**Final\_part\_1**

**1.**ตอบ สำหรับข้าพเจ้าคิดว่าการทำ Agile ถือเป็นทางเลือกที่ดีอีกทางหนึ่ง เพราะ Agile Method เป็นวิธีการพัฒนาซอฟท์แวร์แบบตรงกันข้ามกับวิธีที่เน้นการวางแผนแบบเป็นระบบ และมีระเบียบ การระบุเช่นนี้อาจจะทำให้เกิดการเข้าใจผิดว่า Agile Method ไม่มีการวางแผนและไม่มีระเบียบ ซึ่งจริงๆแล้วไม่ใช่  Agile Method เน้นความสามารถในการปรับเปลี่ยนได้อย่างรวดเร็วตามความเป็นจริงที่เปลี่ยนแปลงอยู่เสมอ เมื่อความต้องการของผู้ใช้เปลี่ยนแปลงไป ทีมงานพัฒนาก็ต้องพร้อมที่จะรับการเปลี่ยนแปลงตามไปด้วย Agile Method ไม่มีอะไรที่เหมือนกับ Waterfall Model เลย แต่ Waterfall Model ก็ยังเป็นวิธีการพัฒนาระบบที่ใช้กันอยู่ค่อนข้างแพร่หลาย ซึ่งวิธีนี้จะมีลำดับขั้นตอนที่ตายตัว เริ่มตั้งแต่รวบรวมข้อมูล กำหนดความต้องการของผู้ใช้ วิเคราะห์ทางเลือก ออกแบบ เขียนโปรแกรม ทดสอบระบบ และสุดท้ายทำการติดตั้งระบบ โดยแต่ละส่วนของขั้นตอนดังกล่าวจะถือเป็นตัววัดความก้าวหน้าของงาน ปัญหาสำคัญของ Waterfall Model คือขั้นตอนของการพัฒนาที่ไม่ยืดหยุ่น ซึ่งการทำงานที่ไม่ยืดหยุ่นอาจจะไม่เหมาะกับพวกนักออกแบบหรือนักเขียนโปรแกรมมากนัก

**2.** ตอบ ในส่วนตัวของข้าพเจ้า ข้าพเจ้าคิดว่า git และ github ถือเป็นทางเลือกที่น่าสนใจในการทำเรื่อง version control เพราะใช้งานง่ายและสะดวกกว่าพวก centralized ทั้ง SCCS, RCS, CVS, SVN ที่กล่าวทั้งหมดนั้นใช้ central code repository model คือหมายความง่ายๆ ว่า changes ของ file ทั้งหมดถูกเก็บอยู่ที่ central repo หรือศูนย์กลางที่เดียว ถ้าสมมติว่า central repo เกิดพังขึ้นมาหรือผู้ใช้ไม่สามารถติดต่อกับ central repo ได้ก็จะทำให้ไม่สามารถดึงไฟล์ที่พึ่งเปลี่ยนแปลงมาทีตัวเองได้

ส่วน distributed version control ที่ใช้ใน BitKeeper และ git นั้น แต่ละคนจะมี copy ของไฟล์ตั้งแต่เริ่มแรกจนท้ายสุด ดังนั้นแต่ละทีมหรือแต่ละคนสามารถ maintain code ได้โดยที่ทุกคนมี copy ของไฟล์ทั้งหมดอยู่ที่ local computer และ change set ของแต่ละ version ที่เกิดขึ้นจากหลายๆ ทีมนั้นสามารถรวมกันได้ ไม่จำเป็นต้องรอ changes copy จาก central repo อีกต่อไป ดังนั้น distributed version control จึงเร็วกว่า และผู้ใช้งานก็สามารถแก้ได้ทุกที่ โดยไม่จำเป็นจะต้องติดต่อกับ central repo ก็ได้ แกก็ไม่แน่เสมอไปว่าพวก centralized version control อย่าง CVN หรือ SVN จะสูญพันธ์ เพราะวิธีการใช้งานแต่ละประเภทล้วนมีข้อดีและข้อเสียที่แตกต่างกันออกไป แล้วแต่ความสะดวกของผู้ใช้หรือผู้เขียนโปรแกรมมากว่า

**3.**

**4.** ตอบ conflict : จะเกิดขึ้นได้ก็ต่อเมื่อโปรแกรมเมอร์สองคนทำงานในส่วนเดี่ยวกัน ต้องการ merge พร้อมกัน ซึ่งเราสามารถแก้ไข conflict ได้ง่ายๆคือ ต้องคุยกันว่าจะเอาโค้ดเเบบไหนดี , หรือให้โปรเจคเมเนเจอร์ตัดสินเลยว่าจะใช้โค้ดไหน

**5.**

**6.** ตอบ สำหรับข้าพเจ้าชอบการทำ Web Application มากกว่า เพราะข้อดีคือ ข้อมูลต่าง ๆ ในระบบมีการไหลเวียนในแบบ Online ทั้งแบบ Local (ภายในวงLAN) และ Global (ออกไปยังเครือข่ายอินเตอร์เน็ต) ทำให้เหมาะสำหรับงานที่ต้องการข้อมูลแบบ Real Time ระบบมีประสิทธิภาพ แต่ใช้งานง่าย เหมือนกับท่านทำกำลังท่องเว็บ ระบบงานที่พัฒนาขึ้นมาจะตรงกับความต้องการกับหน่วยงาน หรือห้างร้านมากที่สุด ไม่เหมือนกับโปรแกรมสำเร็จรูปทั่วไป ที่มักจะจัดทำระบบในแบบกว้าง ๆ ซึ่งมักจะไม่ตรงกับความต้องการที่แท้จริง ระบบสามารถโต้ตอบกับลูกค้า หรือผู้ใช้บริการแบบ Real Time ทำให้เกิดความประทับใจ เครื่องที่ใช้งานไม่จำเป็นต้องติดตั้งโปรแกรมใด ๆ เพิ่มเติมทั้งสิ้น หมายคนอาจจะคิดต่างกัน เพราะในส่วนตัวใครหลายคนอาจไม่ชอบการทำ web application การนำ CD มาลงในเครื่องอาจจะสะดวกมากกว่า

**7.** กลไกของ MVC ใน Rails

ตอบ 1. เริ่มจาก Client ส่ง Request ไปที่ Web App ซึ่งจะถูกส่งต่อให้ Controller ทำการตรวจสอบข้อมูลที่มาให้ (Request Method, Request Parameters)

2. แล้ว Controller จะเรียก Method ให้ทำงานเพื่อจัดการ Request นั้น

3. Model จะทำการคำนวณและอาจติดต่อกับ Database เพื่อจัดการกับ Request นั้น แล้วส่งผลลัพธ์กลับไปที่ Controller

4. เมื่อ Controller ได้ผลลัพธ์จาก Model แล้วก็ใช้ผลลัพธ์นั้นส่งต่อให้ View ทำงาน

5. View จะสร้าง Page สำหรับแสดงผลลัพธ์นั้น แล้วส่ง page กลับไปที่ Controller

6. Controller ส่ง Page นั้น (เป็น Response) กลับไปยัง Client

หลักการสำคัญก็คือโมเดลจะไม่สามารถเรียกเมดธอดของวิวหรือคอนโทรลเลอร์โดยตรง เนื่องจากมันจะไม่มีตัวแปรเก็บออปเจคที่เป็นวิวหรือคอนโทรลเลอร์ โมเดลมีเพียงแต่ list ของออปเจคชนิด observer (วิวหรือคอนโทรลเลอร์ที่ implement observer interface) ที่เฝ้าสังเกตุการเปลี่ยนแปลงอยู่ ตัวโมเดลจะแจ้งการเปลี่ยนแปลงด้วยการส่ง event notification message ไปให้ observer ทุกๆตัว ถ้า observer นั้นเป็นวิวก็จะอัพเดปหน้าจอด้วยขอมูลใหม่ เป็นต้น ในทำนองเดียวกันตัววิวเองก็มองไม่เห็นคอนโทรลเลอร์ ไม่สามารถเรียกเมดธอดของคอนโทรลเลอร์ได้โดยตรง นอกจากจะส่ง event ไปให้คอนโทรลเลอร์ที่เป็น observer ของวิวนั้นๆ

**8.** ตอบ framework ไม่ต้องเขียนโค้ดเองทำให้คนอื่นทำงานต่อได้ง่าย

ข้อดีของ Framework

* มีธรรมเนียมการเขียนที่ชัดเจน โปรแกรมเมอร์คนเก่าลาออก คนใหม่เข้ามารู้ Framework ตัวนั้นเขียนต่อได้เลย
* ลดเวลาในการสร้าง Create, Read, Update และ Delete หรือ CRUD เพราะ Framework ส่วนใหญ่มีระบบ Generator ให้ พูดง่ายๆ ว่า Gen Code Insert Update Delete ให้ได้เลย ลดเวลาในการเขียนส่วนนี้ไปอีก

ข้อเสียของการไม่ใช้ Framework

* การจัดการเรื่องระบบ Template ที่ต้องการเปลี่ยน Title ของ Web Application ตามเนื้อหา และหน้าตาของระบบทั้งหมด หากมีการเปลี่ยนแปลงต้องไล่แก้ Code กันให้วุ่น
* ระบบการจัดการกับข้อมูลต้องมาเขียน Class ต่างๆ ในการจัดการ หรือถ้าตอนนี้ดีหน่อยที่มี PDO มาช่วย
* เราเขียน Code ไว้สำเร็จระบบสวยงามทำงานได้ แต่พอเอาให้คนอื่นพัฒนาต่อ แฮะๆ เขาจะบอกว่า "ผมเขียนใหม่ดีกว่า"
* หากมีการเขียนกันหลายคนต้องมานั่งวางระเบียบและธรรมเนียมให้ตรงกัน ไม่งั้นเพื่อนๆ จะบอกว่าเอะ "อันนี้คืออะไร มันมาจากไหน"

Rails เป็น Web Framework มีลักษณะเป็น MVC (Model-View-Controller) พัฒนาโดย [37signals](https://37signals.com/)  
Rails ถูกออกแบบมาให้มีการใช้งานที่ง่ายและรวดเร็ว ลดปัญหางานทีี่ต้องทำซ้ำๆ ทำให้ได้ productivity  
ที่สูงขึ้น สอดคล้องกับ Methodology แบบ Agile

**9.** ตอบHeroku 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 ได้อีกด้วยโดยการจะ push ไปที่ heroku นั้น ทาง heroku เองจะมี  [toolbelt](https://toolbelt.heroku.com/) เป็น Heroku Client มาให้เราใช้ ซึ่งจะมีทุก platform เลย ไม่ว่าจะเป็น (window ,osx ,linux) แล้วใน toolbelt มันจะมี foreman ติดตั้งมาด้วย ซึ่งมันจะทําหน้าที่เป็น config file ว่าจะให้  Heroku ทํางานยังไง และ มองตัวไหนเป็นตัวที่มันจะเป็นไฟล์หลักทำให้การพัฒนา web application สะดวก รวดเร็ว และใช้งานง่ายขึ้น

**10.** ตอบ เพราะวิชานี้ถือเป็นอีกวิชาหนึ่งที่สามารถนำความรู้ไปต่อยอดในการทำงานหรือในอนาคตข้างหน้าได้จริง Software มีความสำคัญเกี่ยวข้องกับชีวิตเรามากขึ้นเรื่อยๆ ไม่ว่าจะเกี่ยวข้องกับวัตถุในชีวิตประจำวันมนุษย์เช่นโทรศัพท์ เครื่องซักผ้า รถยนต์ รถไฟฟ้า หรือเกี่ยวกับธุรกิจเช่นธนาคาร ดังนั้นการสร้าง Software ต้องสร้างให้ได้คุณภาพดี ไว้ใจได้ เข้าใจการทำงานของโปรแกรมระบบมากขึ้น มีการทำงานเป็นทีม และต้องบริหารควบคุมความก้าวหน้าของทีม ของโครงการพัฒนาซอฟต์แวร์ ต้องเข้าใจความต้องการของผู้ใช้ให้ชัดเจน ถูกต้อง กำหนดขอบเขตให้ชัดเจน