ข้อ 1. สำหรับผมแล้วคิดว่าในบทความข้างต้น อาจเป็นเรื่องที่ถูก ว่า การทำ แบบ Agile มันเป็นการทำแบบไม่ได้มีเอกสารกำกับที่แน่นอน เพราะกับทำแบบ Agile มันก็คือการ พัฒนาไปเรื่อยๆนั่นเอง มันเหมาะสำหรับ การพัฒนาระบบแบบที่ ใกล้ชิดลูกค้าคือพัฒนาเรื่อยๆ มีอะไรที่ลูกค้าต้องการ ก็สามารถพัฒนาให้เข้ากับการใช้งานได้เรื่อยๆ จนกว่าจะส่งมอบ

ต่างกับแบบ water fall เพราะกับพัฒนาแบบ waterfall นั้นเป็นการลงมือทำและพัฒนาทีเดียวให้เสร็จเลย มีเอกสารกำกับอย่างเห็นได้ชั เละลูกค้าไม่ต้องเข้าไปดูเรื่อยๆเหมือนแบบ Agile มันก็ใช้ได้ครับเป็นการเลือกที่ด้วย มันเหมาะกับการพัฒนาที่ลูกค้าอยู่ไกล ส่วนตัวผมคิดว่า มันก็ดีทั้ง สองอย่างแหละครับ เพราะไม่ว่าเราจะ ใช้ waterfall หรือ agile งานก็ออกมาเสร็จเหมือนกัน แต่มันก็ดีคนละแบบอย่างเห็นได้ชัดครับ สรุปก็คือ มันก็ดีทั้งสองอย่างแหละครับแค่เราเลือกใช้ให้ถูกกับประเภทงานเป็นพอเราก็จะได้งานที่ดีและสมกับคุณภาพที่ออกมาครับ

ข้อ 2. จากบทความข้างต้น ผมเห็นด้วยนะครับ เพราะ ณ ขณะนี้ ก็ใช้ Git และ Github ผมคิดว่ามันได้เปรียบกว่าพวก CVS SVN ก็ตรงที่มันทันสมัยมากกว่า แต่ก็ไม่อาจว่าได้เต็มปากหลอกครับเพราะ ตอนนี้ก็ยังมีคนใช้ CVS SVN อยู่ . การใช้ Git, Github การใช้ Git ก็ไม่ได้ใช้ยากมากจนเกินไปครับ

Contralized เป็นแบบรวมศูนย์กลางserverไว้ที่เดียว ข้อดีคือ แต่ละคน merge ในแต่ละส่วนก่อนได้ และลดปัญหาคอขวด ข้อเสียฃคือ versionใช้ได้ใน V. เดียว

Distributed เป็นแบบกระจาย จุดเด่น : ไม่มีคอหนา , versionไม่นิ่ง , แต่ละคนมี repo เป็นของตัวเอง

ข้อ 3. สร้างrepo ใน Github ชื่อว่า feature1

วิธีการcommit ใช้ คำสั่งดังนี้ครับ

* git init
* git add -A
* git commit -m "first commit for feature1"
* git remote add origin https://github.com/57160449/ feature1.git
* git push -u origin master

ข้อ 4. ผมคิดว่าจากบทความข้างต้นการที่รุ่นพี่บอกแบบนั้นคงเป็นการเตือนๆนั่นแหละครับ แต่การที่เกิดเหตุการณ์นี้อาจเป็นตัวรุ่นพี่ทำงานไม่ clean รึเปล่า อย่างไรก็ตาม ก่อนทำแบบนั้น จำไว้ว่า working directory หรือ staging area ของเรามีความเปลี่ยนแปลงที่ยังไม่โดน commit ซึ่ง conflict กับ branch ที่เรากำลัง checkout Git ก็เลยไม่ปล่อยให้ switch branches ถ้าจะให้ดี ควรจะมีสถานะการทำงาน cleanๆ ก่อนที่จะ switch branches

การ Merge กับ การแก้ไข conflict เมื่อไหร่จะมีการ Merge จะเกิดเมื่องานที่ branch เสร็จแล้ว ก็จะ Merge กลับไปที่ main กรณีการ Conflict เมื่อโปรแกรมเมอร์ทำการแปะโค้ด 2 ส่วนไว้ที่เดียวกัน เมื่อ Merge ก็จะเกิด Conflict การแก้ปัญหา Conflict นั้นทำอย่างไร มีอยู่ 2 แบบ แบบที่1. คุยกันว่าจะเอาแบบไหน แบบที่2. ให้ project manager ตัดสิน

ข้อ 5. จากคำสั่งในภาษา ruby

('a'..'e').each {|ch| print ch} จะแสดงผลลัพธ์นี้ครับ abcde“a”..“e”

ข้อ6. จากบทความข้างต้น ผมคิดว่า จริงๆ การทำ web application นั้นมันดีกว่าการทำsoftwareลงในCDนะครับ มัน ไม่ค่อยจะสะดวกสบายสักเท่าไร และยังล้าหลังด้วยแต่มันก็มีข้อดีของมันก็คือ ปลอดภัยจากข้างนนอกแต่ก็ต้องเฏ็บรักษาแผ่นด้วยนะครับ จริงๆการทำ software ในรูปแบบของบ application นั้นมันดีนะครับ ตรงที่ประหยัดการใช้ทรัพยากร อย่างเช่นเราต้องซื้อแผ่นเพื่อเอาไปลง software มันก็สิ้นเปลืองเงินค่าแผ่น การทำบน webapplicationยอมได้เปรียบกว่าอยู่แล้วครับแต่มันก็เสี่ยงเรื่องความปลอดภัยนิดหน่อยครับ 55+ ทั้งยังสะดวก แต่โดยรวมก็ดีผมคิดว่าwebappดีกว่าด้วยซ๊ำ

ข้อ 7. มาอธิบายในส่วนของ Model คือ ส่วนที่เราเอาไว้ใช้ทำงานกับ ข้อมูลดิบ ๆ มันจะทำงานเมื่อ Controller ต้องการข้อมูล มันก็จะดึงข้อมูลส่งไปให้ Controller ไปจัดการ ถ้านึกภาพเล่น ๆ ก็เหมือนกับ บรรณารักษ์ ที่คอยเฝ้า, จัดเรียง และสามารถค้นหาหนังสือ ที่เราต้องการได้

Views คือ พริตตี้สาว สวย หมวย ที่คอยเป็นหน้า เป็นตาให้กับโปรแกรมเรา บางครั้ง เราก็จะเรียกมันว่า User Interface เพราะมันเป็นส่วนที่ใช้ติดต่อกับผู้ใช้นั่นเอง

Controller คือ กรรมกรหนุ่มล่ำบึก ที่คอยเป็น เบ้ และ มันสมอง ให้กับโปรแกรม โดยส่วนนี้จะรับหน้าที่ในการจัดการโปรแกรมเรา เช่นการประมวลผล หรือรับข้อมูลเป็นต้น และเป็นส่วนที่เราจะเก็บพวก Logic ของโปรแกรมเราไว้ในนี้เลย

ขั้นตอนตามภาพก็มีดังนี้

1. เริ่มจาก Client (users)ส่ง Request ไปที่ rails
2. ซึ่งจะถูกส่งต่อให้ Controller    
   ทำการตรวจสอบข้อมูลที่มาให้ (Request Method, Request Parameters)
3. แล้ว Controller จะเรียก Method ให้ทำงานเพื่อจัดการ Request นั้น
4. มันก็จะไปติดต่อข้อมูลมาจากDatabase

5.Model จะทำการคำนวณและอาจติดต่อกับ Database เพื่อจัดการกับ Request    
นั้น แล้วส่งผลลัพธ์กลับไปที่ Controller

6. เมื่อ Controllerได้ผลลัพธ์จาก Model แล้วก็ใช้ผลลัพธ์นั้นส่งต่อให้ View ทำงาน

7. View จะสร้าง Page สำหรับแสดงผลลัพธ์นั้นแล้วส่ง page กลับไปที่ Controller

8. Controller ส่ง Page นั้น (เป็น Response) กลับไปยัง Client (users)

ข้อ 8. ถ้าจะให้ยกตัวอย่างการใช้framworkก็คงหนีไปพ้น Eclipse เพราะใช้เรียนภาษา java มาตั้งแต่ปี1 มันต่างกันยังไง ระหว่าง rails กับ framework rails พัฒนาบนinternetก็ได้ ส่วน framework เป็นโปรแกรมที่ต้อง install ติดตั้ง rails ดีตรงที่ สะดวก สบาย ไม่ต้องลง โปรแกรมใดๆ ข้อเสียคือ ต้องเชื่ออมต่อกับ internet และบางสถานะการณ์ก็ คอมไพร ไม่ค่อยดี

ส่วน framework นั้น ดีตรงที่ เสถียร ข้อเสียคือ อาจจะช้าในบางโอกาส

ข้อ 9. Heroku คือตัวที่ทำให้เราสามารถทำwebapplicationได้ในรายวิชานี้ มันมีบทบาทมากในการทำเพราะต้องใช้herokuเกือบทุกขั้นตอน

ข้อ 10. วัตถุประสงค์ก็เพื่อสอนในราบวิชา 887342 ให้นิสิตได้รู้ถึงการพัฒนาsoftwareว่า ณ ขณะนี้โลกของเรามี โปรแกรมที่ใช้ทำอะไรบ้าง สอนให้ใช้ github heroku ได้เข้าใจในการ ใช้git การcommit รวมถึงการสร้าง story map ในการวางแผนว่าควรจัดทำแผนในการดำเนินงานอย่างไร