11.เป็นที่จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนึ่งหรือหลายไฟล์เพื่อที่ สามารถเรียกเวอร์ชั่นใดเวอร์ชั่นหนึ่งกลับมาดูเมื่อไรก็ได้

12.Distributed Version Control เป็นระบบ VCS แบบกระจายศูนย์ ทั้ง SCCS, RCS, CVS, SVN ทั้งหมดนั้นใช้ 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

นอกจากนี้เรายังสามารถ fork projects ได้ด้วย แปลง่ายๆ คือดึงโปรเจกต์มาอยู่ในชื่อเราเพื่อแก้ไขในแบบที่เราต้องการหรือใช้สำหรับแก้โค้ดและส่งกลับไปให้กับเจ้าของที่พัฒนาโค้ดก็ได้ การ fork ในที่นี้คือการยอมให้ outside developers นำโค้ดของผู้พัฒนาไปแก้และ pull request กลับไปให้ผู้พัฒนานั่นเอง

13.Centralized Version Control เป็นระบบ VCS แบบรวมศูนย์ ระบบเหล่านี้มีเซิร์ฟเวอร์กลางที่เก็บไฟล์ทั้งหมดไว้ในที่เดียวและผู้ใช้หลาย ๆ คนสามารถต่อเข้ามาเพื่อดึงไฟล์จากศูนย์กลางนี้ไปแก้ไขได้ การทำงานแบบนี้มีประโยชน์ในหลายด้าน เช่น ทุกคนสามารถรู้ได้ว่าคนอื่นในโปรเจคกำลังทำอะไร ผู้ควบคุมระบบสามารถควบคุมได้อย่างละเอียดว่าใครสามารถแก้ไขอะไรได้บ้าง การจัดการแบบรวมศูนย์ในที่เดียวทำได้ง่ายกว่าการจัดการฐานข้อมูลใน client แต่ละเครื่องเยอะ

14.มี 4 วิธีตามลำดับดังนี้

1. ทำการ Merge บ่อย ๆ

2.Single Responsibility Principle (SRP)เป็นหนึ่งในแนวทางการออกแบบระบบงานที่ดี

3. การพูดคุย การสื่อสาร มันสำคัญอย่างมาก

4.การ Mob programming

15.ทำการ merge บ่อยๆ นั่นคือ ทุกครั้งเมื่อคุณทำการเปลี่ยนแปลง หรือ commit source code นั่นเองจะช่วยลดข้อขัดแย้งต่าง ๆ ลงไปอย่างมากถึงจะเกิดข้อขัดแย้ง ก็เป็นเพียงปัญหาเล็ก ๆซึ่งสามารถแก้ไขได้อย่างง่ายดาย

การออกแบบที่ดีมันช่วยให้ทีมทำงานร่วมกันได้อย่างดียิ่งแต่ละส่วนการทำงานเล็กๆ แล้วก็ยิ่งทำให้คุณภาพของการออกแบบระบบดีรวมทั้ง ลดความเสี่ยงจาก Merge conflict อีกด้วย

Mob programming เป็นวิธีการที่ทรงประสิทธิภาพอย่างมากและเชื่อได้เลยว่า แก้ไขปัญหา Merge conflict ได้ 100%เนื่องจากทุกคนมานั่งทำงานด้วยกันใช้เครื่องทำงานเดียวกันดังนั้น ไม่มีทางที่ source code จะขัดแย้งกันแต่มันมีค่าใช้จ่ายที่สูงมาก ๆ

16.Git เป็น revision control แบบ distributedและ แบบ non-linear history ดังนั้นคอนเซปต์ของ Git นั้นต่างจาก revision control รุ่นก่อนหลายอย่าง

GitHub คือ เว็บที่ให้บริการพื้นที่จัดเก็บโครงการโอเพ่นซอร์สด้วยระบบควบคุมเวอร์ชันแบบ Git โดยมีจุดประสงค์หลักคือ ทำให้การแบ่งปันและพัฒนาโครงการต่างๆด้วยกันเป็นไปได้ง่ายๆ

17.การแยกตัวออกมาจาก main line ของการพัฒนาและทำงานต่อไปบนบนนั้นโดยไม่ไปยุ่งเกี่ยวกับ main line ในหลายๆ VCS การทำแบบนี้ค่อนข้างจะเปลือง ส่วนใหญ่จะเป็นการ copy ทั้ง directory ของ source code ซึ่งจะกินเวลานานมากบน project ใหญ่ๆ

18.Fast forward เกิดขึ้นจากเวลาเราใช้คำสั่ง merge แล้ว git พบว่ามีประวัติมาจากที่เดียวกัน มันจะพยายามทำเป็นเส้นตรงแนวเดียวกัน

19.คือการรวมโค๊ดจาก remote มายัง local โดยที่เราไม่สามารถรู้ได้เลยว่าจะรวมโค๊ดอะไรบ้าง รู้แค่หลังจาก pull เสร็จแล้ว

20.มันคือ gitflow จากภาพ เราจะเห็นวิธีการตั้งเวอร์ชั่นของโปรแกรมที่เราเขียนด้วย ถ้าหากเป็นการเอาโค้ดจาก Release ขึ้นโปรดักชั่น เขาจะใช้หมายเลขเวอร์ชั่นใหญ่ เช่น 1.0, 2.0, 3.0 แต่ถ้าเป็นการ Hotfix เขาจะใช้เลขเวอร์ชั่นย่อย เช่น 1.1, 1.2, 2.1, 3.1 เป็นต้น