลักของโปรแกรม ซึ่งการทำงานขึ้นอยู่กับสิ่งที่ผู้ใช้ร้องขอมา เช่น

• ถ้าผู้ใช้ร้องขอ ข้อมูลใน Database

- Controller จะติดต่อไปยัง Model

- Model ก็จะมีตัว Active Record สำหรับจัดการทุกอย่างที่เกียวข้องกับ

Database

- สุดท้ายก็จะส่งข้อมูลที่ได้จาก Model กลับไปให้ Controller

- Controller ส่งผลลัพธ์กลับไปยังผู้ใช้ผ่าน View

• ถ้าผู้ใช้ ร้องขอข้อมูลในรูปแบบของ Webservice

- Controller จะมีการทำงานโดยใช้ Action Web service

- และส่งผลลัพธ์กลับไปยังผู้ใช้ผ่าน View

• ถ้าเป็นในรูปแบบของ Mail

- Controller จะใช้ Active mailler สำหรับจัดการ การรับ-ส่ง

Email

- Refresh ตัวมันเอง

6.สุดท้ายข้อมูลต่่างๆ ที่ผ่านการประมวลผลใน Controller แล้ว จะถูกส่งมายังส่วนของ View

7.View ส่งข้อมูลกลับมายัง Web browser เพื่อแสดงผลกับผู้ใช้งาน ซึ่งรูปแบบการแสดงผลก็ได้แก่ HTML,CSS,Javascript,Image,XML เป็นต้น

• การสนับสนุนต่างๆของ Ruby On Rails

- Database

- MySQL

- PostgreSQL

- SQLite

- SQL Server

- DB2

- Oracle

- OS

- Windows

- Unix/Linux

- Mac OS X

8.ตอบ การสร้างเว็บด้วยajax **Rails** เป็น Web Framework มีลักษณะเป็น MVC (Model-View-Controller) พัฒนาโดย [37signals](https://37signals.com/)  
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

• สรุปข้อดีของ 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 Oriented

9.ตอบ

ในตอนนี้ใครทำแอป หรือเว็บแอปแล้ววางเครื่องเซิร์ฟเวอร์เองบ้าง ก็อาจจะยังมีอยู่ครับ แต่หลายๆคน อยากลดเวลาในการพัฒนาแอปโดยตั้งเป้าหมายไปที่การพัฒนาแอปเลย ไม่ต้องมาตั้งค่าเครื่องเซิร์ฟเวอร์เองให้เปลืองเวลา เพราะอาจจะไม่มีทรัพยากรณ์มากพอ ทั้งในด้านบุคคลและอื่นๆ อยากจะมุ่งเน้นไปที่พัฒนาโปรแกรมเพียงอย่างเดียว

Heroku เป็น 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

**ข้อควรระวัง เมื่อทํางานกับ Heroku**

* Heroku ใช้ Git ในการ push ขึ้นไปที่ Repository มัน แต่มันรับเฉพาะ master branch เท่านั้นน่ะ เหมือนจะไม่รับตัวอื่นนะ
* ถ้าเราทํา Node.js มันจะมี ให้ระบุ Port ใช่ไหมล่ะ แนะนําใช้ ยังงี้จะได้เป็น dynamic นิดๆ ไม่ static ไป( process.env.PORT || 5000 ) ส่วน Node client ก็ใช้ DNS แทนน่ะ
* MongoHQ ไม่ให้ สิทธิ์ Admin กับเราหรอกน่ะ เพราะฉะนั้นคําสั่งของ MongoDB จะถูกลดไปเยอะเช่น show dbs เป็นต้นจ้า ใช้อื่นๆแทนน่ะ
* ถ้าจะดู Logs ก็ใช้คําสั่ง*heroku logs -t*น่ะ จะได้เห็นว่า application เราเป็นไงบ้าง เพียงแต่ต้องเข้าไปใน folder app เราก่อนน่ะ

10.ตอบ

นิสิตรู้จักการเขียนโค้ด Software มีความสำคัญเกี่ยวข้องกับชีวิตเรามากขึ้นเรื่อยๆ ไม่ว่าจะเกี่ยวข้องกับวัตถุในชีวิตประจำวันมนุษย์เช่นโทรศัพท์ เครื่องซักผ้า รถยนต์ รถไฟฟ้า หรือเกี่ยวกับธุรกิจเช่นธนาคาร ดังนั้นการสร้าง Software ต้องสร้างให้ได้คุณภาพดี ไว้ใจได้นักศึกษากล้านั่งรถไฟฟ้าที่ตนเองเป็นคนเขียน Software ควบคุมการเบรคหรือไม่ ปัญหาการสร้าง Software ในปัจจุบันยังเป็นวิกฤต คือไม่น่าเชื่อถือ ทำงานไม่ถูกต้องไม่ได้ตามที่คาดไว้ สร้างไม่สำเร็จ ไม่ทันตามกำหนดเวลา งบประมาณ หรือต้องล้มเลิกกลางคันจำนวนมาก ยกตัวอย่างเช่น ระบบควบคุมการส่งยานอวกาศ Ariane 5 ของยุโรปต้องกดระเบิดยานทิ้งเพราะวิถีการเคลื่อนที่ผิดพลาดมาก ระบบจัดการรถพยาบาลของLondon (London Ambulance) ที่ทำงานไม่ถูกต้องและต้องยกเลิกการใช้หลังเสียเงินไปจำนวนมาก