(11.)

1. ช่วยให้สามารถย้อนไฟล์บางไฟล์หรือแม้กระทั่งทั้งโปรเจคกลับไปเป็นเวอร์ชั่นเก่าได้

2. ช่วยให้เปรียบเทียบการแก้ไขที่เกิดขึ้นในอดีต ดูว่าใครเป็นคนแก้ไขคนสุดท้ายที่อาจทำให้เกิดปัญหา แก้ไขเมื่อไร

3. ช่วยให้สามารถกู้คืนไฟล์ที่คุณลบหรือทำเสียโดยไม่ตั้งใจได้อย่างง่ายดาย

4.ช่วยแก้ปัญหาเผลอเขียนทับไฟล์ที่ไม่น่าจะเขียนทับ หรือทำการคัดลอกไฟล์ที่ไม่น่าจะคัดลอก

5. เพื่อเก็บการแก้ไขทั้งหมดที่เกิดขึ้นกับไฟล์ที่อยู่ภายใต้ revision control

---

(12.)

ข้อได้เปรียบของ Distributed Version Control มีดังนี้

1.ระบบที่ทันสมัยยังมีการบีบอัดไฟล์ที่จะใช้พื้นที่แม้แต่น้อย

2.ไม่จำเป็นต้องพึ่งพาเซิร์ฟเวอร์กลางในการจัดเก็บข้อมูล

3.การเคลื่อนย้ายการเปลี่ยนแปลงของเองเพื่อพื้นที่เก็บข้อมูล

ส่วน Centralized Version Control

1.การแก้ไข ซึ่งก็เป็นกลุ่มของการเปลี่ยนแปลงอาจจะหลายไฟล์

(13.)

ข้อได้เปรียบของ Centralized Version Control มีดังนี้

1.การบันทึกการเปลี่ยนแปลงในระบบกลาง เขียนโปรแกรมอื่น ๆ นั้นจะสามารถเห็นการเปลี่ยนแปลงนี้

2.เครื่องมือการควบคุมเวอร์ชันโดยอัตโนมัติจะปรับปรุงเนื้อหาของไฟล์ใด ๆ

3.เช่น การเปลี่ยนแปลงไฟล์ส่วนหัว C และแฟ้ม .c สอดคล้องกันเสมอควรจะเก็บไว้ด้วยกัน

ส่วน Distributed Version Control

1.การเปลี่ยนแปลงในกลุ่มไฟล์ wholes เชื่อมโยงกัน ไม่ใช่ diffs ไฟล์เดียว

(14.)

สำหรับปัญหาแรก นักพัฒนาจะถูกบังคับให้แก้ไข โดยการ Merge Source Code ด้วยมือ เรียกอย่างเป็นทางการ ก็คือ Conflict Resolution หรือการแก้ Conflict

ส่วนปัญหาที่สอง มีทางเดียวคือต้องใช้ CI

(15.)

ผู้พัฒนา อาจจะทำงานในบริเวณเดียวกันของไฟล์ แต่ข้อดีของ SVN ก็คือ Source Code ที่อยู่ใน Repository จะไม่มีทางอยู่ในสภาพที่เสียหายโดยเด็ดขาด เพราะว่า ก่อนการ Commit Code เข้าไปยัง Repository ตัว SVN จะตรวจว่า มีการเปลี่ยนแปลงที่ไฟล์เดียวกันหรือไม่ ถ้ามี ผู้พัฒนา จะต้องทำการ Update เพื่อดาวน์โหลด Source Code มาทำการ “ทดลอง Merge” ในเครื่องก่อน ถ้าเกิดว่า Merge ไม่ผ่าน ก็จะไม่สามารถ Commit ได้

(16.)

GitHub คือ เว็บที่ให้บริการพื้นที่จัดเก็บโครงการโอเพ่นซอร์สด้วยระบบควบคุมเวอร์ชันแบบ Git โดยมีจุดประสงค์หลักคือ ทำให้การแบ่งปันและพัฒนาโครงการต่างๆด้วยกันเป็นไปได้ง่ายๆ

(17.)

เรามักจะแตก branch ออกไปให้แต่ละกลุ่มพัฒนากันใน branch การแตก branch นั้นมันจะทำให้โค้ดของแต่ละกลุ่มไม่แตะต้องกันเลย คือ branch หนึ่งจะไม่เห็นการเปลี่ยนแปลงโค้ดของอีก branch หนึ่ง มันแยกอิสระกันเลยทีเดียว ทำให้การพัฒนาโค้ดง่ายมาก ถ้ากลุ่มไหนพัฒนาเสร็จก่อน ก็ไม่จำเป็นจะต้องรอให้อีกกลุ่มเสร็จ branch ที่แตกออกไปก็สามารถเทสโค้ดได้อย่างอิสระ

(18.)

หลังจากที่ผสาน ยังไม่ได้แยกออกแทนการสร้างใหม่กระทำ Git ก็จะชี้ต้นแบบล่าสุดกระทำ

เห็นคำว่า Fast forward เวลาสั่ง git pull มานานแล้ว เวลาเราสั่ง git merge การทำงานของมันสามารถแบ่งได้เป็น 3 ประเภท

1.ถ้า merged commit ที่เราดึงมา อยู่ใน Head(current tree ของเรา) ของเราแล้ว, ก็จะแสดงผลลัพท์ "Already up-to-date." แล้วก็จบการทำงาน

2.ถ้า Head ของเราอยู่ใน commits ที่ดึงมา case นี้มักเกิดจากคำสั่ง "git pull" เพื่อดึง code จากต้นน้ำมา update code(ที่ไม่มีการเปลี่ยนแปลง) ของเรา, สิ่งที่เกิดขึ้นก็คือ git จะ update HEAD ของเราให้ตรงตาม HEAD ของ merged commit (โดยไม่มีการสร้าง commit object ใหม่ขึ้นมา) มีศัพท์เฉพาะสำหรับกรณีนี้ว่า "Fast-forward"

3.สุดท้ายเป็นกรณีที่เกิดการ merge จริงๆ นั่นคือ ตัว HEAD ของเรา independent กับ merged commit ดังนั้น กรณีนี้จะเกิดการ merge จริง และมีการสร้าง commmit object ใหม่ขึ้นมา

(19.)

ใช้ดึงความเปลี่ยนแปลงจาก remote มายัง local และรวมเข้าด้วยกัน (มีความเข้ากับ fetch + merge)

(20.)

คือการ Merge ไปข้างหน้าเป็นขั้นๆ ในระดับ V0.1 V0.2 ไปเรื่อยๆ จนถึง V1.0