Waterfall

* เป็นแนวทางการพัฒนาซอฟต์แวร์ที่มีความเป็นระบบสูงและเป็นลำดับชัดเจน โดยจะเริ่มจากความต้องการซอฟต์แวร์ของลูกค้า ต่อเนื่องไปถึงการวางแผน การออกแบบ การสร้าง และการนำไปใช้ซึ่งเหมาะกับสถานการณ์ที่มีการเพิ่มความสามารถให้กับระบบที่มีอยู่แล้ว หรือถ้าเป็นการสร้างฟังก์ชั่นการทำงานใหม่ก็จะเป็นการพัฒนาเพิ่มแบบไม่มากนัก มีความต้องการที่ชัดเจนและไม่เปลี่ยนแปลง

Agile

* สำหรับแนวทางการพัฒนาแบบอาไจล์จะเน้นการวิเคราะห์และออกแบบให้พอเหมาะเพื่อให้ซอฟต์แวร์สามารถส่งมอบได้ทันเวลาและมีการสื่อสารกันอย่างต่อเนื่องระหว่างลูกค้าและนักพัฒนา

Centralized

* Centralized Version Control Systems (CVCSs) หรือระบบ Version Control Systems แบบรวมศูนย์
* ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง
* คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้
* จุดอ่อน : ตรงที่การรวมศูนย์ทำให้มันเป็นจุดอ่อนจุดเดียวที่จะล่มได้เหมือนกันเพราะทุกอย่างรวมกันอยู่ที่เซิร์ฟเวอร์ที่เดียว ถ้าเซิร์ฟเวอร์นั้นล่มซักชั่วโมงหนึ่ง หมายความว่าในชั่วโมงนั้นไม่มีใครสามารถทำงานร่วมกันหรือบันทึกการเปลี่ยนแปลงงานที่กำลังทำอยู่ไปที่เซิร์ฟเวอร์ได้เลย

Distributed

* Distributed Version Control Systems (DVCSs) หรือระบบ VCS แบบกระจายศูนย์
* แต่ละคนไม่เพียงได้ก๊อปปี้ล่าสุดของไฟล์เท่านั้น แต่ได้ทั้งก๊อปปี้ของ repository
* หมายความว่าถึงแม้ว่าเซิร์ฟเวอร์จะเสีย client ก็ยังสามารถทำงานร่วมกันได้ต่อไป และ repository เหล่านี้ของ client ยังสามารถถูกก๊อปปี้กลับไปที่เซิร์ฟเวอร์เพื่อกูข้อมูลกลับคืนก็ได้
* ระบบนี้ยังทำงานกับหลาย ๆ repository ได้อย่างดี ทำให้สามารถทำงานกับคนหลายกลุ่มซึ่งทำงานในรูปแบบต่างกันในโปรเจคเดียวกันได้อย่างง่ายดาย

SaaS application มีจุดเด่นอย่างไร

* ย่อมาจาก “Software as a Service” หรือเรียกว่า “On Demand Software” คือรูปแบบการให้บริการซอฟต์แวร์ผ่านทางอินเทอร์เน็ต
* สามารถเข้าใช้ผ่านเว็บ Browser ผ่านอุปกรณ์ เช่น เครื่องคอมพิวเตอร์พีซี เครื่องคอมพิวเตอร์พกพา (Laptop) หรือโทรศัพท์มือถือ โดยไม่ต้องติดตั้งซอฟต์แวร์ที่เครื่องของผู้ใช้
* ระบบหลักของ SaaS จะถูกควบคุมจากผู้ให้บริการ SaaS เอง โดยผู้ใช้บริการเพียงแค่ access เข้ามาด้วย Username และ Password เพื่อระบุตัวตนเท่านั้น

ประโยชน์ของการมี framework

* เป็น MVC Framework
* มีระบบ template
* ลดการเขียน code ที่ซ้ำไปซ้ำมา เพราะว่ามีการเขียนเป็น class , function
* มีการวางโครงสร้างไฟล์ ไว้อย่างเป็นระบบ ระเบียบที่แน่นอน

Rails framework

* Rails เป็น Web Framework มีลักษณะเป็น MVC (Model-View-Controller)
* Rails ถูกออกแบบมาให้มีการใช้งานที่ง่ายและรวดเร็ว ลดปัญหางานที่ต้องทำซ้ำๆ ทำให้ได้ productivityที่สูงขึ้น สอดคล้องกับ Methodology แบบ Agile

Model-View-Controller (MVC)

* 1. Model คือ ส่วนที่ใช้ในการติดต่อกับฐานข้อมูล ซึ่งทำหน้าที่ในการดึงข้อมูลขึ้นมาจาก

ฐานข้อมูลขึ้นมา และทำการจัดการข้อมูลต่างๆไว้ในรูปแบบที่เหมาะสม

* 2. View คือส่วนที่จะนำข้อมูลจาก Model ไปใช้แสดงผลให้ผู้ใช้เห็นผลลัพธ์ออกมาใน User Interface
* 3. Controller คือส่วนที่จะคอยรับ Input จาก Client เข้ามาแล้วนำคำสั่งไประมวลผล

เพื่อสั่งงาน View และ Model ให้ประมวลผลออกมาเป็นอย่างไร

REST: Representational State Transfer

* การใช้งาน resource ต่างๆ
* ผ่าน HTTP protocol
* ผ่าน URI ที่สื่อสารได้ชัดเจน
* ผ่าน HTTP method ที่เหมาะสม เช่น

HTTP GET สำหรับการดึงข้อมูล

HTTP POST สำหรับการสร้างข้อมูล

HTTP PUT สำหรับการแก้ไขข้อมูล

HTTP DELETE สำหรับลบข้อมูล

* รูปแบบของข้อมูลที่ใช้สื่อสารกันอยู่ในรูปแบบ XML, JSON, Atom และอื่นๆ อีกมากมาย แต่หนึ่งในรูปแบบที่ได้รับความนิยมคือ JSON

การ branch

* แก้ไขอะไรก็ตาม ไม่ให้กระทบกับตัวงานหลัก ก็เพียงแค่สร้าง branch ใหม่ขึ้นมา เมื่อแก้ไขหรือทำอะไรเสร็จแล้ว ก็ค่อยเซฟกลับมาที่ master เหมือนเดิม

การ merge

* การรวมไฟล์แต่ละไฟล์มาอยู่ที่เดียวกัน

การแก้ไข conflict

* เปิดไฟล์ที่มีปัญหาขึ้นมา ลบโค๊ดส่วนเกินออก แล้วแก้ไขใหม่ให้เรียบร้อย จากนั้นลองเช็คสถานะเมื่อมั่นใจว่าโปรแกรมทำงานถูกต้องแล้ว ก็สั่ง commit และ push เป็นอันเสร็จเรียบร้อย