**El problema de compartir ficheros**

Considere este escenario: suponga que tiene dos compañeros de trabajo, Harry y Sally. Cada uno decide editar el mismo fichero del repositorio a la vez. Si Harry graba sus cambios en el repositorio primero, el posible que (unos momentos después) Sally pueda accidentalmente sobreescribirlos con su propia versión nueva del fichero. Mientras que la versión del fichero de Harry no se ha perdido para siempre (porque el sistema recuerda cada cambio), cualquier cambio que Harry hizo *no estará* en la versión nueva del fichero de Sally, porque para empezar ella nunca vió los cambios de Harry. El trabajo de Harry está aún efectivamente perdido - o al menos falta en la última versión del fichero - y probablemente por accidente.

¡Esta es una situación que definitivamente tenemos que evitar!

****

**2.2.2. La solución bloquear-modificar-desbloquear**

Muchos sistemas de control de versiones utilizan un modelo *bloquear-modificar-desbloquear* para enfrentarse a este problema, que es una solución muy simple. En estos sistemas, el repositorio sólo permite que una persona cambie un fichero al mismo tiempo. Harry primero debe *bloquear* el fichero antes de que pueda empezar a hacer cambios en él. Bloquear un fichero se parece mucho a tomar prestado un libro de la biblioteca; si Harry ha bloqueado un fichero, entonces Sally no puede hacer ningún cambio en él.

Si ella intenta bloquear el fichero, el repositorio le denegará la petición. Todo lo que ella puede hacer es leer el fichero, y esperar a que Harry termine sus cambios y libere su bloqueo. Después de que Harry desbloquee el fichero, se acabó su turno, y ahora Sally puede bloquear y editar.



**Figura 2.3. La solución bloquear-modificar-desbloquear**

El problema con el modelo bloquear-modificar-desbloquear es que es un poco restrictivo, y a menudo se convierte en una calle cortada para los usuarios:

• *El bloqueo causa muchos problemas administrativos.* A veces Harry bloqueará un fichero y luego se olvidará de ello. Mientras tanto, dado que Sally está aún esperando para editar el fichero, sus manos están atadas. Y Harry se va de vacacioens. Ahora Sally tiene que buscar a un administrador para que libere el bloqueo de Harry. La situación acaba causando un montón de retraso y pérdida de tiempo innecesarios.

• *El bloqueo puede causar procesos en serie innecesarios.* ¿Qué ocurre si Harry está editando el inicio de un fichero de texto, y Sally simplemente quiere cambiar la parte final del mismo fichero? Esos cambios no se superponen en absoluto. Ellos podrían fácilmente editar el fichero de forma simultánea, y no habría ningún daño, asumiendo que los cambios se fusionaran correctamente. No hay necesidad de que se turnen en esta situación.

• *El bloqueo puede causar una falsa sensación de seguridad.* Imagine que Harry bloquea y edita el fichero A, mientras Sally simultáneamente bloquea y edita el fichero B. Pero suponga que A y B dependen uno del otro, y que los cambios hechos a cada uno son semánticamente incompatibles. De repente A y B ya no funcionan juntos. El sistema de bloqueo no tiene forma de prevenir este problema - sin embargo, de alguna forma dió una sensación de falsa seguridad.

Es fácil para Harry y Sally imaginar que al bloquear los ficheros, cada uno está empezando una tarea segura y aislada, y por tanto les inhibe de discutir sus cambios incompatibles en un momento temprano.

**2.2.3. La solución copiar-modificar-fusionar**

Mercurial y otros sistemas de control de versiones (Subversion, CVS) utilizan un modelo *copiar-modificar-fusionar* como alternativa al bloqueo. En este modelo, el cliente de cada usuario lee el repositorio y crea una *copia de trabajo* personal del fichero o del proyecto. Luego, los usuarios trabajan en paralelo, modificando sus copias privadas. Finalmente, las copias privadas se fusionan juntas en una nueva versión final. El sistema de control de versiones a menudo ofrece ayuda en la fusión, pero al final la persona es la responsable de hacer que ocurra correctamente.

Aquí hay un ejemplo. Digamos que tanto Harry como Sally crean copias de trabajo del mismo proyecto, copiado del repositorio. Ellos trabajan concurrentemente, y hacen los cambios al mismo fichero A dentro de sus copias. Sally es la primera en grabar sus cambios en el repositorio. Cuando Harry intenta grabar sus cambios más tarde, el repositorio le informa que su fichero A está *desactualizado*. En otras palabras, que el fichero A en el repositorio ha cambiado de alguna forma desde la última vez que lo copió. Por lo que Harry le pide a su cliente que *fusione* cualquier nuevo cambio del repositorio dentro de su copia de trabajo del fichero A. Lo más seguro es que los cambios de Sally no se superpongan a los suyos; por lo que una vez que ambos conjuntos de cambios se han integrado, él graba su copia de trabajo de nuevo en el repositorio.

****

**Figura 2.4. La solución copiar-modificar-fusionar**



**Figura 2.5. ...Copiar-modificar-fusionar continuado**

¿Pero qué ocurre si los cambios de Sally *sí* se superponen a los cambios de Harry? ¿Qué hacemos entonces? La situación se denomina un *conflicto*, y normalmente no es mucho problema. Cuando Harry le pide a su cliente que fusione los últimos cambios del repositorio en su copia de trabajo, su copia del fichero A se marca de alguna forma como que está en un estado de conflicto: él será capaz de ver ambos conjuntos de cambios conflictivos, y manualmente podrá elegir entre ellos. Tenga en cuenta que el software no puede resolver conflictos automáticamente; sólo los humanos son capaces de entender y hacer las elecciones necesarias de forma inteligente. Una vez que Harry haya resuelto manualmente los cambios que se superponían (¡quizás discutiendo el conflicto con Sally!), puede volcar de forma segura el fichero fusionado al repositorio.

El modelo copiar-modificar-fusionar puede sonar un poco caótico, pero en la práctica, funciona extremadamente bien. Los usuarios pueden trabajar en paralelo, sin que tengan que esperar nunca uno por otro. Cuando trabajan en los mismos ficheros, resulta que la mayoría de los cambios concurrentes no se superponen en absoluto; los conflictos no son frecuentes. Y el tiempo que lleva resolver conflictos es mucho menor que el tiempo perdido por un sistema bloqueante.

Al final, todo se reduce a un factor crítico: la comunicación entre usuarios. Cuando los usuarios se comunican de forma pobre, aumentan los conflictos sintácticos y semánticos. No hay sistema capaz de forzar a los usuarios a comunicarse perfectamente, y no hay sistema que pueda detectar conflictos semánticos. Por lo que no hay motivo para que se le prometa falsamente que un sistema con bloqueos prevendrá de alguna forma los conflictos; en la práctica, el bloqueo parece inhibir la productividad más que otra cosa.

Hay una situación común donde el modelo bloquear-modificar-desbloquear resulta mejor, y es cuando tiene ficheros no-fusionables. Por ejemplo si su repositorio contiene algunas imágenes gráficas, y dos personas cambian la imagen a la vez, no hay forma de fusionar esos cambios. O Harry o Sally perderán sus cambios.