# final\_part\_1

1.

มันเหมือนไม่มีแบบแผน แต่จริง ๆแล้วมันมี แต่ไม่เข้มงวดเท่านั้นเอง เพราะลูกค้าส่วนใหญ่นั้น ไม่รู้แน่ชัดว่าตนเองต้องการอะไรกันแน่ ผมคิดว่าการพัฒนาแบบ agile นั้น ทำให้ software ของเราตรงสเปคที่ลูกค้าที่ต้องการมากที่สุด และยังการพัฒนา software ที่เป็นธรรมชาติมากกว่าด้วย เพราะ agile นั้นเอื้อให้สามารถเกิดการเปลี่ยนแปลงได้ตลอดเวลา แต่เพื่อจะทำแบบนั้น ลูกค้าเองก็จำเป็นต้องมาคุยกับเราบ่อย ๆ เพราะเราไม่รู้แน่ชัดว่าลูกค้าต้องการอะไรกันแน่ แต่ถ้าลูกค้าคนนั้นรู้ความต้องการของตนเองแน่ชัด และไม่สะดวกมาพบเราบ่อย ๆ แบบนั้นก็น่าจะเหมาะกับ waterfall มากกว่า

2.

ผมคิดว่าทั้งสองอย่างนั้น ล้วนมีข้อดีข้อเสียแตกต่างกัน พวก CVS หรือ SVN นั้นก็ดีในเรื่องของการจัดการที่ง่าย Github ก็ดีในเรื่องของการประหยัดทรัพยากร ไม่ต้องให้ Server ทำงานหนัก เพราะฉะนั้นสิ่งที่ผมคิดว่าคืออนาคตที่แท้จริงคือ การทำข้อดีของทั้งสองอย่างมารวมกัน

3.

git checkout -b feature1

git commit -a -m “add feature1”

git push origin feature1

4.

จากที่ฟังมาผมคิดว่าโครงการของพี่ต้องมีทำงานเยอะมากแน่ ๆ ผมว่า conflict เกิดจากการแบ่งงานที่ยังไม่ชัดเจนเท่าไหร่ จึงทำให้คนในทีมเกิดไปยุ่งกับงานของคนอื่น แล้วเวลานำงานมารวมกัน(merge) จึงเกิด conflict ส่วนวิธีแก้ปัญหานั้น แนะนำว่าให้เลือกตามความเหมาะสม อาจจะประชุมหารือ แล้วลงมติกันว่าจะใช้โค้ดของใคร หรืออาจจะตกลงกันเลยว่าให้โค้ดของหัวหน้านั้นมีความสำคัญที่สุด อันนี้ก็แล้วแต่ว่าทีมนั้นเหมาะกับวิธีแบบไหน ส่วนวิธีป้องกัน คือให้แบ่งงานกันให้ชัดเจนไปเลย อาจจะมีการทำคอมเม้นไวในโค้ดว่าตรงนี้งานของใคร และห้ามไปยุ่งกับโค้ดคนอื่นเด็ดขาด

5.

abcde

6.

ผมเข้าใจเรื่องความเป็นส่วนตัวของคุณ เพราะสมัยนี้มันช่างเป็นสิ่งที่หายากเหลือเกิน อย่างไรก็ตามผมเห็นว่า web application มีข้อดีอยู่มากมาย ไม่ต้องติดตั้ง แจกจ่ายได้ง่าย ใครก็ใช้ได้ คุณไม่ต้องอัพเดทโปรแกรม แถมคุณยังได้ใช้โปรแกรมรุ่นใหม่ตลอดด้วย ซึ่งโปรแกรมรุ่นใหม่นั้น ก็จะปรับปรุงในเรื่องของประสิทธิภาพ รวมถึงความปลอดภัย ความเป็นส่วนตัว แถม web application เดียวนี้ ก็เข้ารหัสเป็นส่วนมาก เรื่องความเป็นส่วนตัวไม่น่าจะน่ากังวลเท่าไหร่ ส่วนเรื่องการทำงานแบบออฟไลน์นั้นก็มี progressive web app มาแล้ว ทำให้สามารถใช้แบบออฟไลน์ได้ ถ้าหากบอกว่าเป็นการตามแฟชั่น ผมก็คิดว่านี่เป็นแฟชั่นที่ดีต่อทุกคนจริง ๆ

7.

1. ผู้ใช้(Google Chrome) ร้องขอ url ที่ /users จากนั้นคำร้องจะถูกส่งไปที่ตัวจัดการเส้นทางของ rails
2. เมื่อได้รับคำขอมาแล้วตัวจัดเส้นทางจะส่งหน้าที่ต่อให้ controller ซึ่งก็คือผู้ควบคุมว่าจะจัดการอย่างไรกับคำขอนั้น
3. Controller รู้ว่า model user สามารถทำหน้าที่นี้ได้ จึงส่งต่อหน้าที่ให้
4. Model user ซึ่งเปรียบเสมือนพนักงานระดับล่าง จึงไปค้นหาข้อมูลในฐานข้อมูล เมื่อได้มาแล้วจึงส่งกลับให้ผู้จัดการ ซึ่งก็คือ controller
5. Controller ได้ข้อมูลมาแล้ว แต่จะต้องนำไปแสดง controller รู้ว่าคนที่ทำแบบนี้ได้ก็คือ view
6. Controller ได้ส่งข้อมูลให้ view ซึ่งเปรียบเสมือนพนักฝ่ายวาดรูป
7. view จะข้อมูลที่ได้จาก controller มาสร้างหน้าเว็บใหม่ จากนั้นจะส่งกลับให้ controller
8. ในที่สุด controller ก็ได้ข้อมูลจนครบ จากนั้นจึงนำไปแสดงผลให้ผู้ใช้ดู

8.

ผมเคยใช้ framework codeigniter ของ php สำหรับความแตกต่างนั้น ในด้านของ design pattern ไม่ต่างกัน เพราะ codeigniter ก็เป็น mvc เหมือนกัน แต่ถ้าโดยรวมแล้วผมว่า rails มีความสามารถมากกว่า ข้อดีของ codeigniter คือ ใช้ง่ายเพราะเป็นภาษา php ที่คุ้นชินอยู่แล้ว และทำงานได้เร็ว ตัวไฟล์ไม่ใหญ่มาก ข้อเสีย ไม่สามารถสร้าง rest url crud แบบรวดเร็วแบบ rails ได้ ซึ่งนี่ก็คือข้อดีของ rails

9.

Heroku คือ Platform as a Service (Paas) หรือบริการ web hosting ช่วยให้เราสามารถมี Server ได้โดยไม่ต้องซื้อเครื่อง Server มาเอง ซึ่งเกี่ยวข้องกับการพัฒนา web application ต้องที่เราจะต้องนำ web application ของเรานั้น deploy ขึ้น server เพื่อให้คนอื่นสามารถใช้งานได้ ซึ่งผู้ที่จะมาทำหน้าที่ตรงนี้ก็คือ Heroku นั้นเอง

10.

เพื่อให้นิสิตสามารถไปทำงานด้านพัฒนา software ได้ราบรื่นยิ่งขึ้น เนื่องจากงานพัฒนา software จริง ๆ นั้นโดยส่วนมากจะทำงานเป็นทีม และต้องมีการคุยกับลูกค้า จึงต้องมีระบบ หรือกระบวนใดกระบวนการหนึ่ง มาช่วยในการบริหารจัดคนในทีม เพื่อให้ทุกคนรู้งานของตนเอง เมื่อมีปัญหาเกิดขึ้น จะได้แก้ปัญหาได้ เมื่อเจอกับงานพัฒนา software ที่ใหญ่ ๆ ทำให้สามารถแบ่งงานกันทำได้ และจะทำให้งานเสร็จเร็วขึ้น