**1.**

**Agile** เป็นหลักการในการพัฒนา software แบบใหม่ที่เน้นว่าใครถนัดอะไร และการพูดคุยสื่อสารกัน มากกว่า การยึดติดที่เครื่องมือและกระบวนการ เช่นเปลี่ยนให้โปรแกรมเมอร์ไปคุยกับลูกค้าแทน ลูกค้าบอกอะไรมาก็ทำตามนั้นได้เลยให้ทำงานโดยยึดที่ผลผลิตหรือ software เป็นหลัก เช่น เดิมเน้นเอกสารแต่ Agile ไม่สนมากนัก แต่สนที่ว่าเรามี SW หรือของส่งให้ลูกค้าหรือยังให้ความสำคัญเรื่องของการติดต่อสื่อสาร เช่น เดิมมีสัญญาหรือ contact กันแต่ Agile ไม่สนใจ ให้มองที่ความสัมพันธ์ระหว่างผู้พัฒนาและลูกค้ายอมรับความเปลี่ยนแปลง เช่น เดิมต้องวางแผนให้ครบเป็นอย่างดี และทำตามแผน (Gantt chart) ให้ได้ แต่ Agile ไม่ต้องทำตามแผนแต่เน้นการสนองความเปลี่ยนแปลงที่เกิดขึ้นได้

**Waterfall model** คือกระบวนการพัฒนาระบบซอฟต์แวร์ที่เป็นขั้นเป็นตอนในรูปลักษณะของชั้นน้ำตก หรือ ขั้นบันได นับจากบนสุดไปยังชั้นล่างสุดแต่ละชั้นจะกินระยะเวลานานมากเหมาะสำหรับระบบซอฟต์แวร์ที่มีความต้องการที่แน่นอนชัดเจน ไม่คลุมเคลือ เพราะแต่ละขั้นตอนจะไม่มีการวนกลับมาทำซ้ำอีกในขั้นที่สูงกว่า แต่บางกรณีที่แบบจำลองตัวนี้ถูกพัฒนาให้สามารถวนกลับมาทำซ้ำขั้นตอนข้างบนได้ชั้นนึง

**วิจารณ์**

จากกระบวนการพัฒนาซอฟแวร์ทั้งสองแบบที่ได้กล่าวมานี้ต่างก็มีข้อดีข้อเสียต่างกันไป อยู่ที่ตัวของผู้พัฒนาเองว่าจะสนใจในรูปแบบไหน อย่าง **Agile** จะเน้นในด้านการสื่อสาร ทำให้โปรแกรมเมอร์สามารถทราบได้ถึงความต้องการของลูกค้า เวลาในการแก้ไขความผิดพลาดก็จะน้อยลง แต่ข้อเสียก็คือ การทำงานที่ไม่ค่อยเป็นระบบ แต่ **Waterfall** จะเน้นการทำงานที่เป็นขั้นตอนทำให้ตรวจสอบความผิดพลาดได้ง่าย แต่มีข้อเสียคือ ใช้เวลาในการทำงานนาน

**2.**

**Git VS SVN**

การ pull โปรเจคลงมาจาก repository  
SVN จะเป็นการ checkout โปรเจคลงมา ซึ่งหมายถึงว่า source code ที่เป็นเวอร์ชั่นล่าสุดจะถูกดึงลงมาไว้ที่เครื่องเรา  
Git จะเป็นการ clone คือ การดึงมาทั้ง repository ไม่ว่าจะมีกิ่งก้านสาขาอะไรก็ตาม ทุกอย่างที่อยู่บน repository จะถูกเอาลงมาทั้งหมด  
การ upload file สู่ repository  
สำหรับ SVN นั้น จะเรียกว่าการ commit และหลังจากที่ commit ไปแล้วไฟล์เหล่านั้นก็จะอยู่ใน repository ทันที แต่ถ้าเป็น   
Git การ commit จะเป็นการเก็บอยู่ใน local ก่อน ซึ่งเราสามารถ commit แต่ละส่วนของโปรแกรมทีละนิดๆ แล้วค่อย push ขึ้นสู่ remote repository ทีเดียวก็ได้  
ลำดับของ revision  
SVN จะใช้เลขจำนวนเต็ม (Integer) รันไปเรื่อยๆ ซึ่งแน่นอนว่า ยิ่งเรา commit บ่อยเท่าไร ก็จะทำให้เลข revision มีเยอะขึ้นๆ  
Git จะใช้ hash ด้วย SHA-1 hash algorithm เนื่องจาก git จะมีการ commit สู่ local repo. ก่อน เพราะถ้าใช้เป็นเลขรันไปเรื่อยๆเหมือน SVN ก็จะทำให้เกิดเลข revision ซ้ำกันได้  
การทำงาน Online / Offline  
SVN เป็น version control แบบ centralized หมายความว่า ทุกอย่างจะถูกเก็บในตัวกลาง ซึ่งก็คือ remote repo. ทั้งหมด ทำใช้งานจำเป็นต้องเชื่อมต่อ network ไปยัง repo. ไม่เช่นนั้นก็จะ commit หรือ check revision ก่อนๆ ไม่ได้เลย  
Git เป็น version control แบบ distributed คือ หลังจากที่ clone remote repo. มาแล้ว ก็เท่ากับว่าเรามี local repo. ที่เหมือนกันกับ remote repo. ด้วย จะ commit ก็สามารถทำได้แบบ offline เลยเพราะเป็นการ commit ไปที่ local repo. แต่ถ้าเราจะ push ไปไว้ที่ remote repo. แน่นอนว่าต้องเชื่อมต่อ network เช่นเดียวกันครับ

**วิจารณ์**  
ประเด็นหลักๆ อยู่ที่การออกแบบที่แตกต่างกันระหว่าง Centralized กับ Distributed ซึ่งแต่ละประเด็นมันก็แตกมาจากตรงนี้ ตัวไหนดีกว่า ตัวไหนเจ๋งกว่า อยู่ที่ปัจจัยหลายอย่าง วิธีการนำไปใช้ จำนวนคนในทีม ปริมาณงานที่ต้องแก้ไขร่วมกัน และอื่นๆ ยังไงการใช้งานของทั้ง 2 ตัวนี้ก็ยังเป็นที่นิยม แล้วแต่ว่าพวกเขาจะเลือกใช้กันแบบไหนเท่านั้นเอง

**3.**

**4.**

**วิจารณ์**

การชนกันของโค้ด เรียกด้วยภาษาทางเทคนิคเขาเรียก conflict เหตุการนี้จะเกิดขึ้นกับโปรเจ็กที่ร่วมกันทำหลายคนเท่านั้น ทำอยู่คนเดียวมันไม่เกิด เมื่อเกิด conflict โปรแกรมเมอร์จะต้องเป็นคนแก้ ซึ่งการ conflict เป็นปัญหาที่ยังไงก็ต้องเจอ แต่ก็ต้องหาทางแก้ไขกันไปให้ได้

**5.**

**6.**

**วิจารณ์**

ส่วนมากเรามักจะคุ้นเคยกับการใช้งานคอมพิวเตอร์ส่วนบุคคลที่ติดตั้ง โปรแกรมพวก Microsoft Office ที่ประกอบด้วย Word ที่สำหรับพิมพ์เอกสาร Excel สำหรับสร้างตารางคำนวณ โปรแกรมพวกนี้เราจะเรียกมันว่า Desktop Application ซึ่งจะติดตั้งบนเครื่องคอมพิวเตอร์ส่วนบุคคลเครื่องใครเครื่องคนนั้น หรือโปรแกรมสำหรับงานบัญชี ที่บางหน่วยงานติดตั้งที่เครื่องคอมพิวเตอร์เป็นลักษณะ Client-Server Application โดยเก็บฐานข้อมูลไว้ที่เซิร์ฟเวอร์ (Server) และติดตั้งตัวโปรแกรมบัญชีที่เครื่องใช้งาน (Client) ซึ่งตอบสนองความต้องการเพิ่มขึ้นในด้าน Multi-User หรือใช้งานพร้อมๆกันได้หลายๆคน โดยใช้ฐานข้อมูลเดียวกัน เก็บฐานข้อมูลไว้ที่ส่วนกลาง

เทคโนโลยี Desktop Application ไม่สามารถตอบสนองความต้องการการบริหารจัดการได้ โดยเฉพาะการทำธุรกิจที่ต้องปรับเปลี่ยนไปตลอดเวลา ข้อมูลมีการเคลื่อนไหวตลอดเวลา เพื่อตอบสนองภาวะตลาดที่แปรเปลี่ยน ระบบ Client-Server Application ตัวโปรแกรมมีความซับซ้อน การแก้ไข การ Upgrade ทำได้ยุ่งยาก อย่างกรณี หากต้องการ Upgrade หรือเพิ่มคุณสมบัติเพิ่มเติมให้กับ Application ที่ตัวเซิร์ฟเวอร์ต้องหยุดระบบทั้งหมด และเมื่อ Upgrade ที่เซิร์ฟเวอร์แล้ว ก็จำเป็นต้อง Upgrade ที่ Client ด้วย หากระบบมีผู้ใช้งานจำนวนมาก จะยิ่งเพิ่มความยุ่งยากมากขึ้น

นอกจากนี้ยังไม่รวมปัญหาว่า ที่เครื่อง Client มีความหลากหลายและแตกต่างกัน เช่น OS (Operating System) ที่ต่างกัน สเปคเครื่องที่แตกต่างกัน ซึ่งหากการ Upgrade แล้วมีความจำเป็นต้องใช้สเปคเครื่องที่สูงขึ้นที่ฝั่ง Client จำเป็นต้อง Upgrade ตัวเครื่องคอมพิวเตอร์ตามไปด้วย

จากตัวอย่างปัญหาเหล่านี้ ถูกจัดการด้วยเทคโนโลยี **Web Application** (เว็บแอพพลิเคชั่น) เพราะ Web Application สามารถตอบสนองปัญหาข้างต้นได้เป็นอย่างดี และสามารถแทนที่ Desktop Application ที่เป็น Client-Server Application ได้เป็นอย่างดี ตัวโปรแกรมของ Web Application จะถูกติดตั้งไว้ที่ Server คอยให้บริการกับ Client และที่ Client ก็ไม่ต้องติดตั้งโปรแกรมเพิ่มเติม สามารถใช้โปรแกรมประเภท Brower ที่ติดมากับ OS ใช้งานได้ทันที อย่าง Internet Explorer หรือโปรแกรมฟรี ได้แก่ FireFox, Google Chrome ซึ่งกำลังเป็นที่นิยมเป็นอย่างมาก ด้วยความสามารถของ Brower ที่หลากหลาย ทำให้ไม่จำกัดว่าเครื่องที่ใช้เป็น OS อะไร หรืออุปกรณ์อะไร อย่างอุปกรณ์ TouchPad หรือ SmartPhone ก็สามารถเรียกใช้งานได้ ลดข้อจำกัดเรื่องสถานที่ใช้งานอีกด้วย

จุดเด่นอีกอย่างหนึ่ง คือข้อมูลที่ส่งหากัน ระหว่าง Client กับ Server มีปริมาณน้อยมาก ทำให้เราสามารถย้ายเซิร์ฟเวอร์ไปอยู่บนเครือข่าย Internet ได้ และสามารถใช้งานผ่าน Internet Connection ที่มีความเร็วต่ำๆได้ จุดเด่นนี้ทำให้ สามารถใช้ Application เหล่านี้จากทุกๆแห่งในโลกได้

**7.**

**Controller**  
•เป็นส่วนที่ทำงานเป็นอันดับแรกเมื่อมีโปรแกรมถูกเรียก จาก Web browser  
•เป็นส่วนที่ติดต่อการทำงานระหว่างผู้ใช้และโปรแกรม  
•มีการติดต่อกับ Database(ฐานข้อมูล) ด้วย Model และแสดงผลข้อมูลผ่านทาง View  
•เป็นส่วนที่มีการประมวลผลหลัก ของโปรแกรม

**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 Active Record?::Base class และโปรแกรมจะเข้าใจอัตโนมัติว่าจะใช้ตาราง RDBMS อันไหน และเรียกคอลัมภ์ต่างๆในตารางเองใน 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 เพื่อให้งานเว็บแอพลิเคชั่นนั้นมีมาตรฐานเดียวกันทั้งหมด ในกรณี template จะสร้างไฟล์ ที่เป็นตัวกำหนด header, content, footer ไว้ที่โฟลเดอร์ layout ภายใต้โฟลเดอร์ view ซึ่งลักษณะการทำงานของไฟล์นี้ จะถูกเรียกใช้ในการแสดงผลทุกครั้ง เป็นต้น

**8.**

Framework ที่เคยใช้

**Bootstrap**

สุดยอด Front-End Framework ที่จะช่วยในการแก้ปัญหาการแสดงผลเว็บไซต์บนอุปกรณ์มากมายหลากหลาย ไม่ว่าจะโทรศัพท์มือถือ แท๊บเล็ต พีซี หรืออุปกรณ์พกพาหลายขนาด ที่ทำให้ผู้ใช้และผู้พัฒนาปวดหัวกับการทำเว็บ Bootstrap จะช่วยแก้ปัญหาเหล่านั้น ด้วยการออกแบบที่พิถีพิถัน เว็บของคุณจะเป็นที่น่าจดจำตลอดไป

ข้อดี  
1. การแสดงผลหลากหลายขนาดหน้าจอ ซึ่ง Bootstrap ถือว่าความมามารถนี้เป็นพระเอกเลยก็ว่าได้ มีกลไกการแสดงผลสำหรับอุปกรณ์ที่หลากหลาย โดยเขียน code แค่ครั้งเดียวแต่รองรับได้หลายขนาดหน้าจอ ไม่ต้องมาทำ web หลายๆ version เหมือนในอดีต   
2. UI (user interface) ส่วนประกอบของหน้า web ที่สวยงาม Bootstrap ได้เตรียม ui ไว้มากมาย แถมมีนักพัฒนาทั่วโลก ทำ ui มาแจกฟรีอีกเพียบครับ เพียงเขียน code ไม่กี่ตัวอักษรก็สามารถแปลง web ธรรมดาให้สวยได้   
3. JavaScript ที่แสนง่ายดาย เมื่อก่อนจะทำไรทีที่เป็น interactive กับ user ที่เป็น JavaScript ต้องใช้การเขียน code ที่เยอะมาก ยุ่งพอสมควร แต่ Bootstrap ได้ช่วยให้เขียน code สั้น และง่ายกว่ามาก ทำให้ใช้เวลาไม่นานแถมประหยัดพื้นที่ source code อีกด้วย  
4. CSS Preprocessors กลไกใหม่ของ css หรือเรียกง่ายๆว่า css ที่มีการ compile ไว้แล้ว โดย Bootstrap ทำงานกับ 2 preprocessors ดัง คือ Less และ Sass

ข้อเสีย  
1. หน้าเว็บที่ใช้ Bootstrap เขียนอาจมีหน้าตาเหมือนๆกัน  
2. นักเขียนเว็บหลายๆคนต่อต้าน Bootstrap

**Ruby on Rails**

Ruby on Rails เป็นโอเพนซอร์สที่เติบโตและได้รับความนิยมอย่างก้าวกระโดดด้วยการพิสูจน์ตัวเองมาแล้วจากหลายบริษัทชั้นนำที่ได้นำไปใช้ อย่าง Twitter หรือ Shopify ด้วยความรวดเร็วในการพัฒนาเว็บแอพพลิเคชั่น และความคล่องตัวในการแก้ไขปรับเปลี่ยน พร้อมด้วยชุมชนผู้พัฒนาขนาดใหญ่ที่ได้ร่วมแบ่งปันสร้างเครื่องไม้เครื่องมือที่พร้อมใช้งานเอาไว้มากมายองค์ประกอบที่สุดยอดเหล่านี้ ช่วยให้เรามีเวลามากมายเพื่อช่วยโฟกัสในการแก้ปัญหาของลูกค้าให้ประสบความสำเร็จ

ข้อดี  
1. ไม่ต้องเสียเวลาพัฒนาเครื่องมือเพื่อใช้เอง จึงสามารถใช้เวลาไปกับการสร้างซอฟต์แวร์ตาม ข้อกำหนดได้มากขึ้น  
2. ส่งเสริมให้วิศวกรซอฟต์แวร์มีระเบียบวินัยที่ชัดเจน เป็นมาตรฐานเดียวกัน  
3. มีระบบการทดสอบด้วยซอฟต์แวร์ ทำให้การทดสอบครบถ้วนสม่ำเสมอ  
4. Ruby on Rails เป็นซอฟต์แวร์ open source ทำให้สามารถตรวจสอบซอร์สโค้ดได้ครบถ้วน ช่วยให้การแก้ปัญหาเป็นไปได้ง่ายขึ้น  
5. ข้อดีอีกข้อของ open source software คือการมีเครือข่ายวิศวกรซอฟต์แวร์ทั่วโลก ช่วยกันประดิษฐ์เครื่องมือเพื่อการใช้งานจำเพาะต่างๆ และแบ่งปันให้เพื่อนวิศวกรซอฟต์แวร์ ได้นำไปใช้และพัฒนาต่อยอดยิ่งๆ ขึ้นไป

ข้อเสีย  
1. การติดตั้ง Ruby on Rails บนเครื่อง development ยุ่งยากกว่าเครื่องมืออื่นๆ เช่น PHP, Java หรือ .NET  
2. การ Deploy Rails application ขึ้นเซิร์ฟเวอร์จริง ก็ยิ่งยุ่งยากมากกว่าการติดตั้งบนเครื่อง development  
3. ความเร็วในการทำงานของภาษา Ruby จัดได้ว่าช้ากว่าคู่แข่งอยู่มาก มีความพยายาม แก้ปัญหานี้จากหลายๆ ฝ่าย เช่น JRuby เป็นการใช้ Java Virtual Machine มารันโปรแกรมที่เขียนขึ้นด้วยภาษา Ruby อาศัยการทำงานของ Java Hotspot Compiler ในการช่วยเพิ่มความเร็วในการทำงาน  
4. ทั้ง Ruby และ Ruby on Rails มีทัศนคติที่ไม่ดีกับ Microsoft Windows การส่งเสริมการ ทำงานบน Windows จึงมีไม่มากนัก ทำให้การพัฒนา Ruby on Rails ต้องทำบน Linux หรือ Apple Mac OS เป็นหลักและ Deploy ขึ้น Linux Server เป็นหลัก  
5. Ruby on Rails ได้รับความนิยมมากในต่างประเทศ แต่ในประเทศไทยเพิ่งเริ่มมีความนิยมในระยะเวลา 1-2 ปีมานี้ ทำให้ตำรับตำราต่างๆ เป็นภาษาต่างประเทศทั้งหมด

**9.**

**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

**10.**

เหตุผลที่คณะวิทยาการสารสนเทศบรรจุวิชานี้ขึ้นมาก็เพื่อ ให้นิสิตได้มีความรู้เรื่องการพัฒนาซอฟต์แวร์ ได้รู้ถึงเรื่องการใช้ Git ซึ่งไม่เคยได้ใช้มาก่อน