

我愛 Git

Jim Huang(黃敬群) “jserv”

website: <http://jserv.sayya.org/>

blog: <http://blog.linux.org.tw/jserv/>

TOSSUG - 5/20, 2008



Agenda

- 版本控制的典範移轉 (paradigm shift)
- 分散式版本控制系統
- **Git** 核心概念與實務



About

- 熱血工讀生
- 參與 GNU Classpath 在內的世界級自由軟體專案
- 熱愛 patch 軟體與衣服



版本控制系统 的典范转移

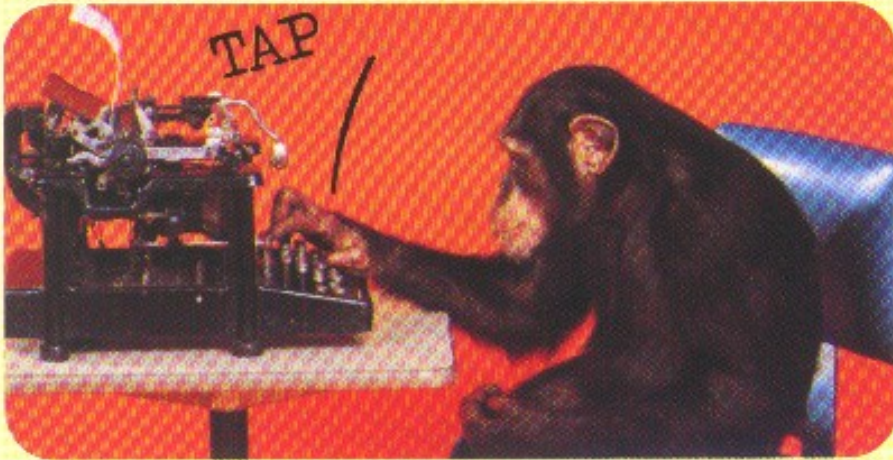


Changing...

- 時代在變
- 科技在變
- 電腦資訊快速改變
- 軟硬體的尺度劇增
- ...



但程式設計的模式基本上一致



來源：<http://www.chaos.org.uk/~eddy/when/2006/monkey.html>

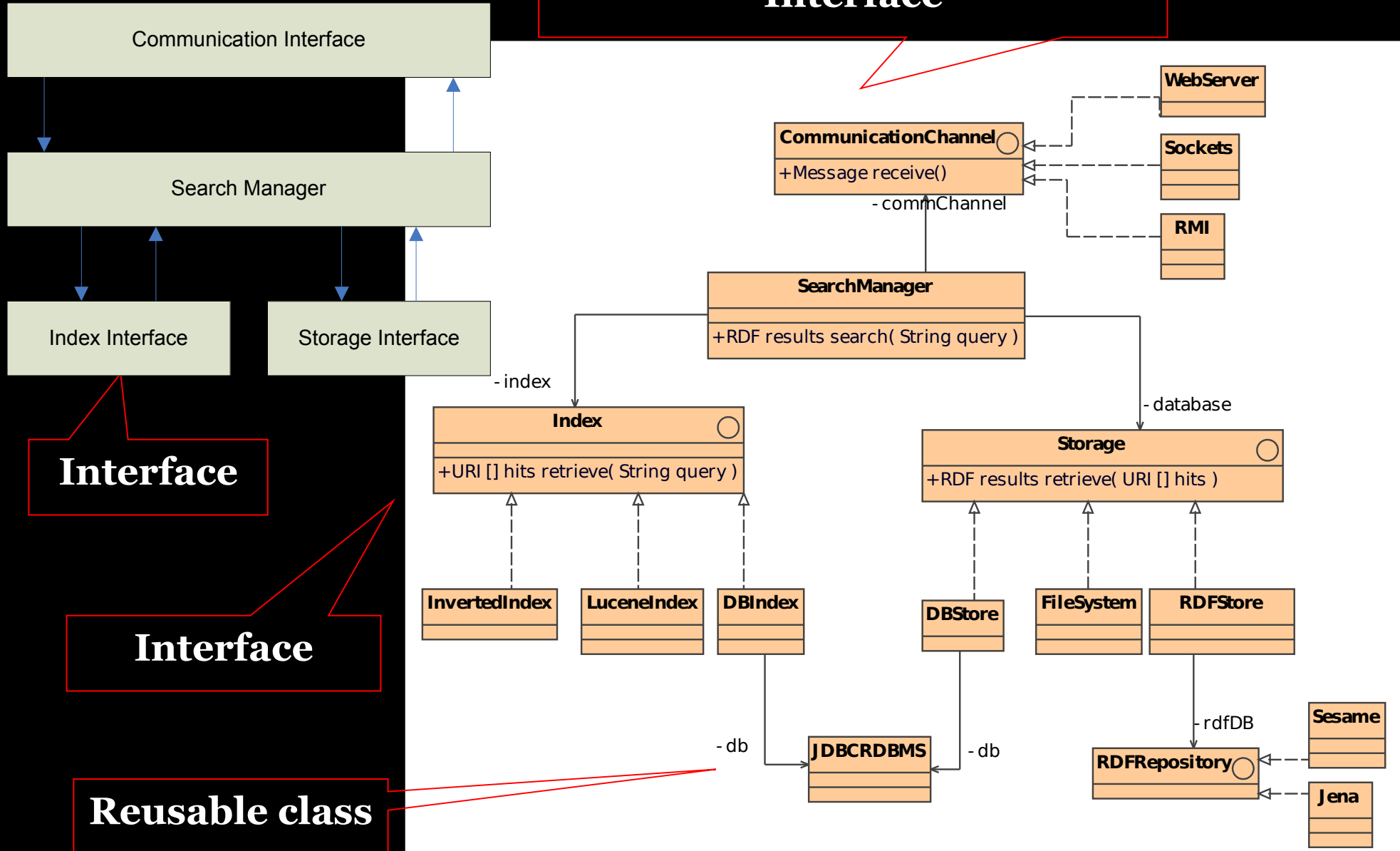


專案管理是大問題 ...



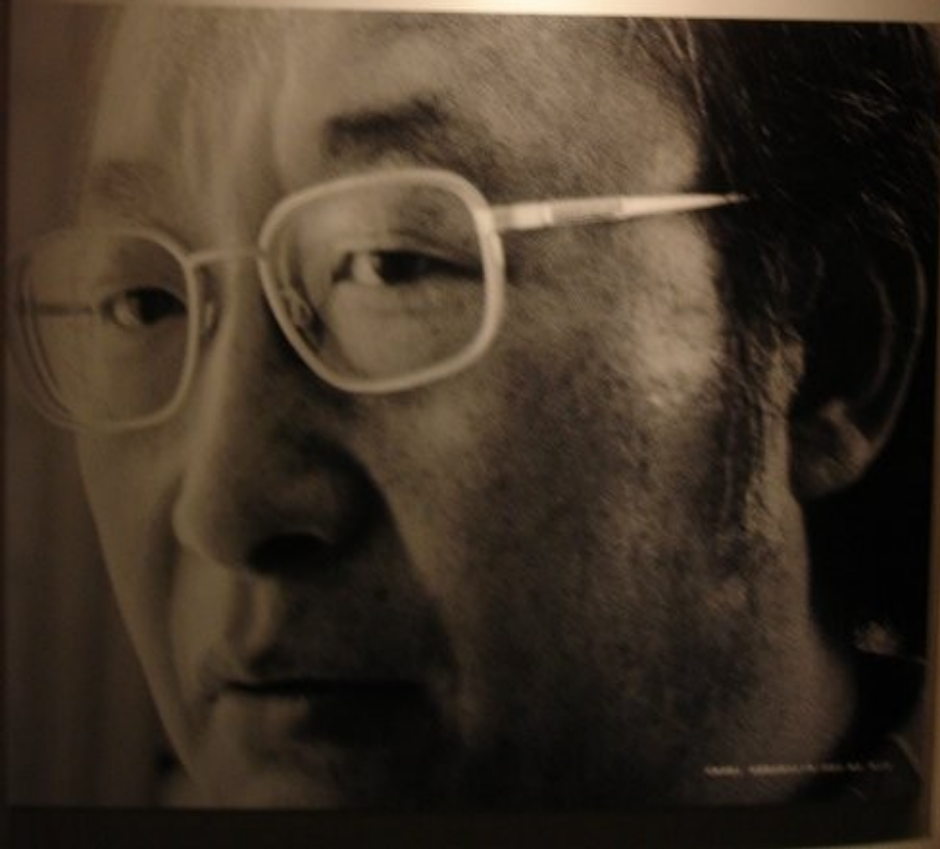
專案需要適度分析與設計

Interface



永懷歷史！

昨天的垃圾是
未來的考古遺物！



- 語出何傳坤博士
- 過程遠比結果重要
- 地理學家：「湖泊是天使的眼淚」
- 如何抱持「考古」的精神去「挖掘」？



Source Control Saves Lives



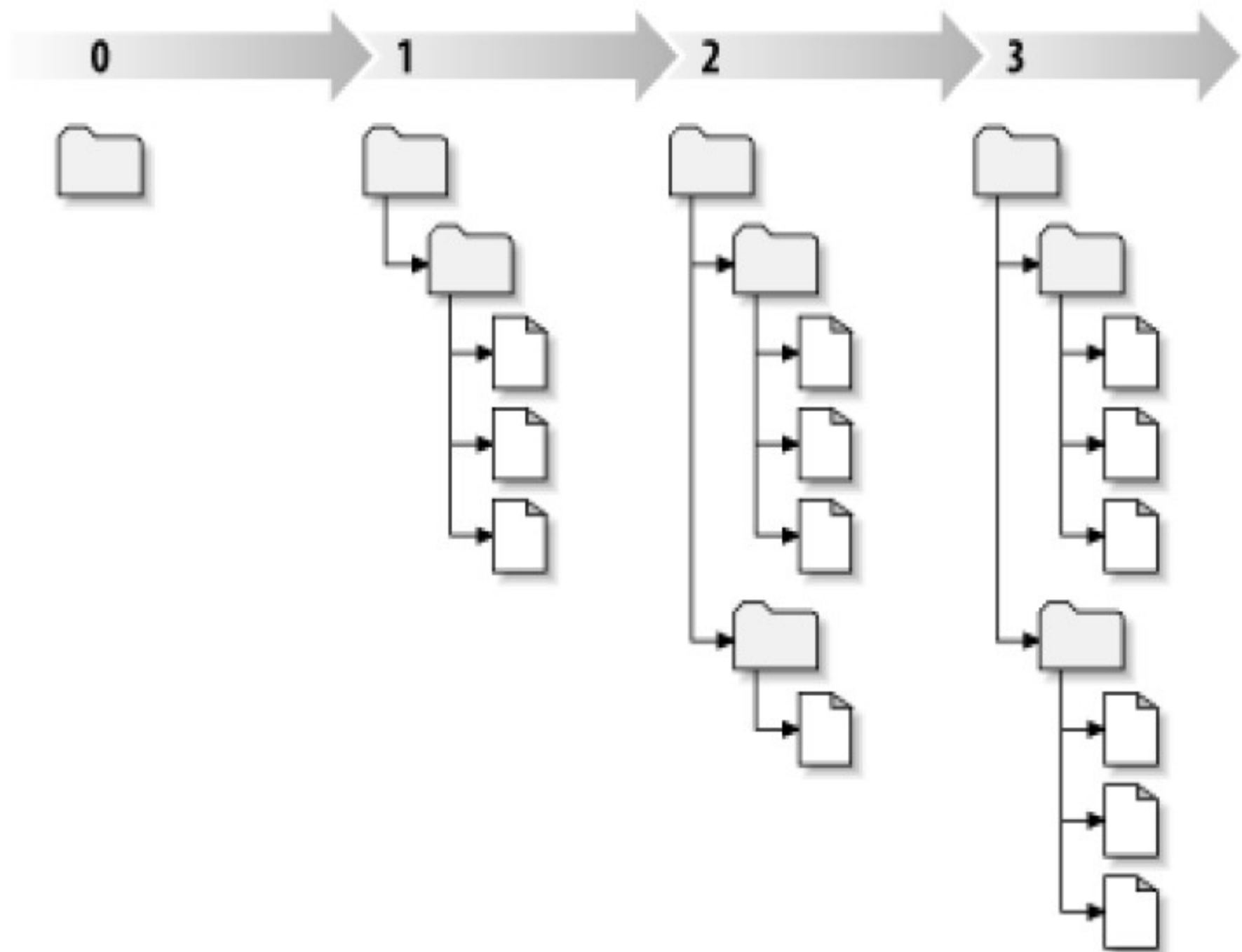


Figure 2.7. The Repository



Version Control / 版本控制

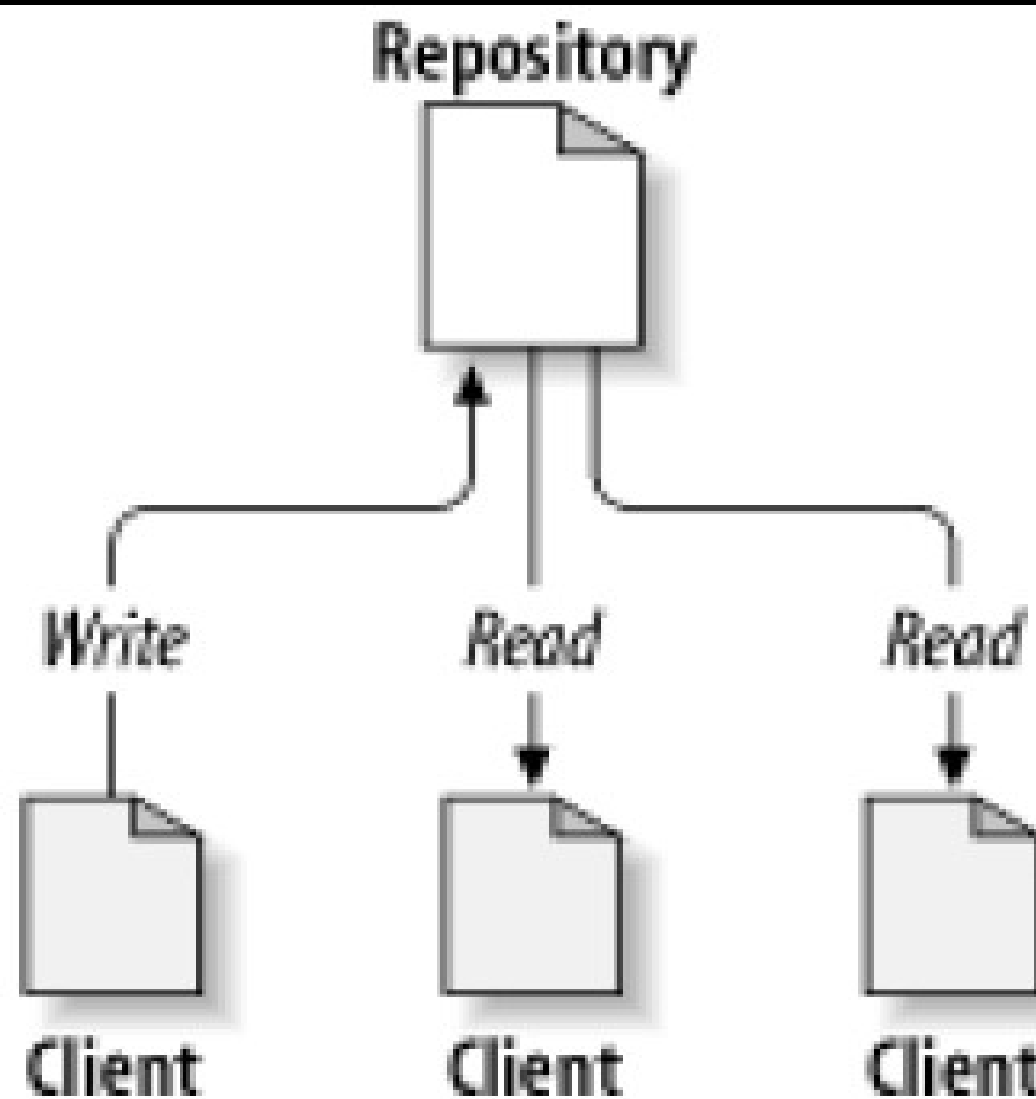


Figure 2.1. A Typical Client/Server System



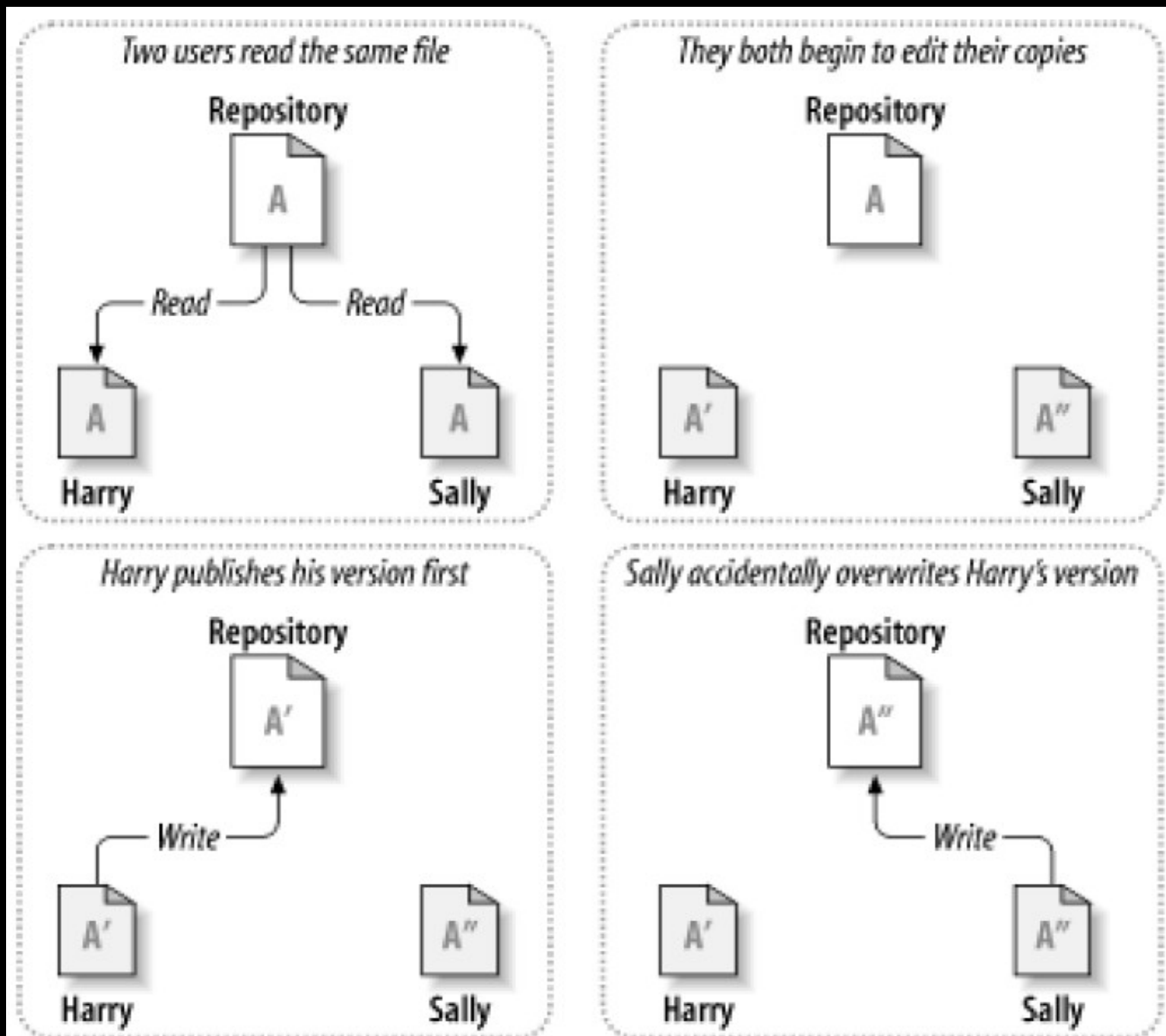


Figure 2.2. The Problem to Avoid



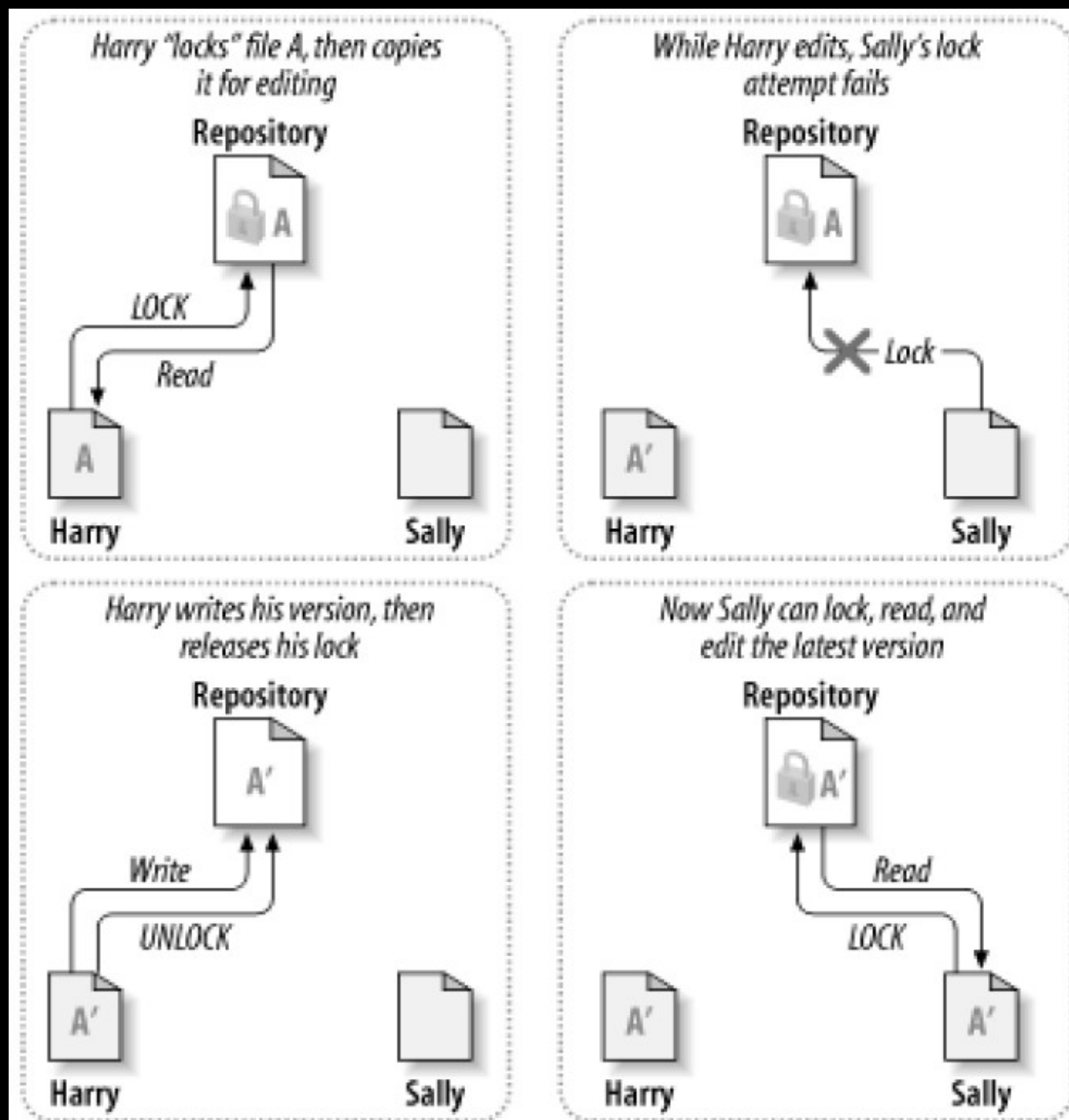


Figure 2.3. The Lock-Modify-Unlock Solution



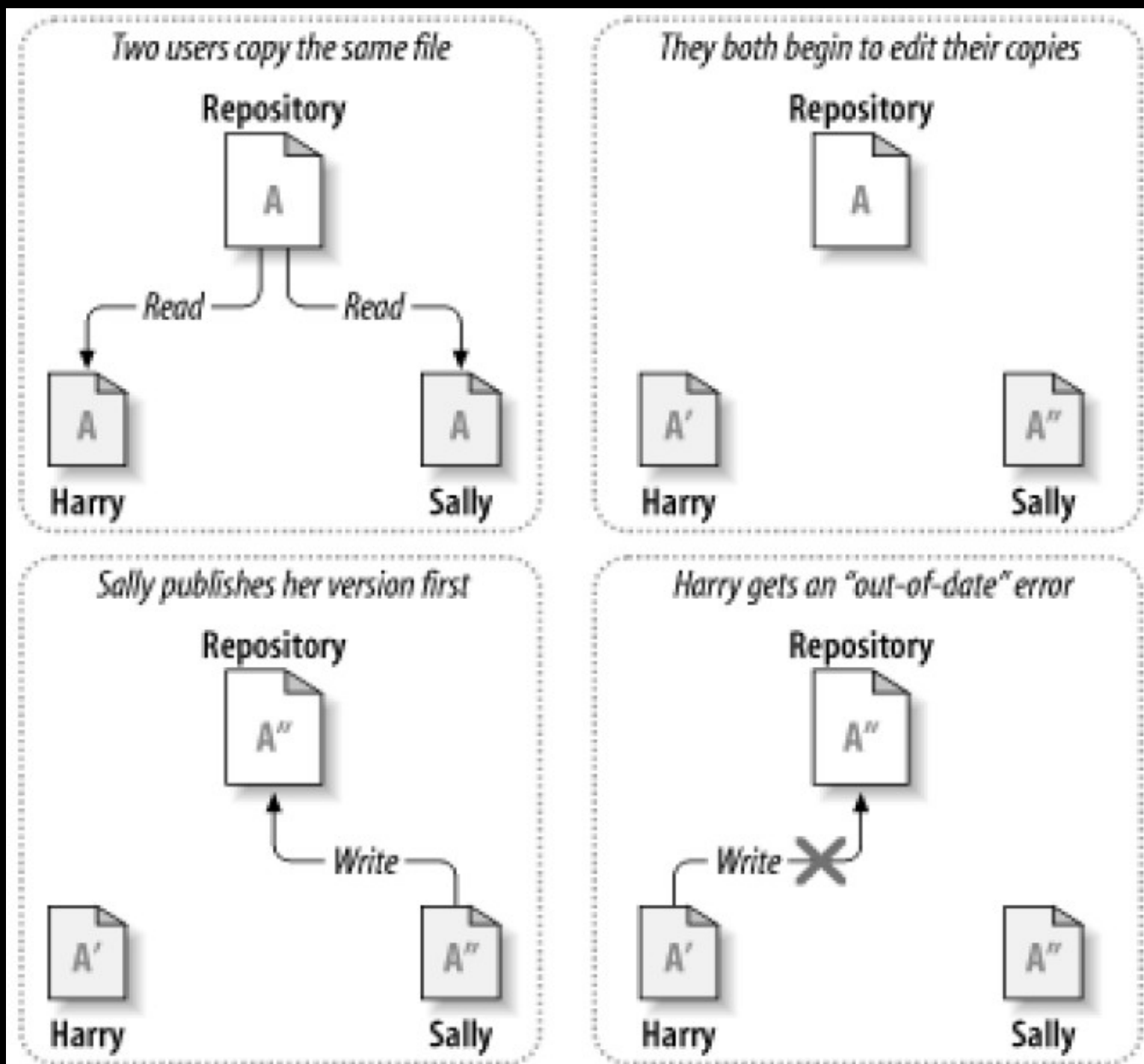


Figure 2.4. The Copy-Modify-Merge Solution



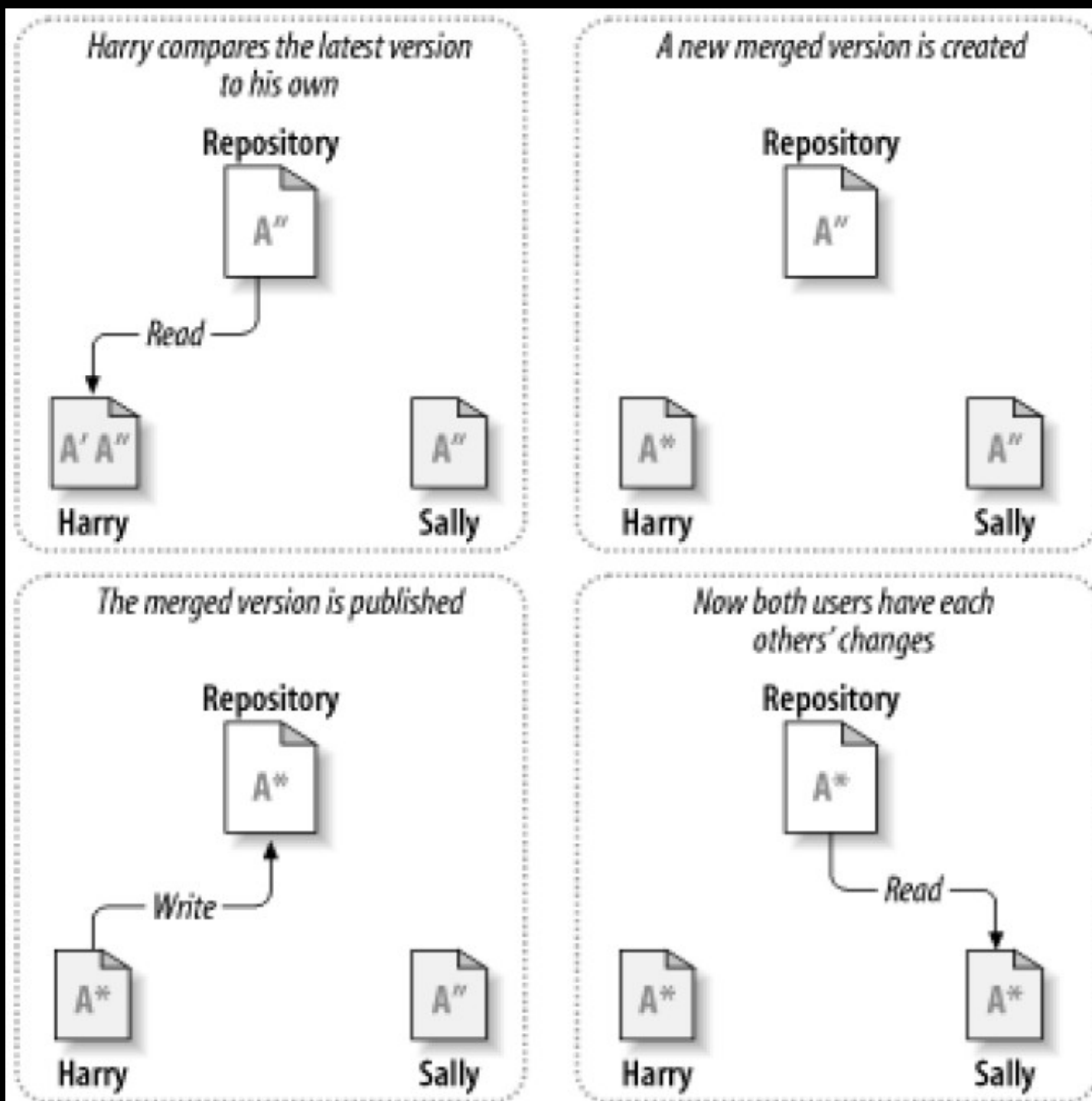


Figure 2.5. ...Copy-Modify-Merge Continued



我們的發現 ...

- 版本控制必須確保每次提交更改的完整性與一致性
 - lock-modify-unlock
 - copy-modify-merge
- 整合來自不同時間點的修改 (merge) 是相當重要的設計
- 時序 (timeline) 與集中式版本控制息息相關



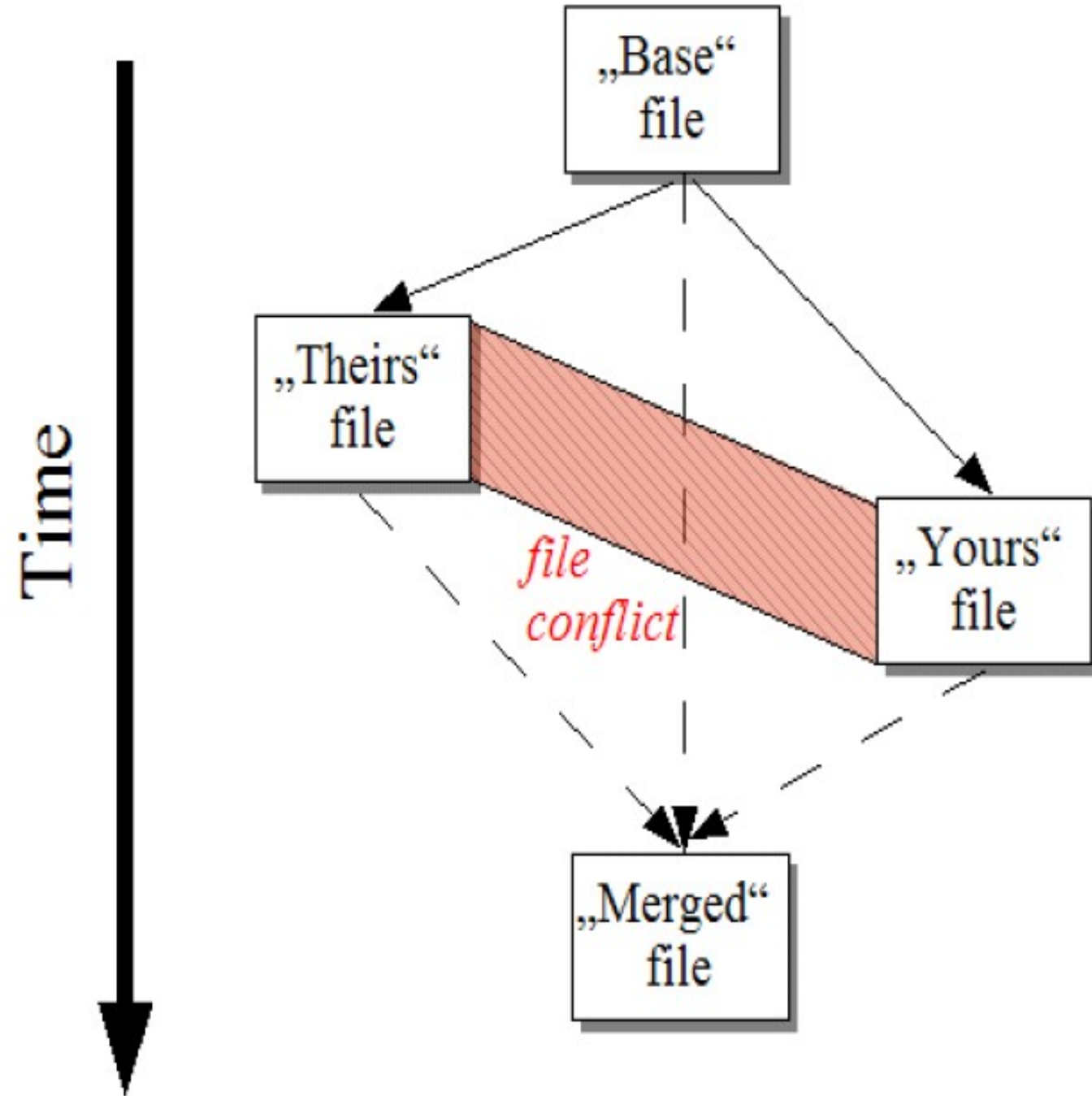
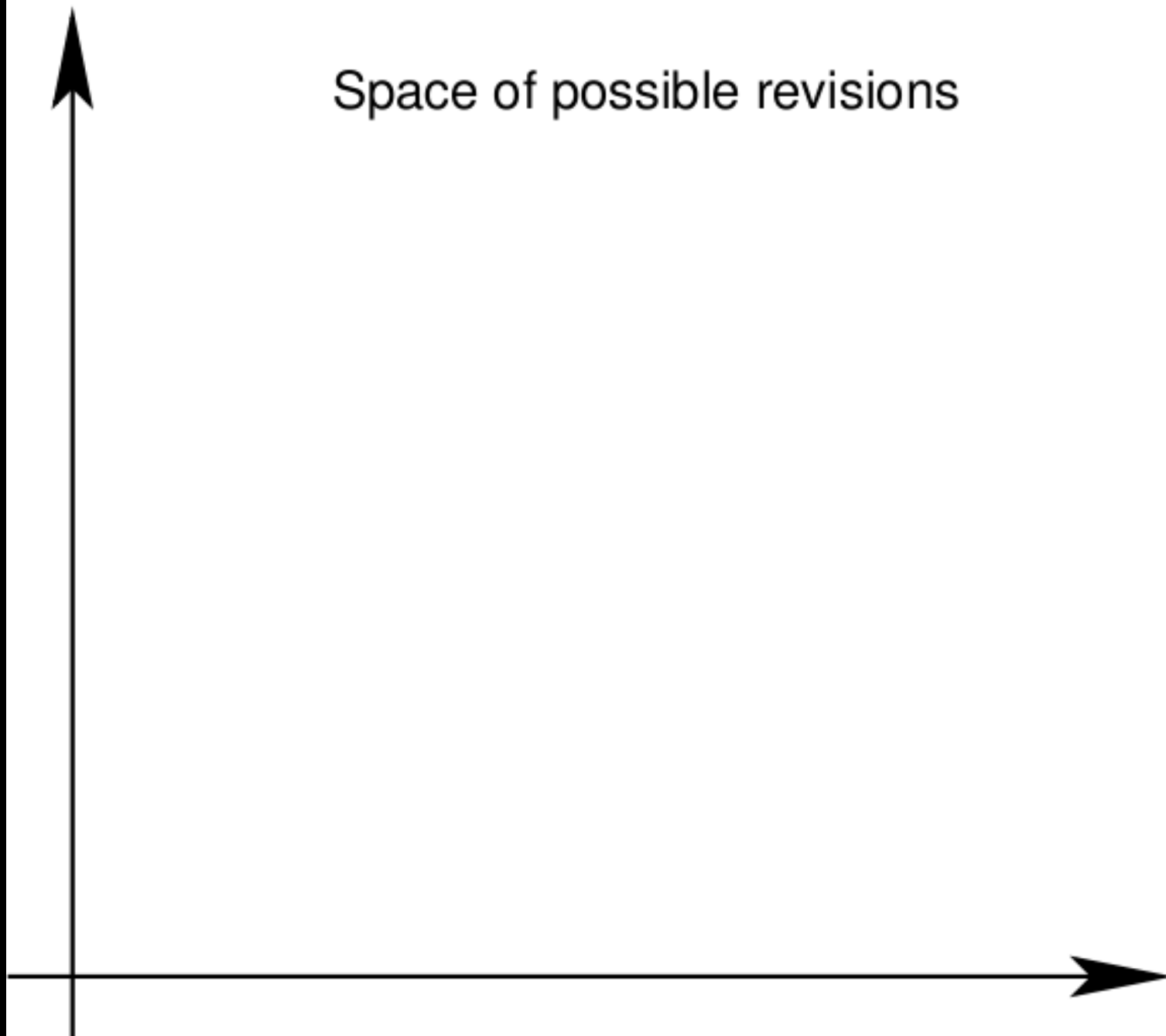


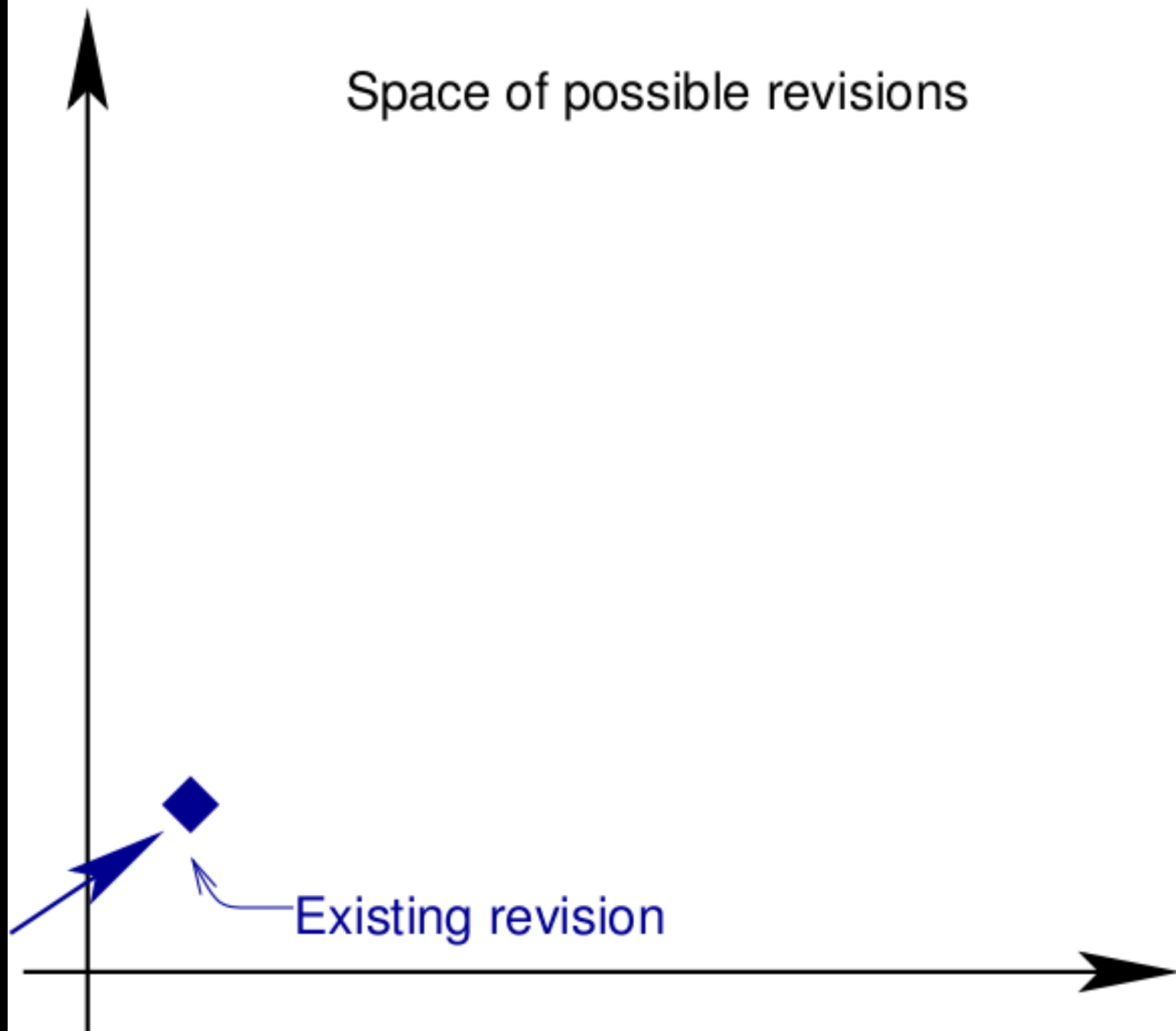
Figure 1.1. File Conflict



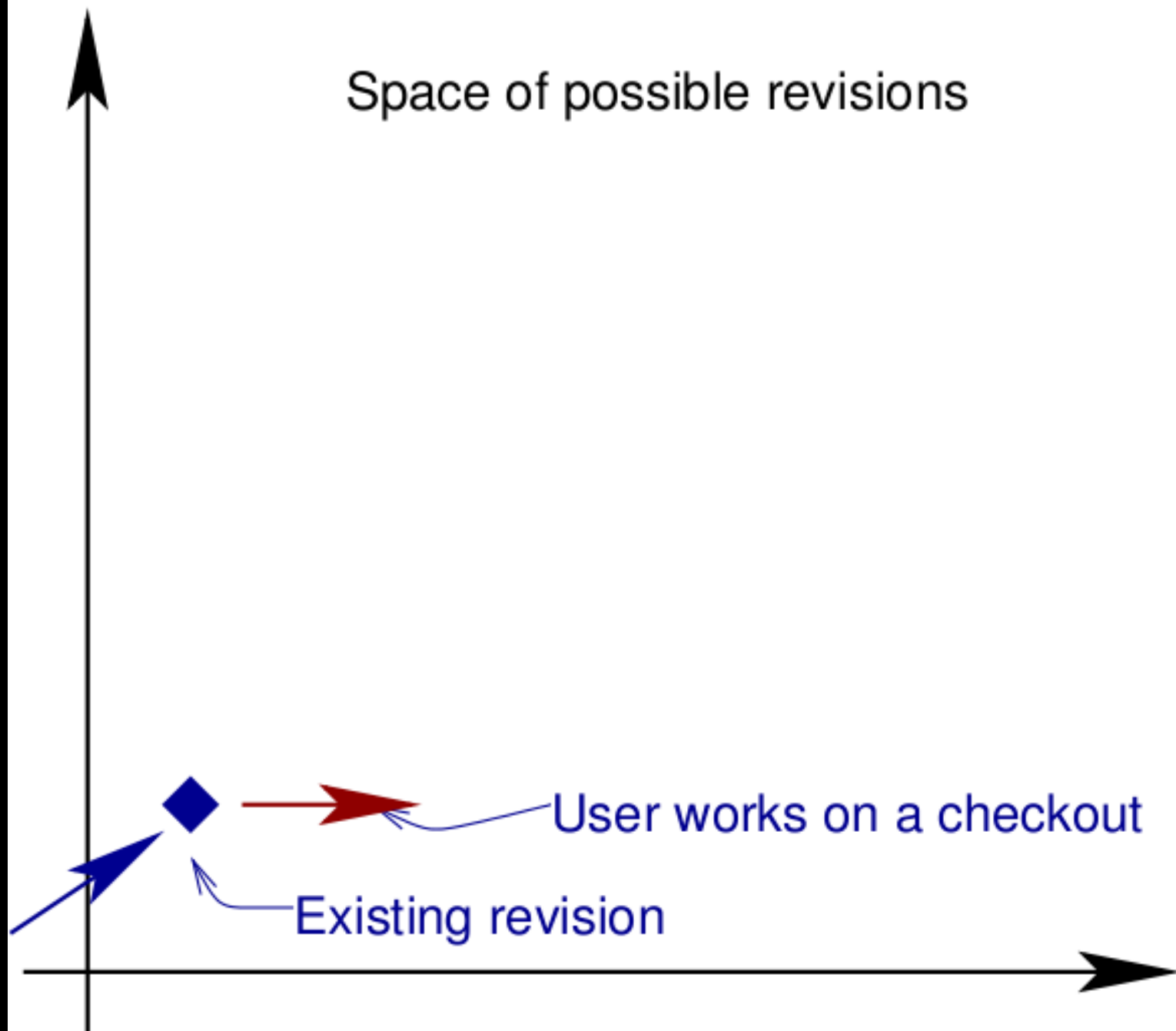
Commit/Update Approach



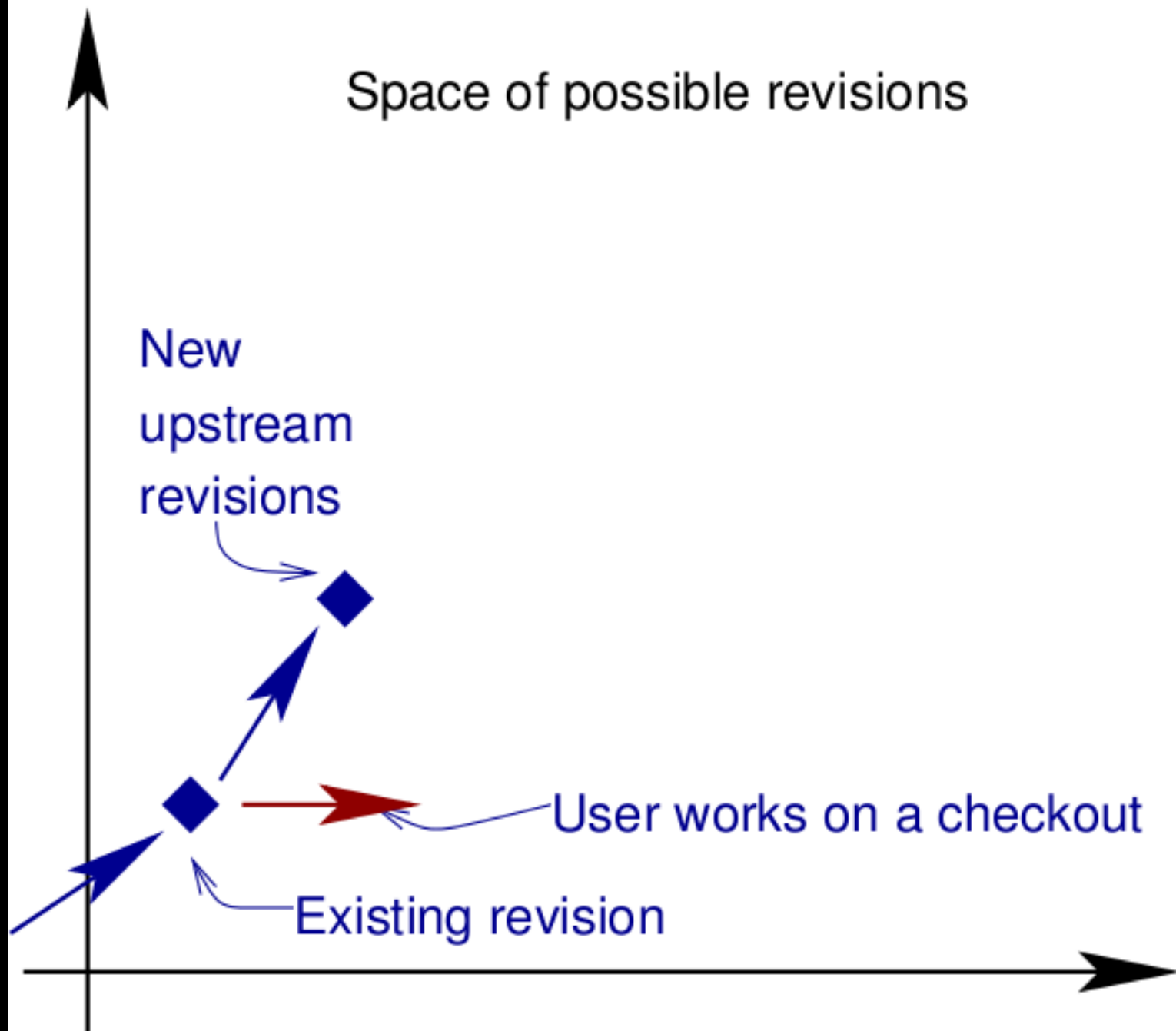
Commit/Update Approach



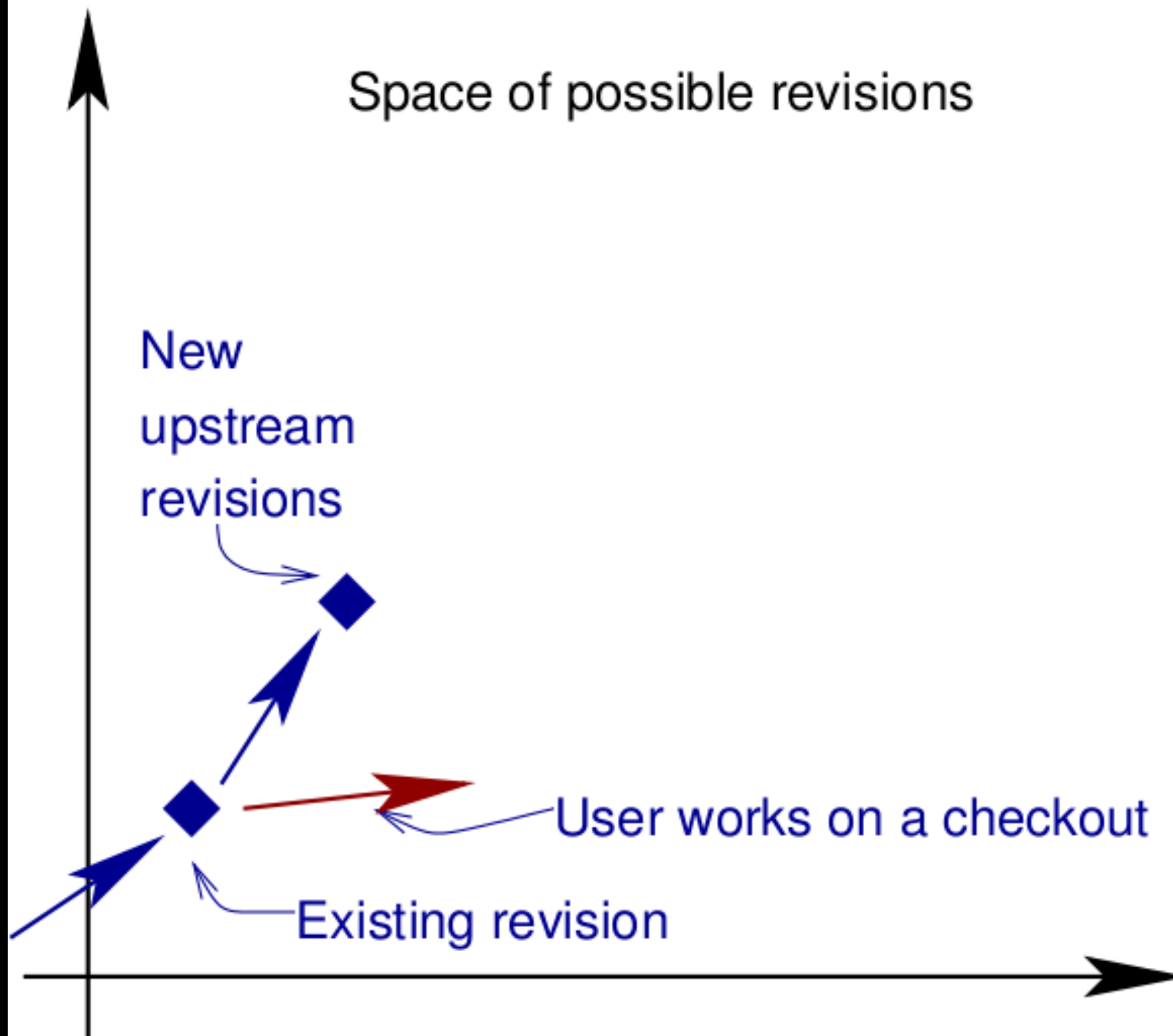
Commit/Update Approach



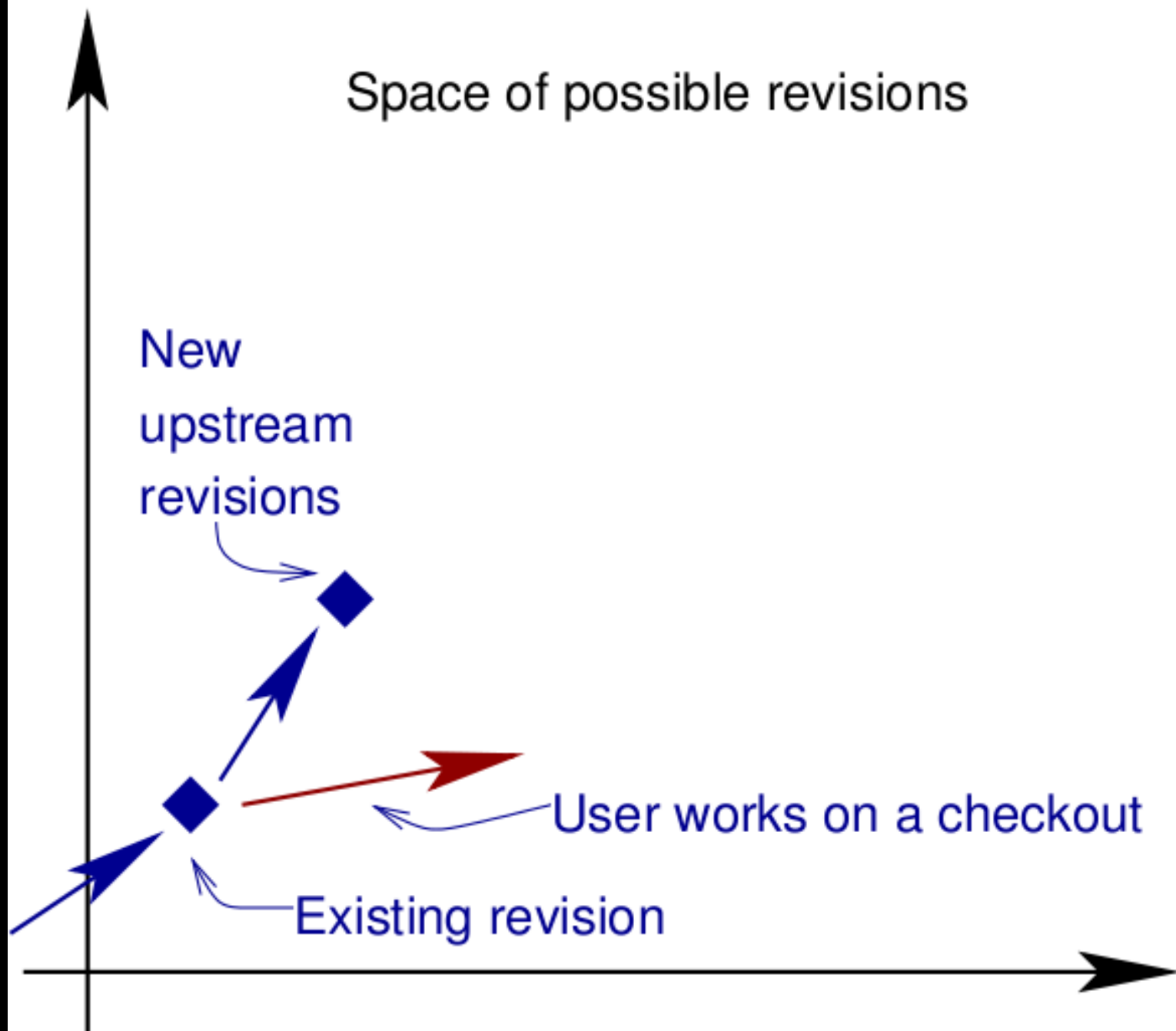
Commit/Update Approach



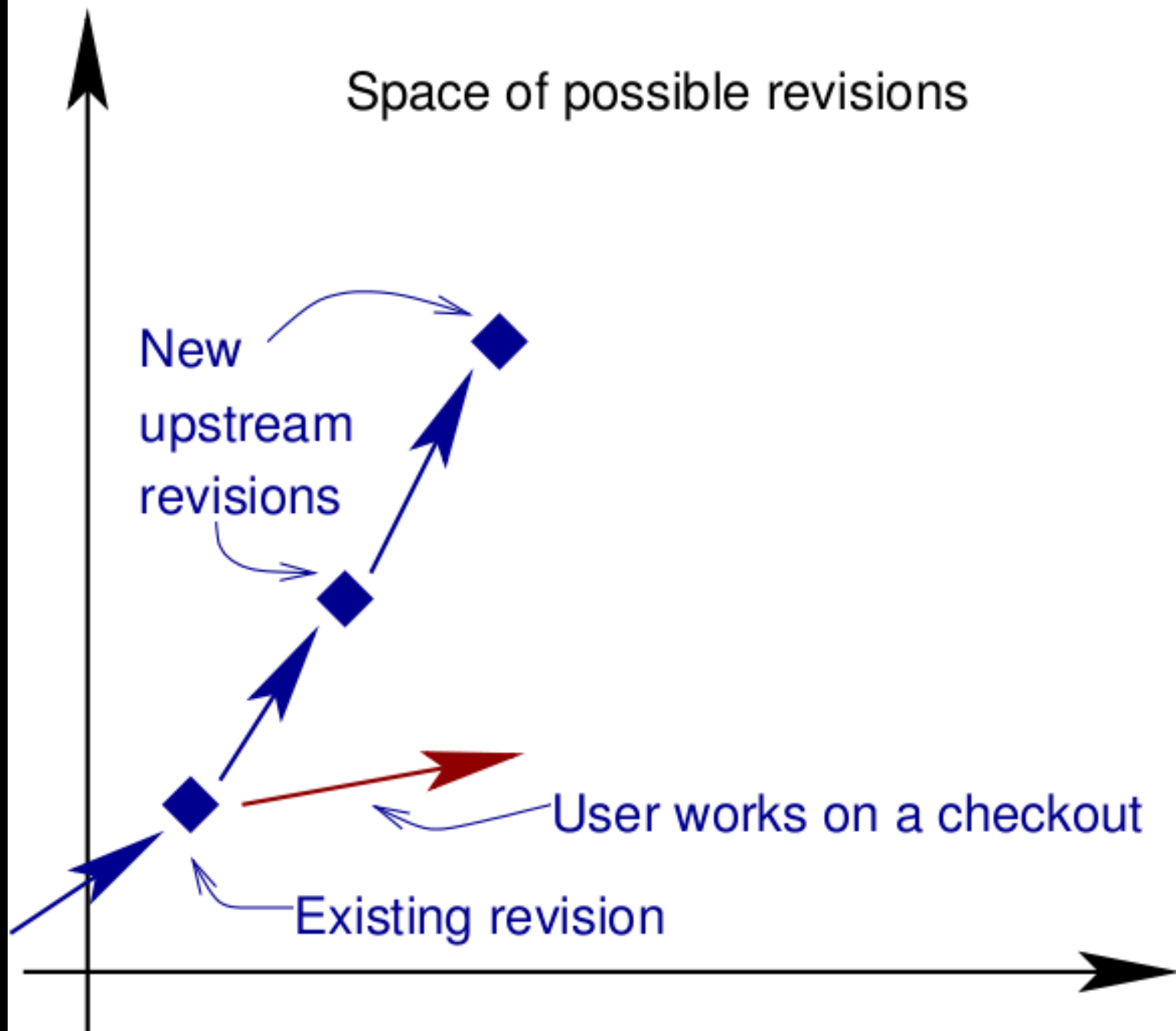
Commit/Update Approach



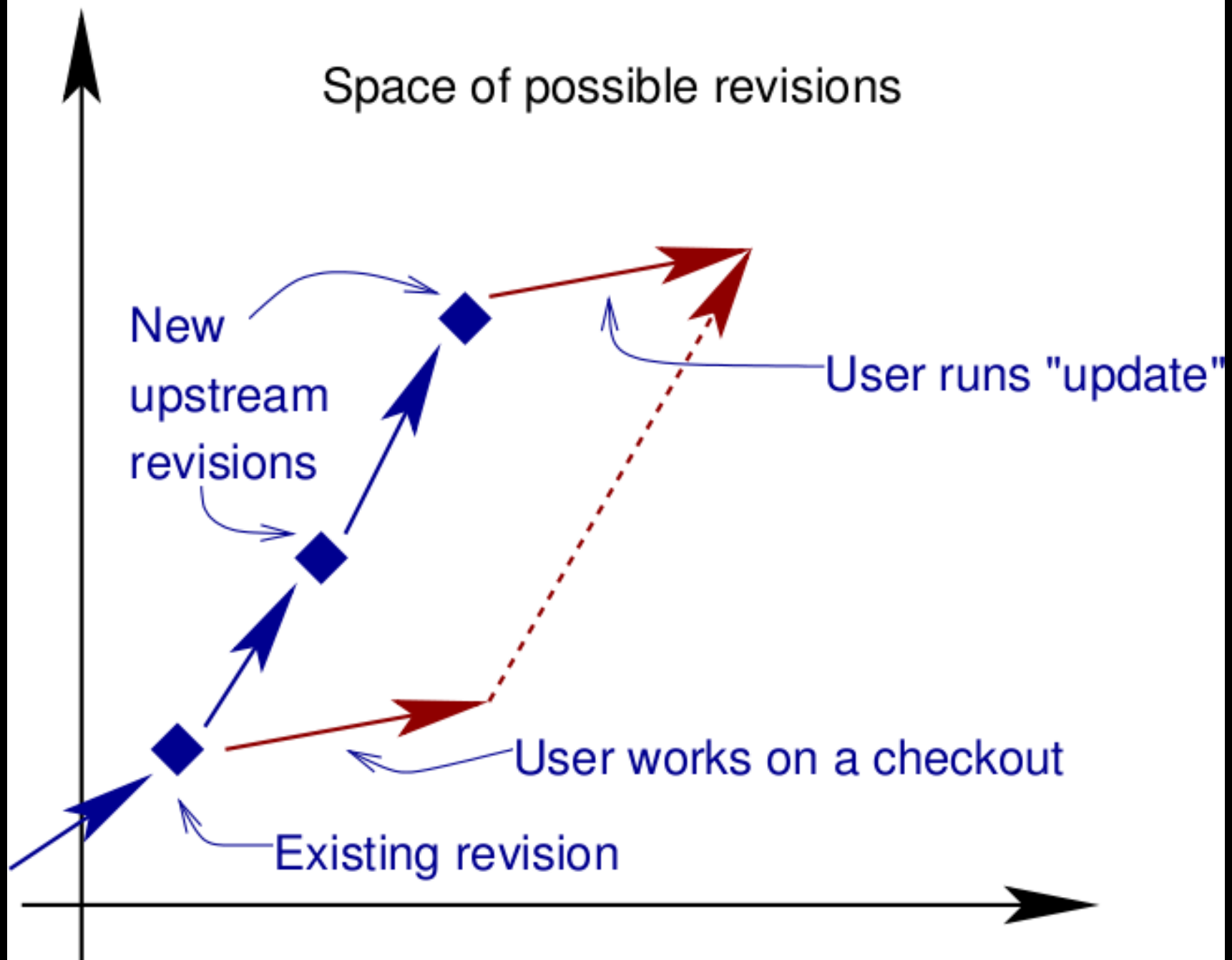
Commit/Update Approach



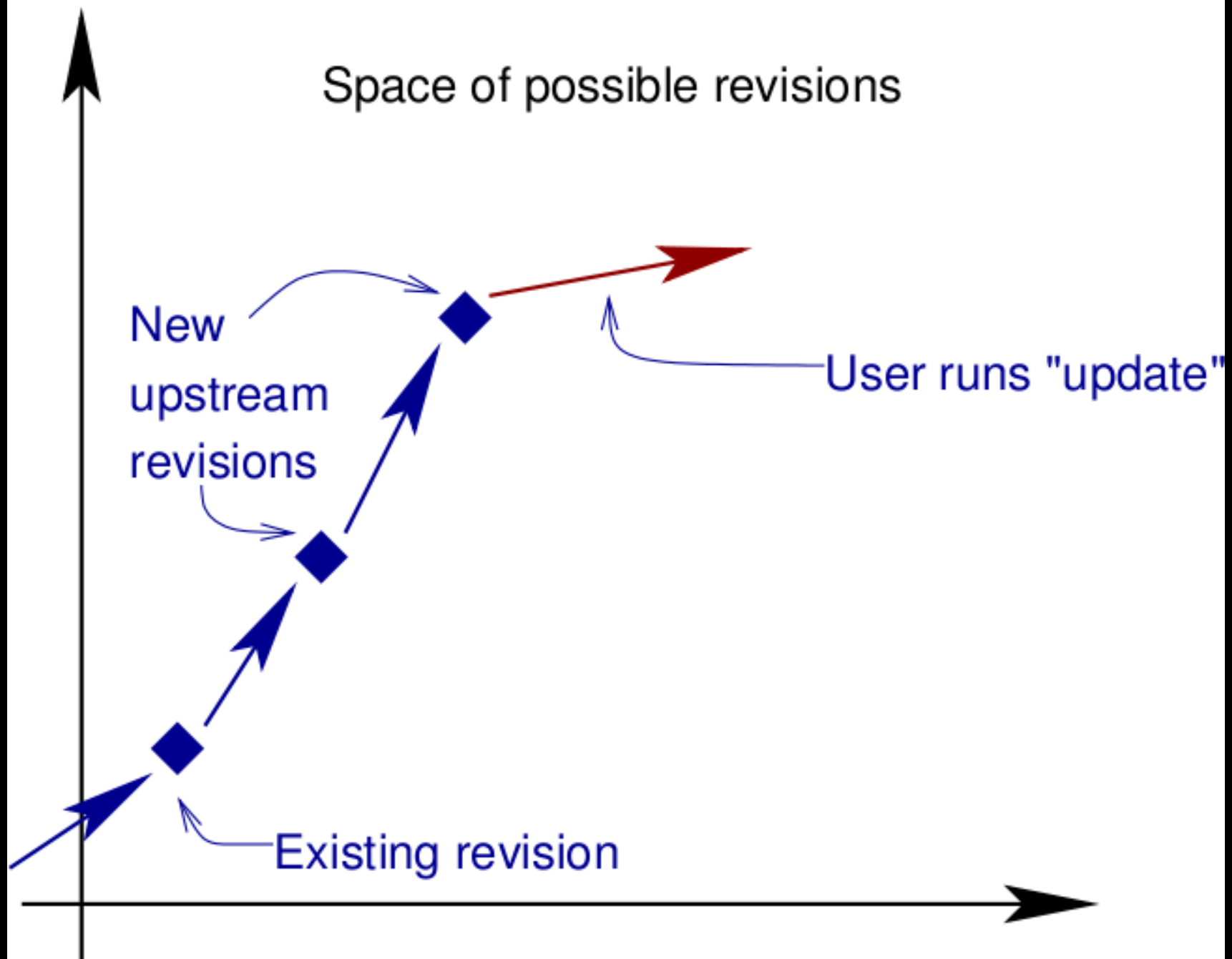
Commit/Update Approach



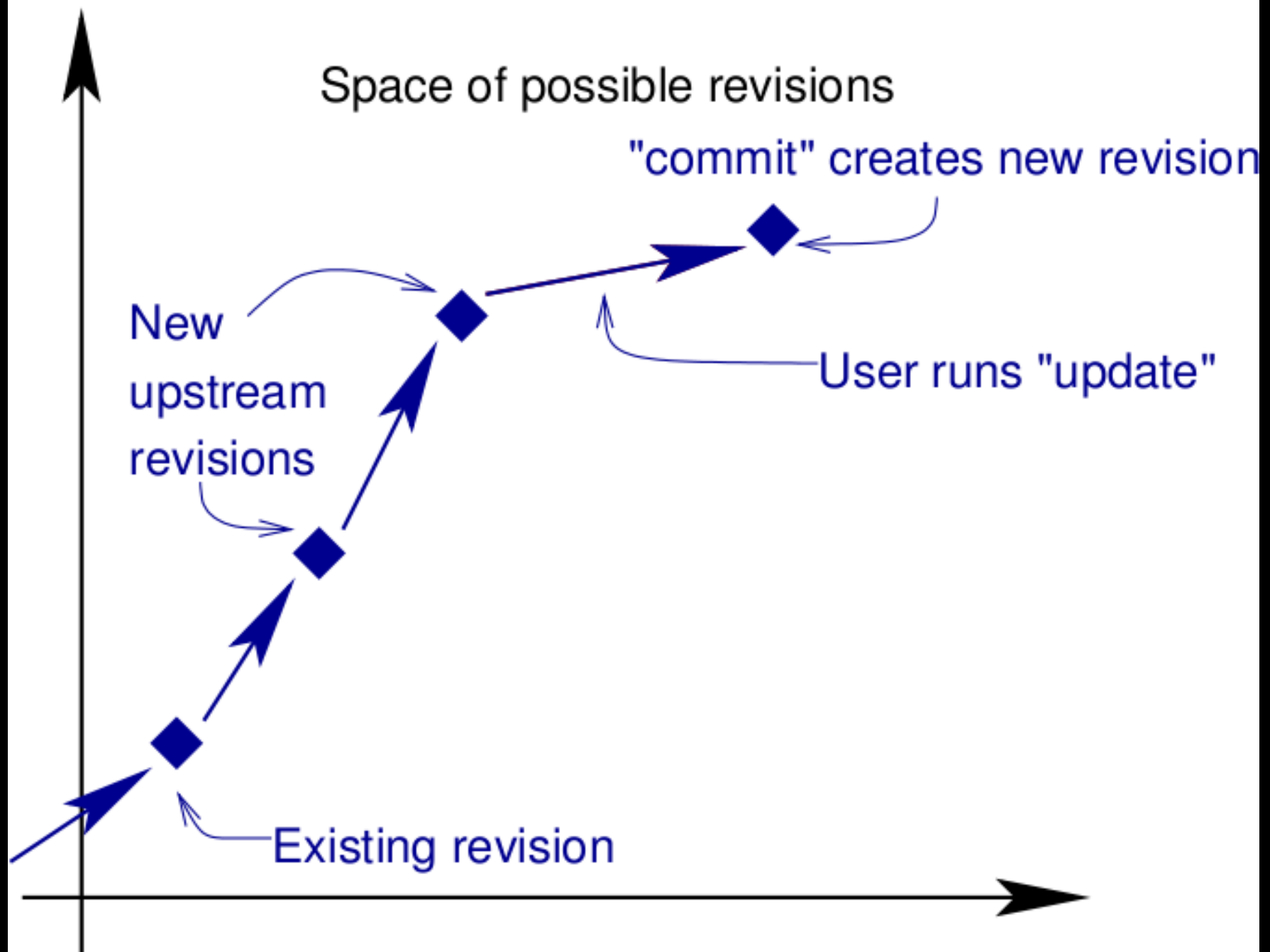
Commit/Update Approach



Commit/Update Approach



Commit/Update Approach



出發點

- 協同合作是軟體專案開發的要素
 - 所以我們有 SCM: CVS, Subversion, Mercurial, GNU Arch, SVK, Darcs, Git, Bzr..
- SCM (Source Control Management) 如何協助開發者?
 - Conflict / Merge



術語

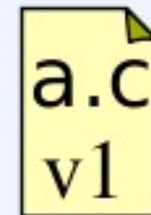
- Repository
- Pull / Push / Checkout
- Branch
- Merge
- Conflict
- Commit
- Revert



術語

• Repository

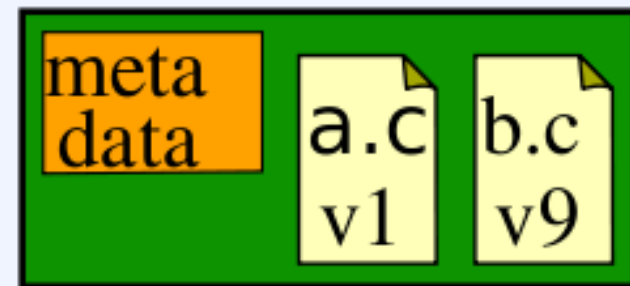
○ objects / blobs / diffs / deltas / patches



術語

• Repository

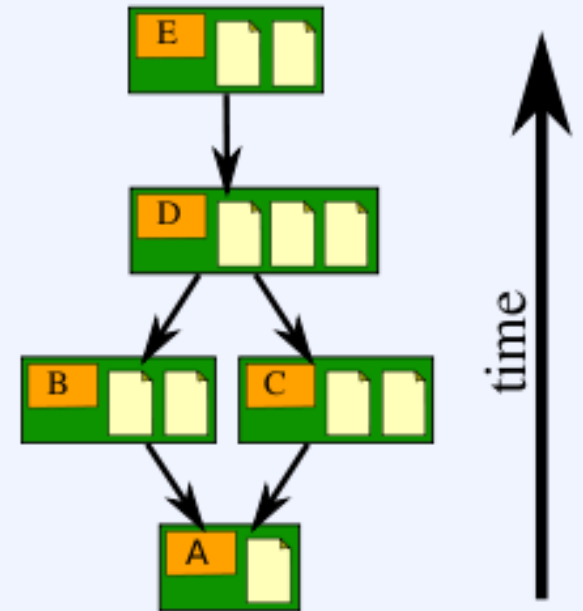
- objects / blobs / diffs / deltas / patches
- commits / changesets / revisions



術語

• Repository

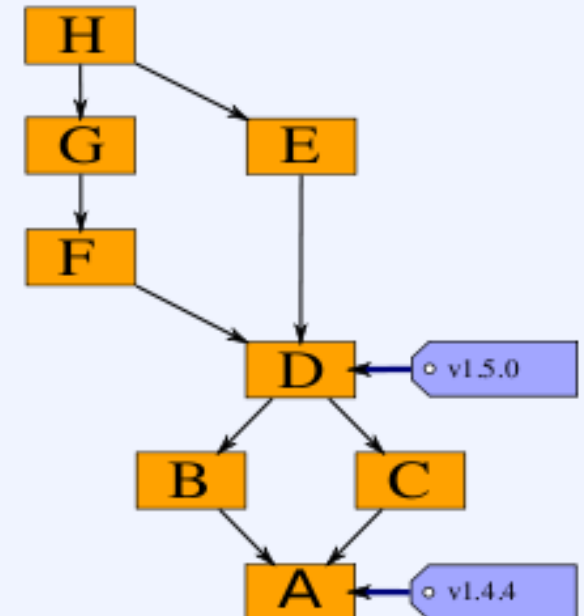
- objects / blobs / diffs / deltas / patches
- commits / changesets / revisions
- ancestry / history



術語

• Repository

- objects / blobs / diffs / deltas / patches
- commits / changesets / revisions
- ancestry / history
- tags / labels



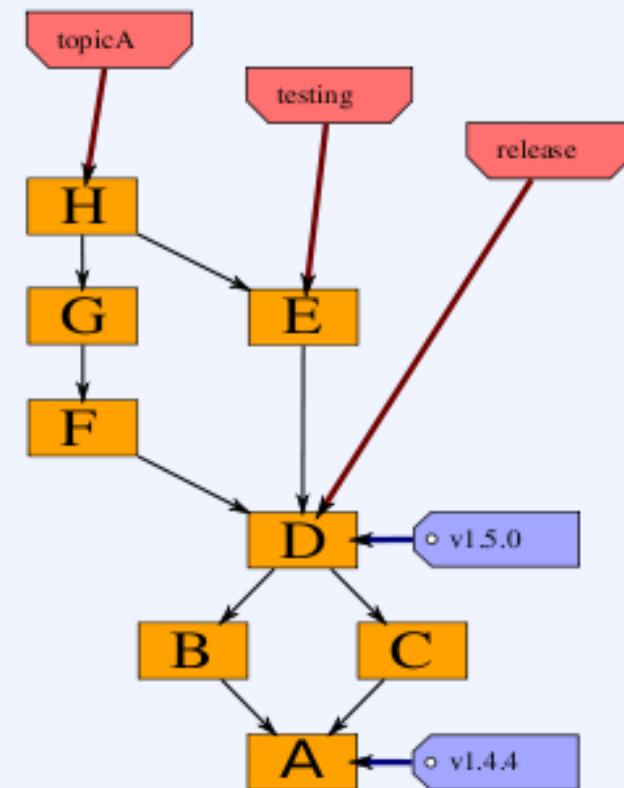
術語

Repository

- objects / blobs / diffs / deltas / patches
- commits / changesets / revisions
- ancestry / history
- tags / labels
- branches / heads

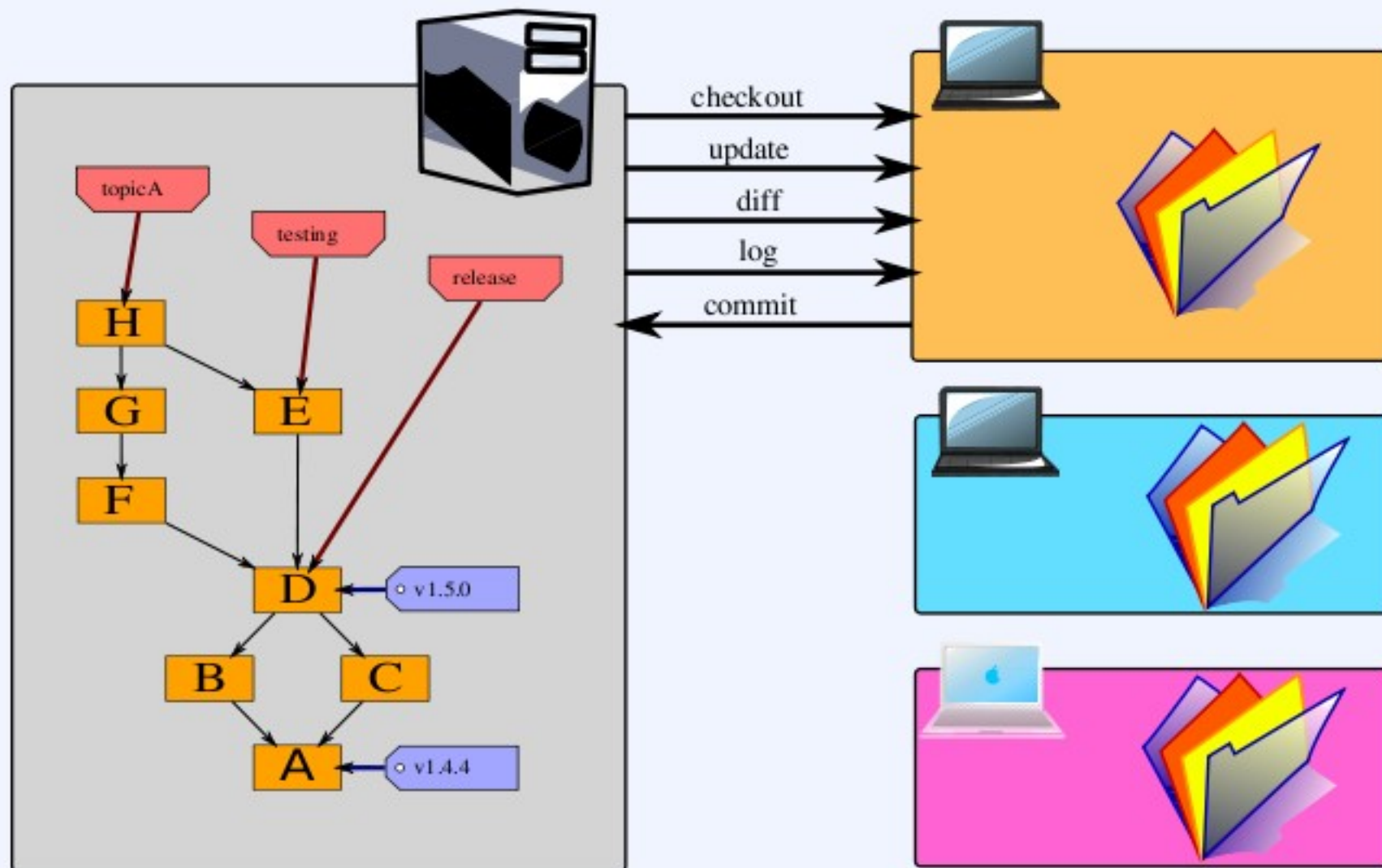
□ Working directory

- files
- list of files to add/delete



術語

• Checkout, Branch, Merge, Conflict, Commit, Revert



分散式版本控制系統



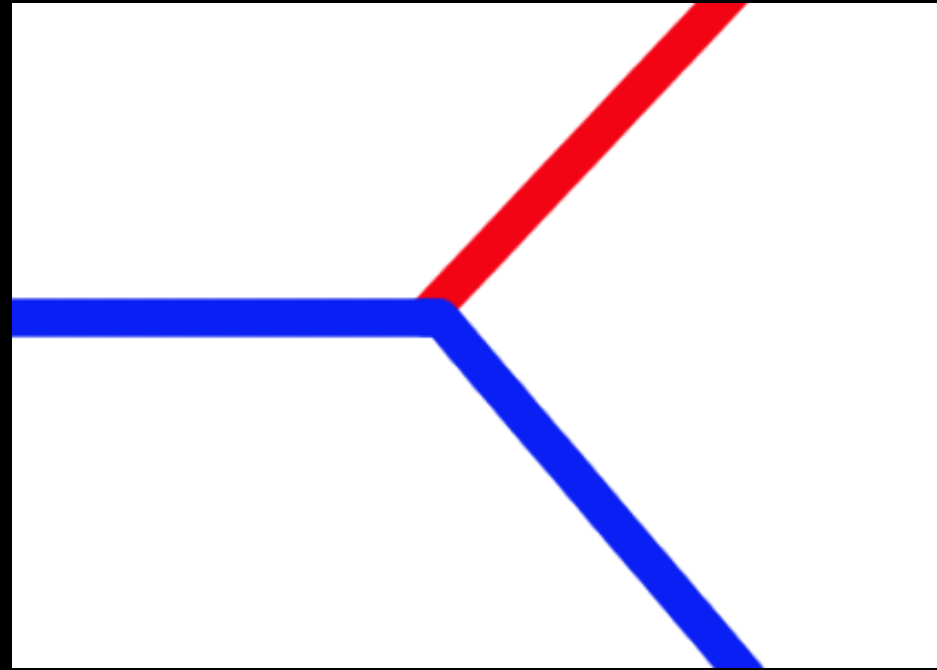
意味著 **fork** 嗎？

- 通常軟體專案的 **fork** 意味著，由不同團體的開發者接手維護並控制衍生的專案
- **fork == branch ?**

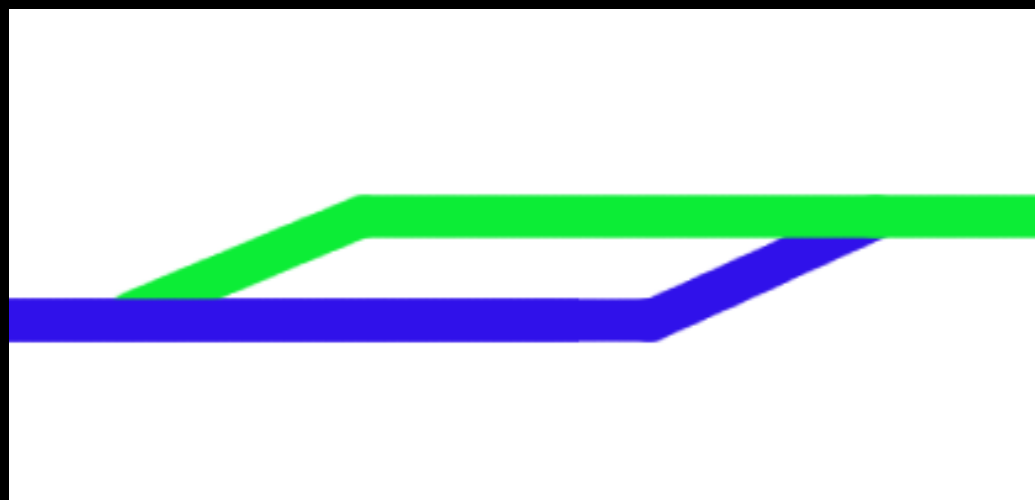
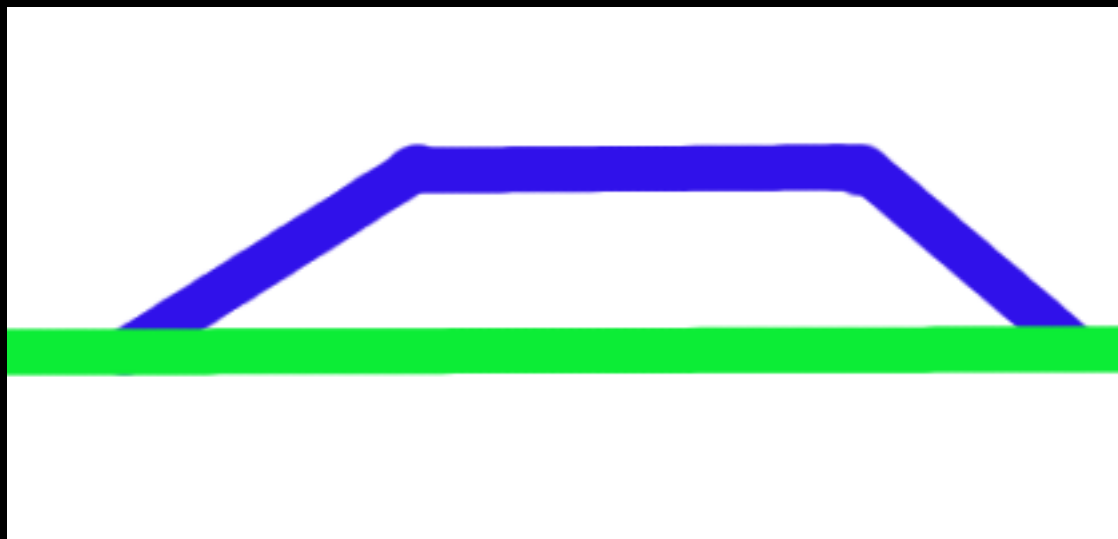


遊戲規則悄悄轉變：**Fork/Branch**

- 基於不同的需求，特別是涉及商業模式
- 基於某些功能 / 平台的獨立開發是有必要的，如 **GCC** 與 **Xorg** 等大型專案



Fork/Branch 的「整合」



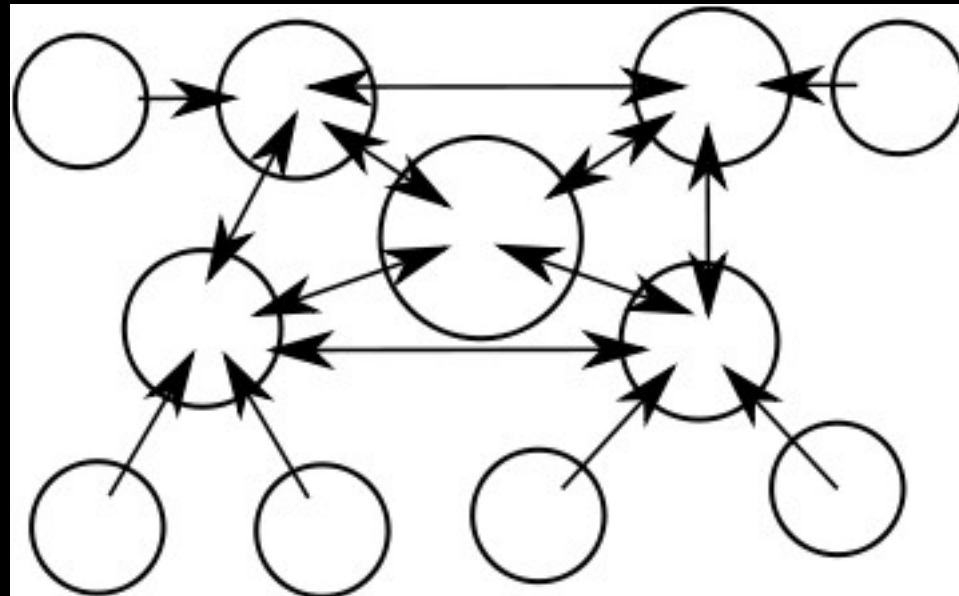
分散式版本控制系統概念

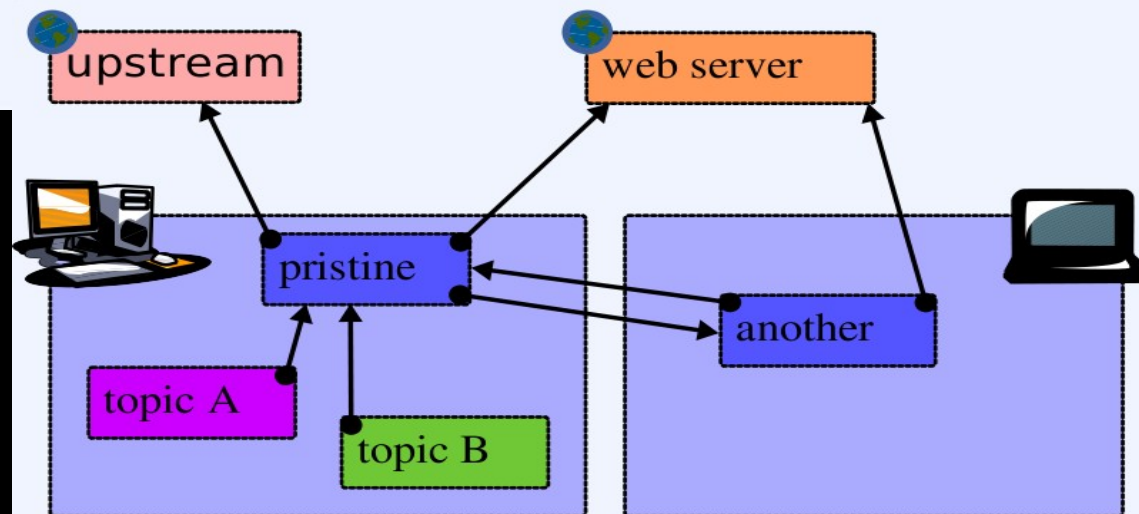
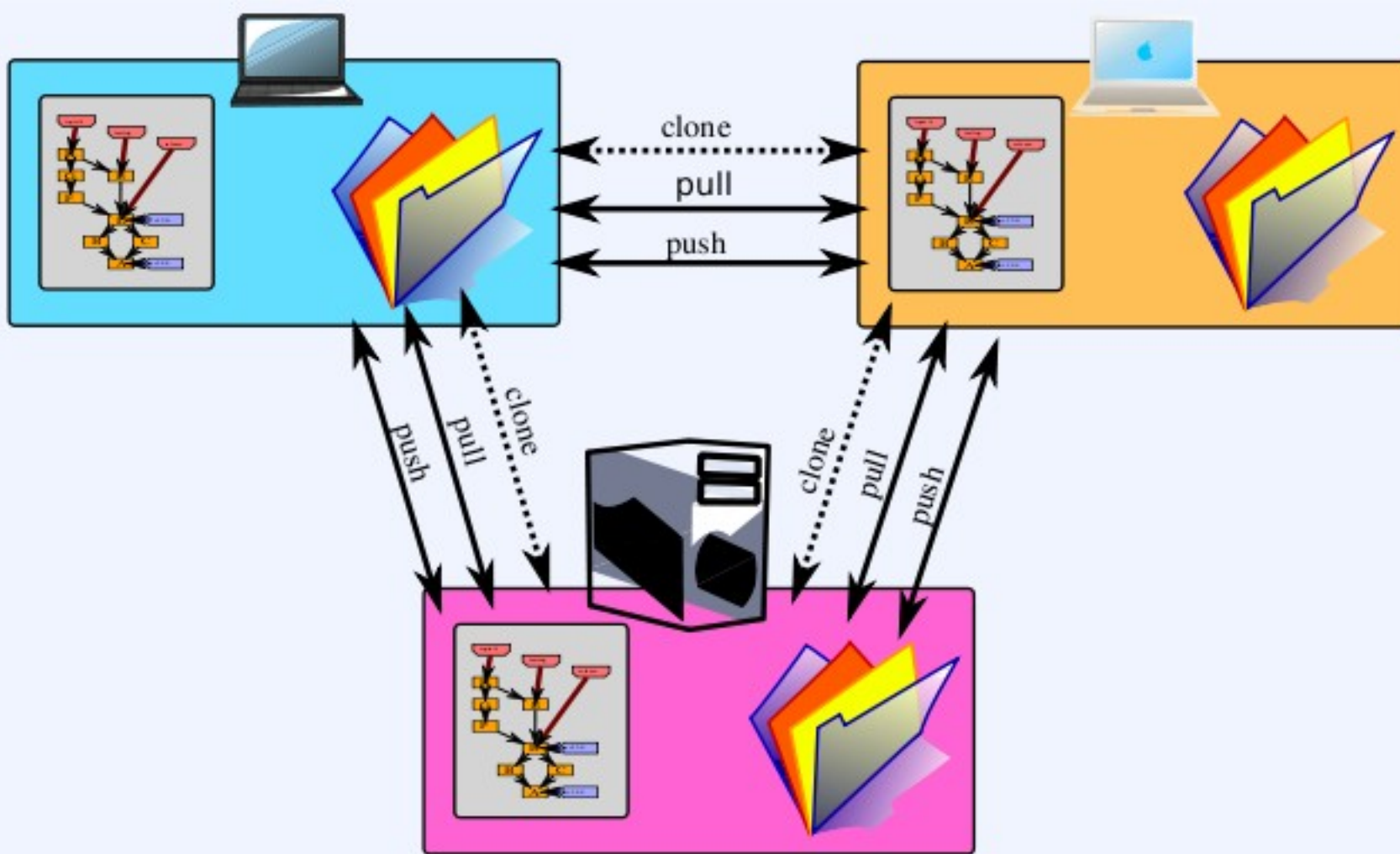
- 不只一處 **Repository**
- 每個使用者都擁有一份完整 **Repository** 並可讀寫存取的动作
- 開發者可公佈 (**publish**) 自己的 **repository** 並要求 **merge**



分散式版本控制系統概念

- 分散：不集中一處
- Branching** 變成理所當然的行為（核心想法）
- 對大型專案來說，有效加速開發速度
- 社群 (Web 2.0 式?) 開發



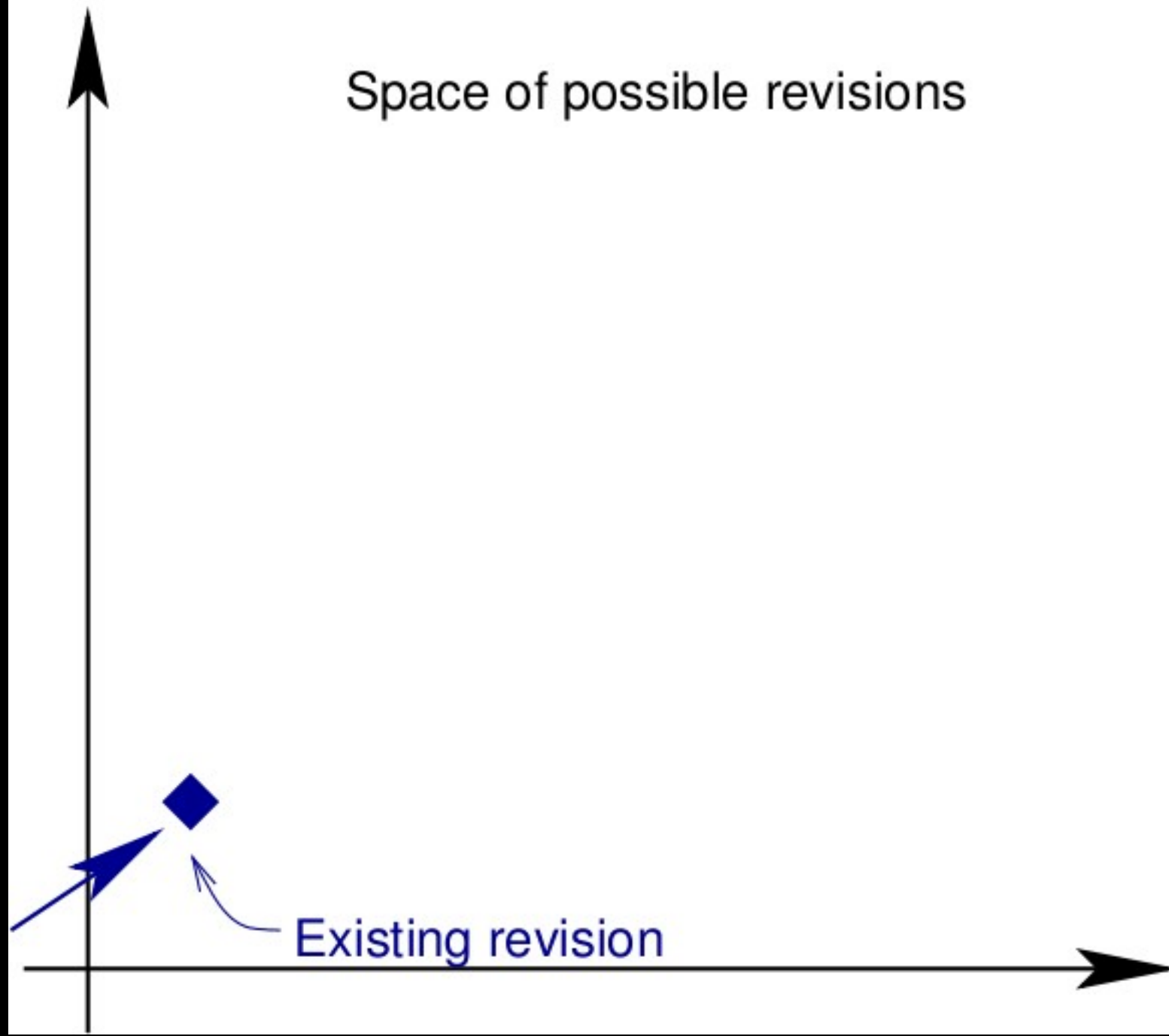


Merging

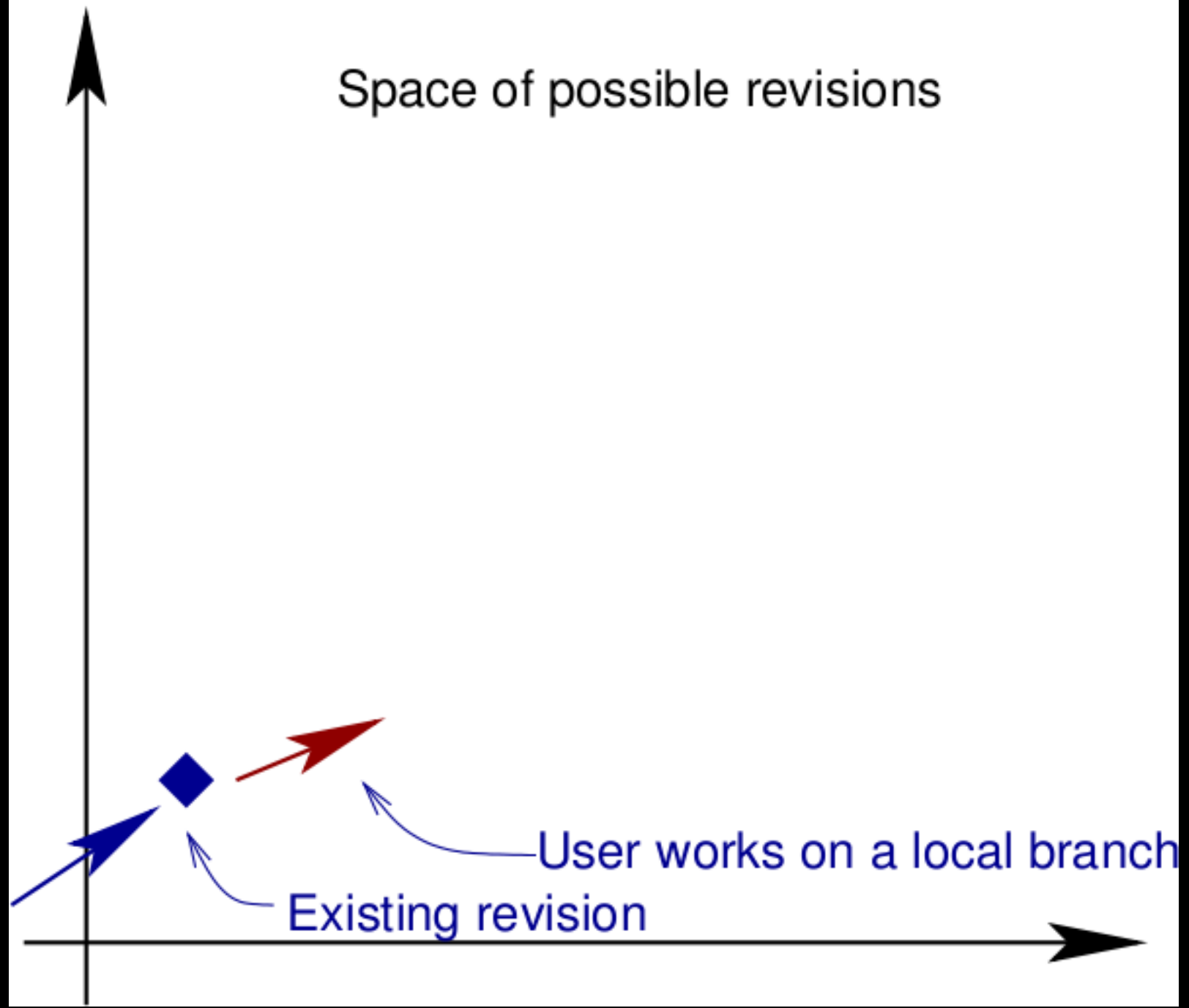
Space of possible revisions



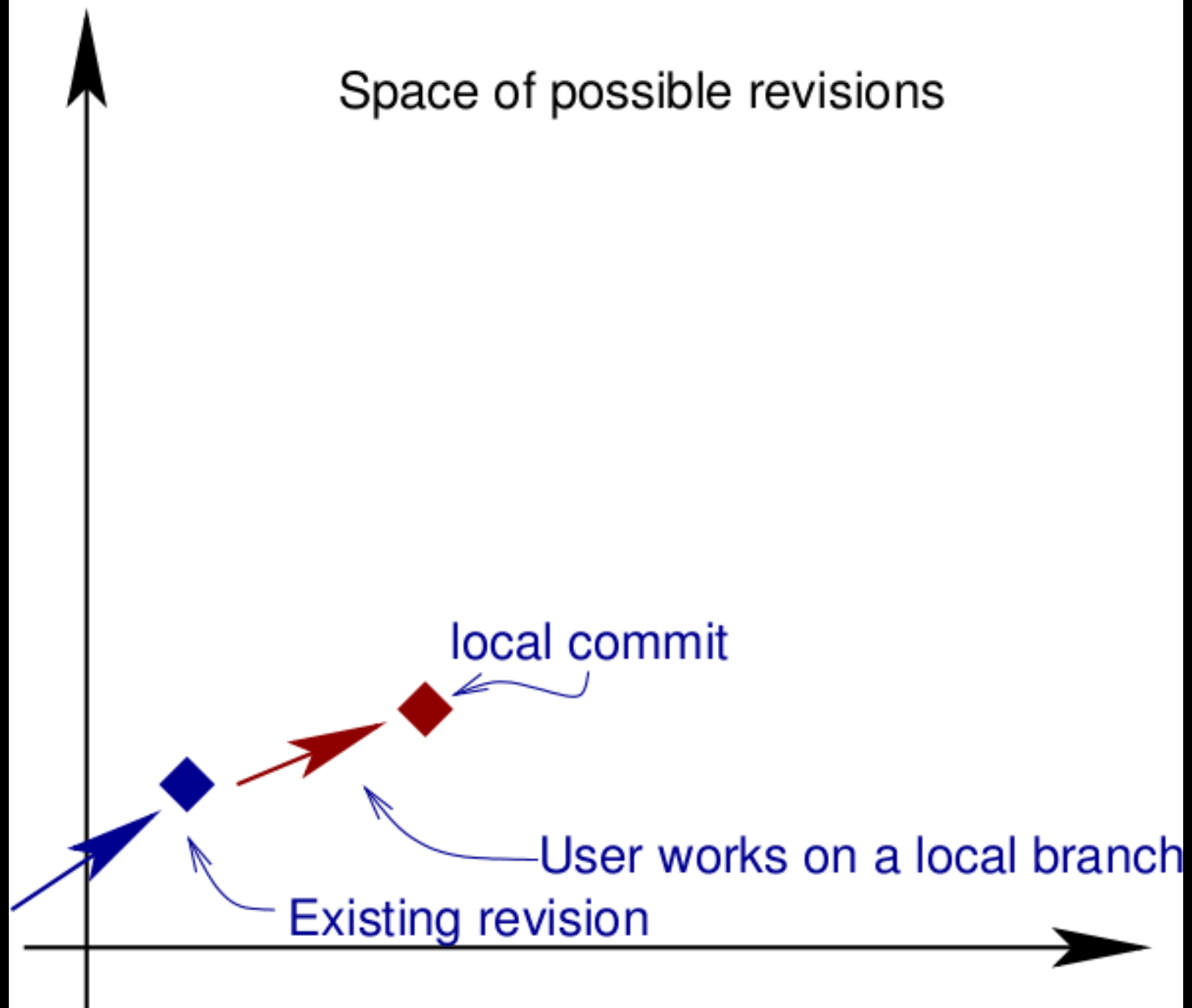
Merging



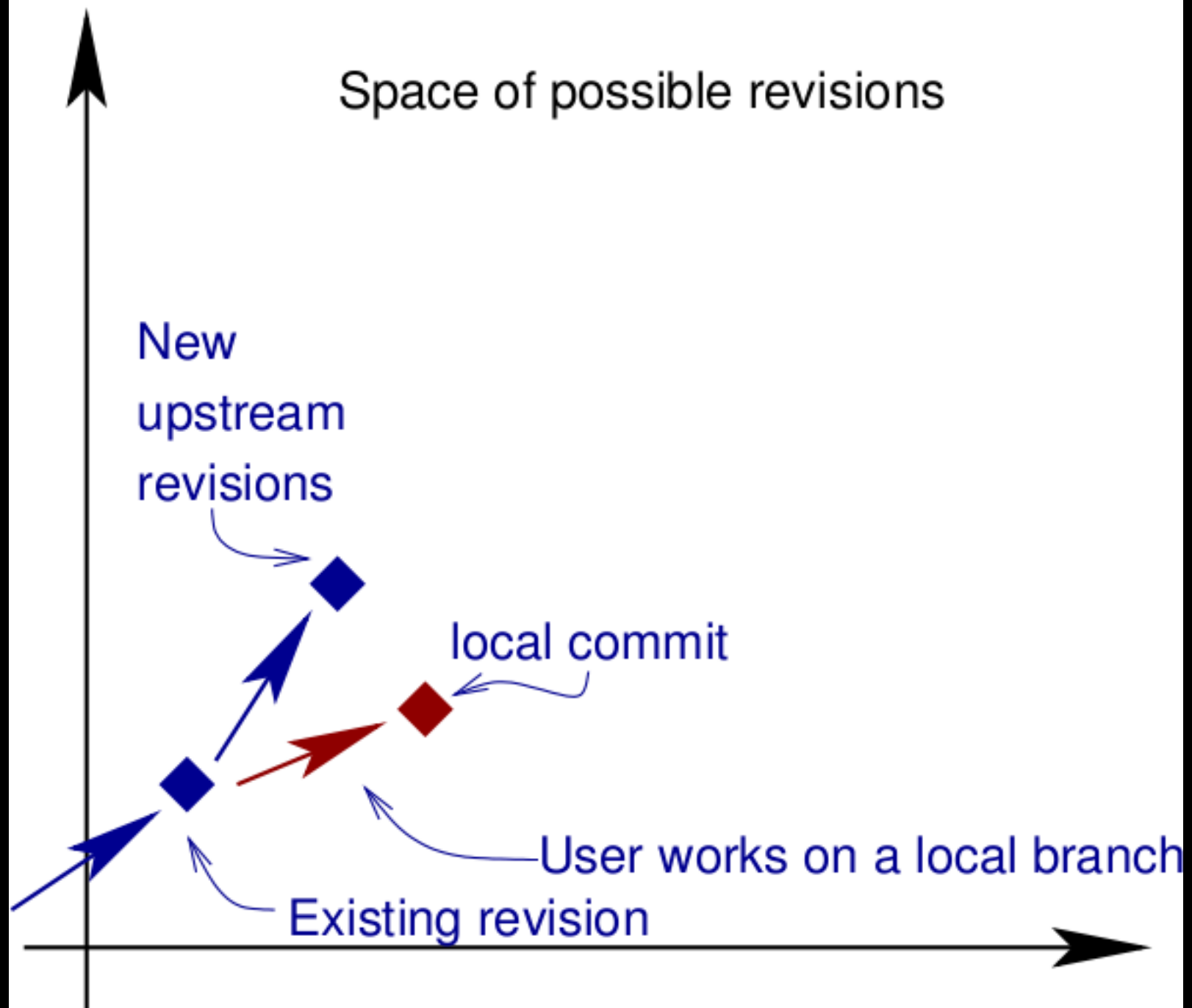
Merging



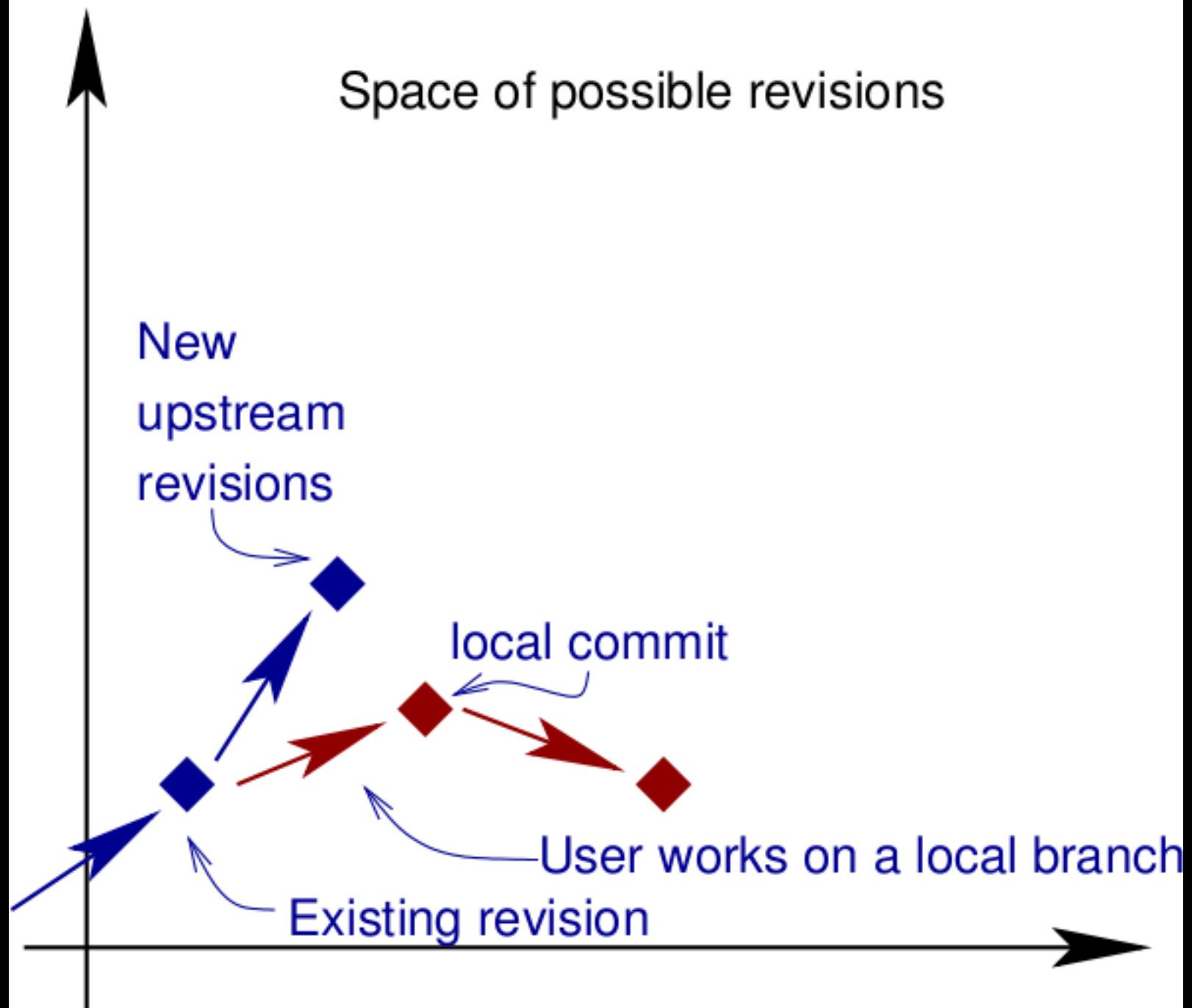
Merging



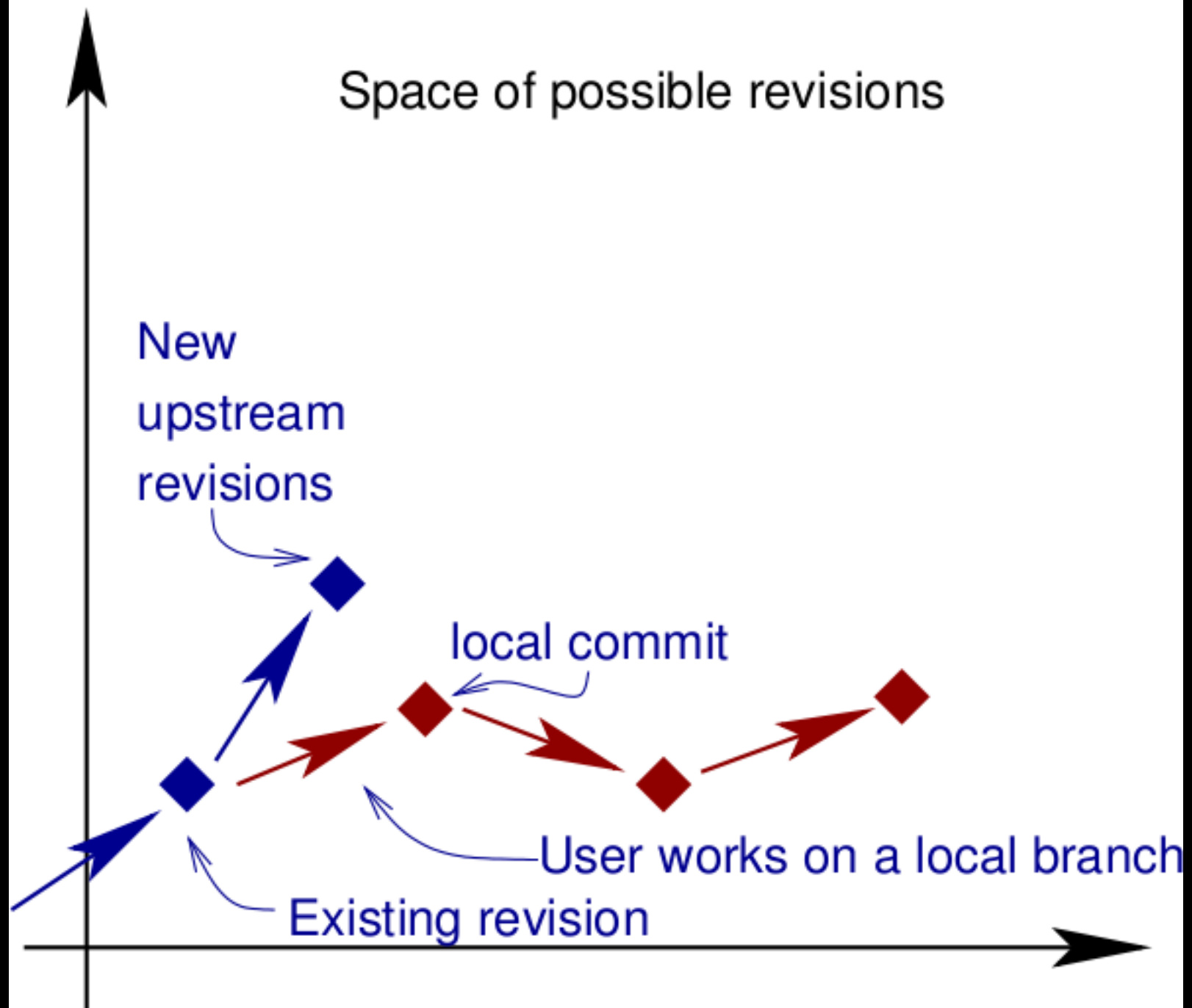
Merging



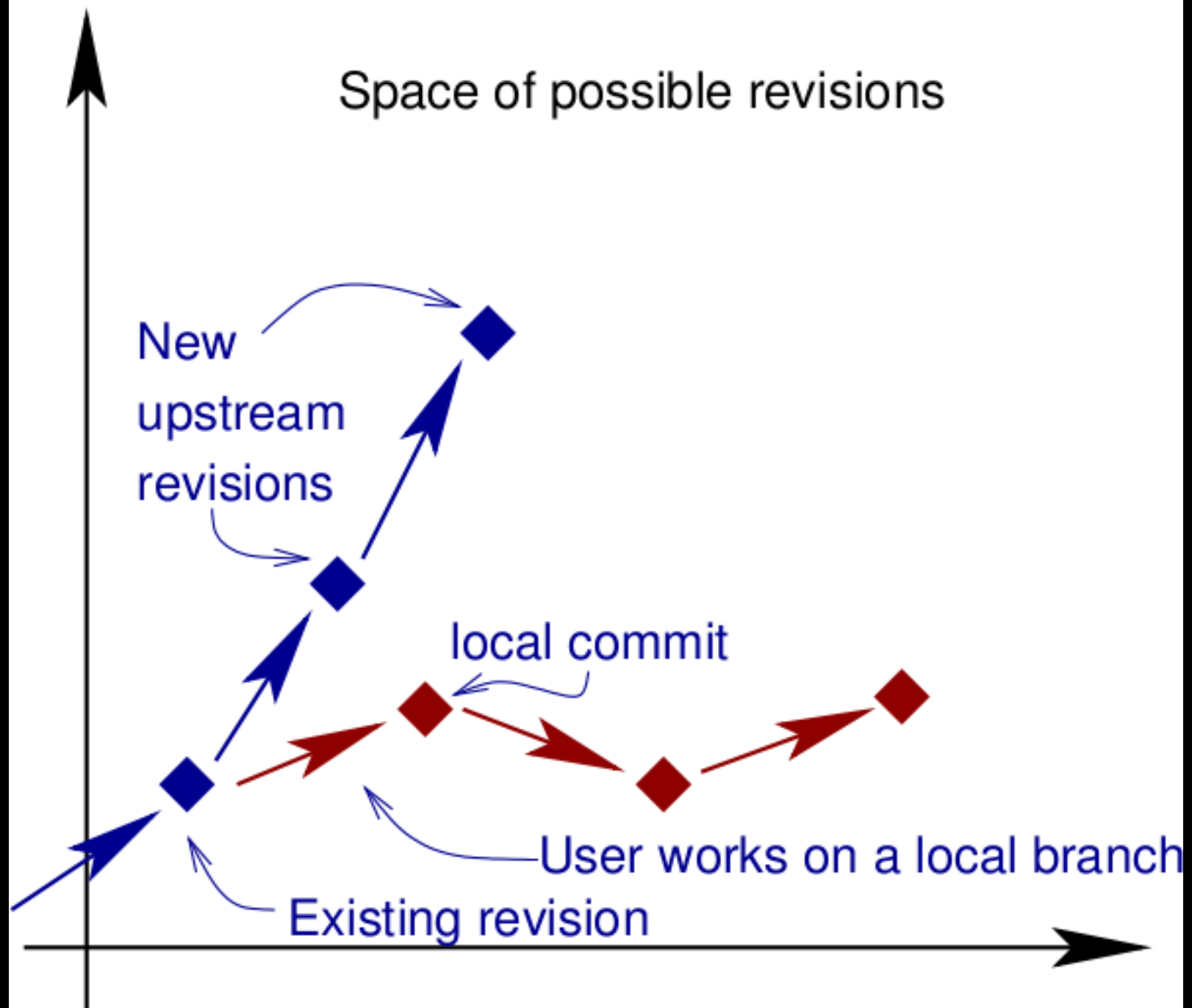
Merging



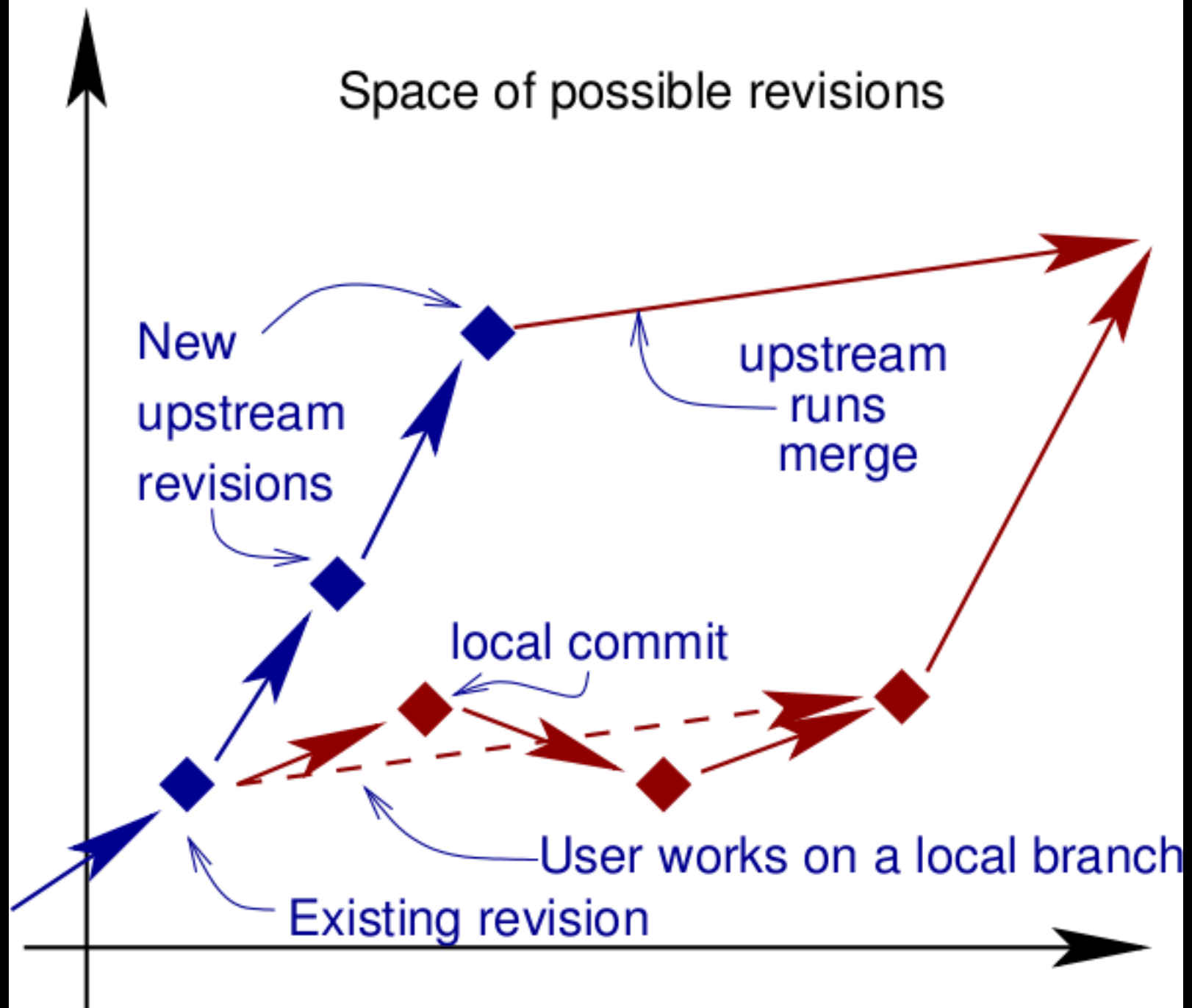
Merging



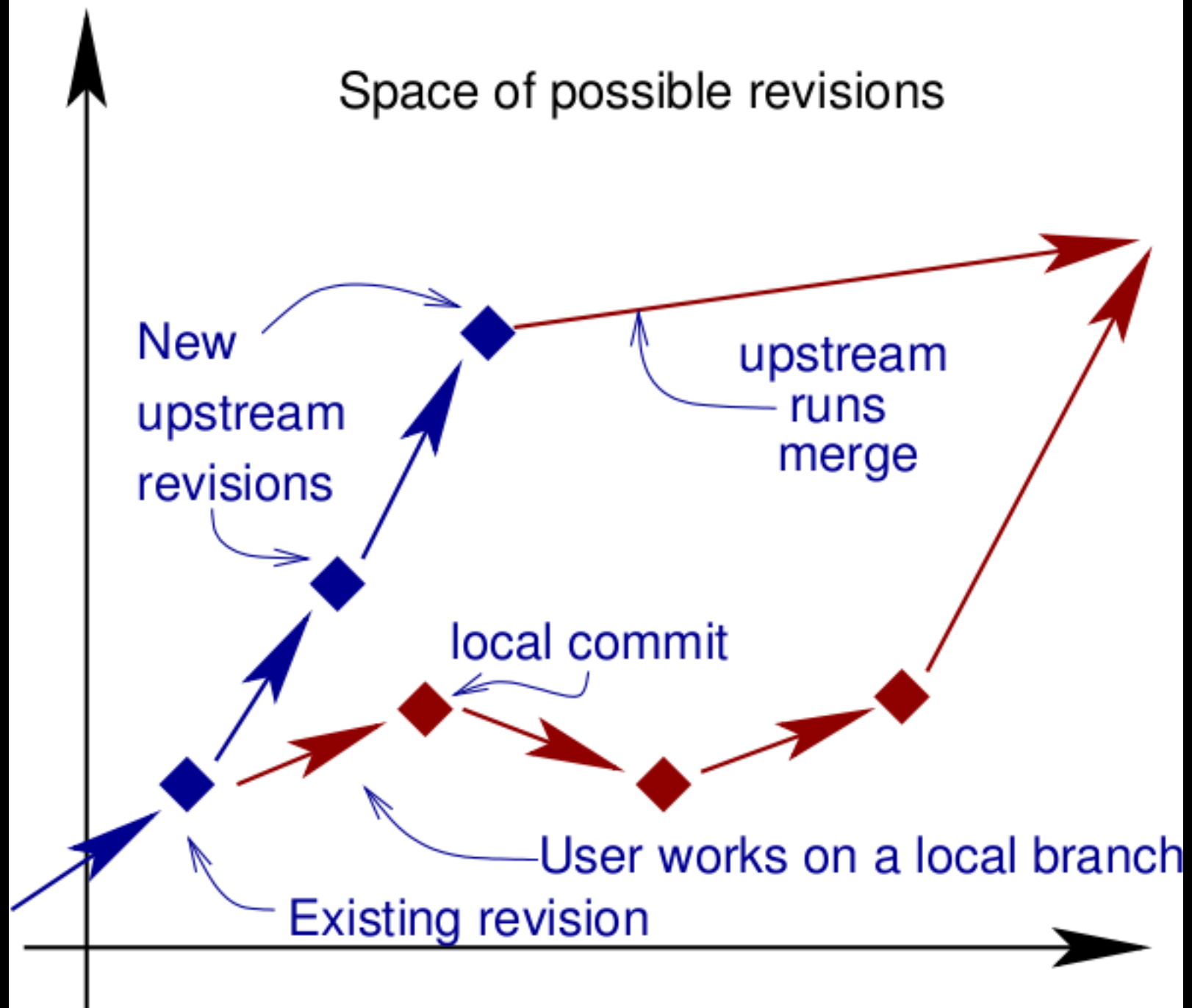
Merging



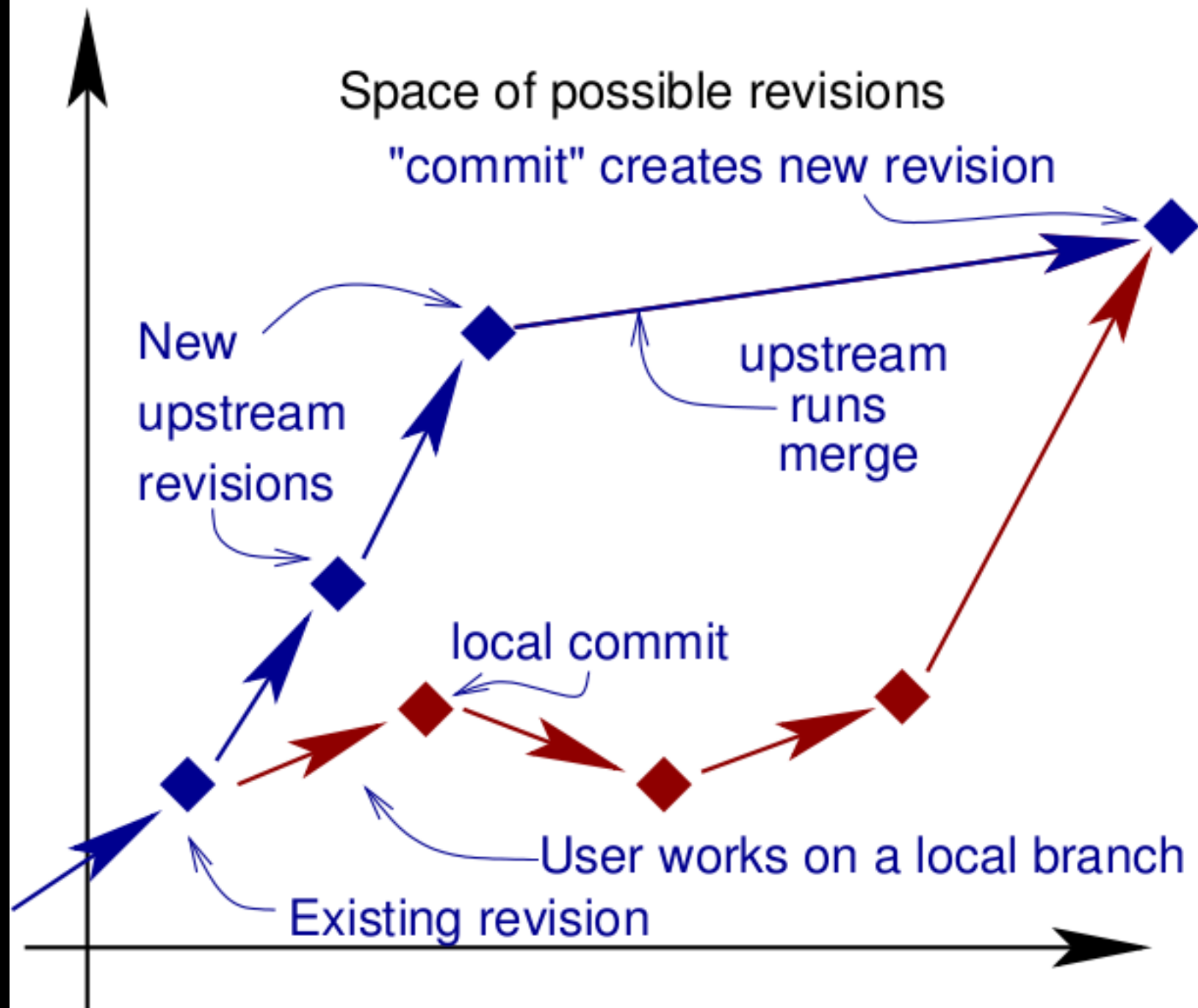
Merging



Merging



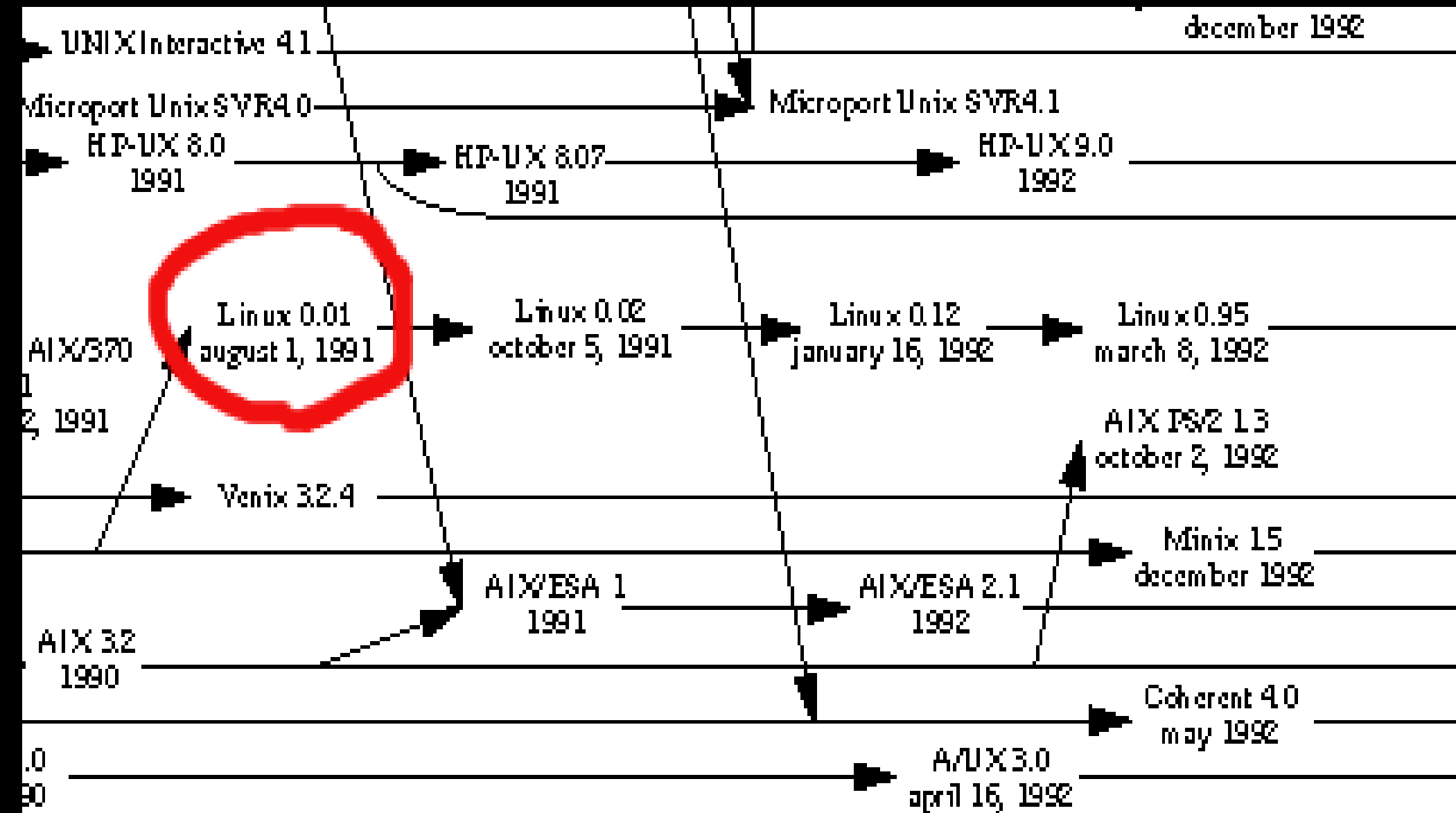
Merging

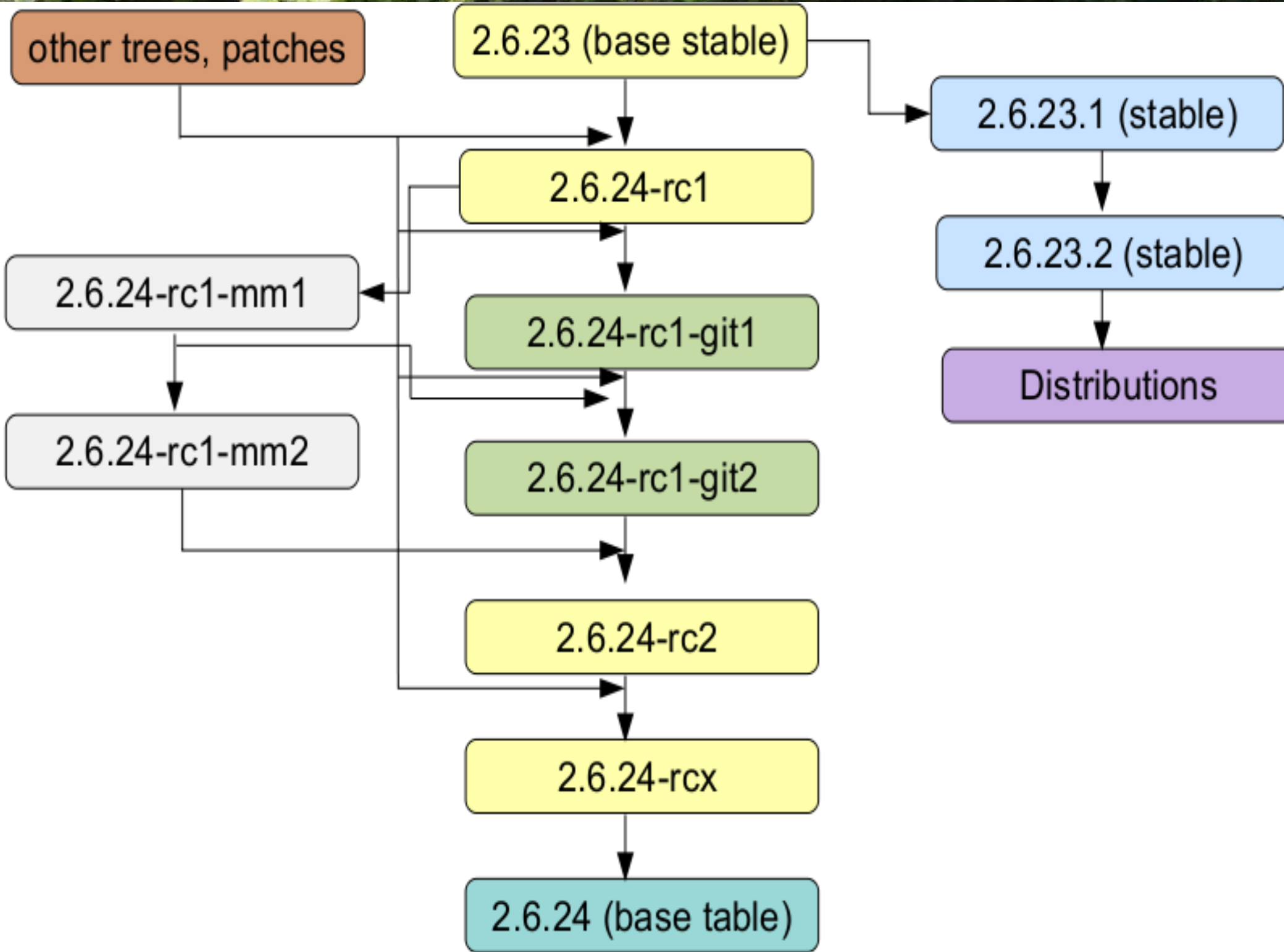


Git 核心概念

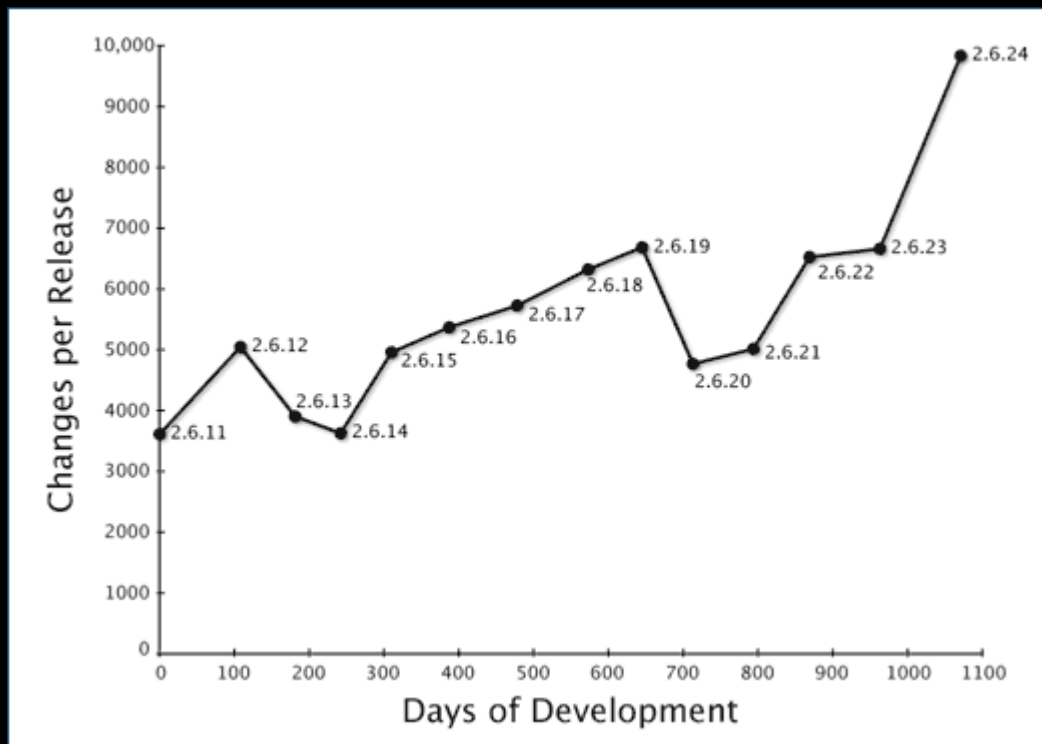


- 芬蘭赫爾辛基大學生 Linus Trovars 於 1991 年新聞組群發表所寫的 386/486 PC 用的作業系統 Linux
- 指標性的自由軟體





- 平均以每小時 **85.6** 行的速度增加
- 2.6.24 版本爲例，每小時 **7** 次更動
- 資料來源：
 - Linux Kernel Development - How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It- Greg Kroah-Hartman , OLS-2007.
 - <http://tree.celinuxforum.org/gitstat/index.php>





Git 背景

- 2002 年左右 Linus Torvald 採用封閉的 BitKeeper(bk) 分散式版本控制系統，降低日趨複雜的核心開發的難度
- Richard Stallman 為此提出警示，包含 Alan Cox 在內的重要核心開發拒絕使用 bk
 - 違反”freedom is more important than convenience.” 的信念

The logo for BitKeeper, featuring three horizontal bars of increasing length to the left of the text.

BITKEEPER
SOURCE MANAGEMENT



Git 背景

- 2005 年 Linus Torvald 著手實做期望中的 Decentralized SCM -- git
- Linux Kernel 的獨特性：或許不是最大的自由軟體專案，但是最活躍的
- Linus: “And then realize that nothing is perfect. Git is just *closer* to perfect than any other SCM out there.”



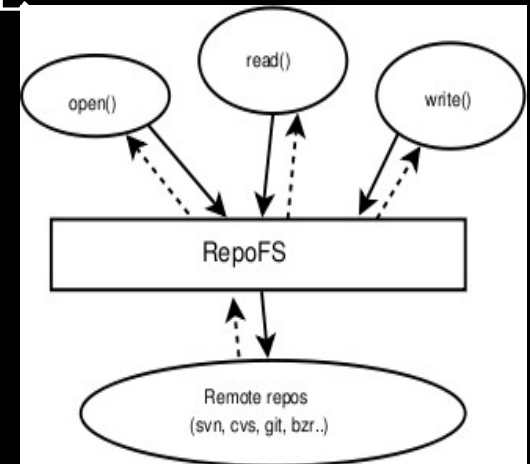
Git 背景

- 考量 SCM 變遷
 - 集中式：單一 repository 僅能由 core team 寫入
 - CVS, Subversion, ...
 - 分散式：到處都有 repository 並有完整的歷史紀錄 + 本地端更動
 - GNU Arch, Bzr, ...
- git：不僅分散，而且 SCM 應該是「檔案工具」，而不是限制開發者的「制度」



Git 目標 (1)

- 完全的分散性
- 沒有集中 repository
- Peer-to-Peer：輕易建立以特定遠端 repository 為基礎的 repository，並重新發佈
- 支援複雜的 merge 處理



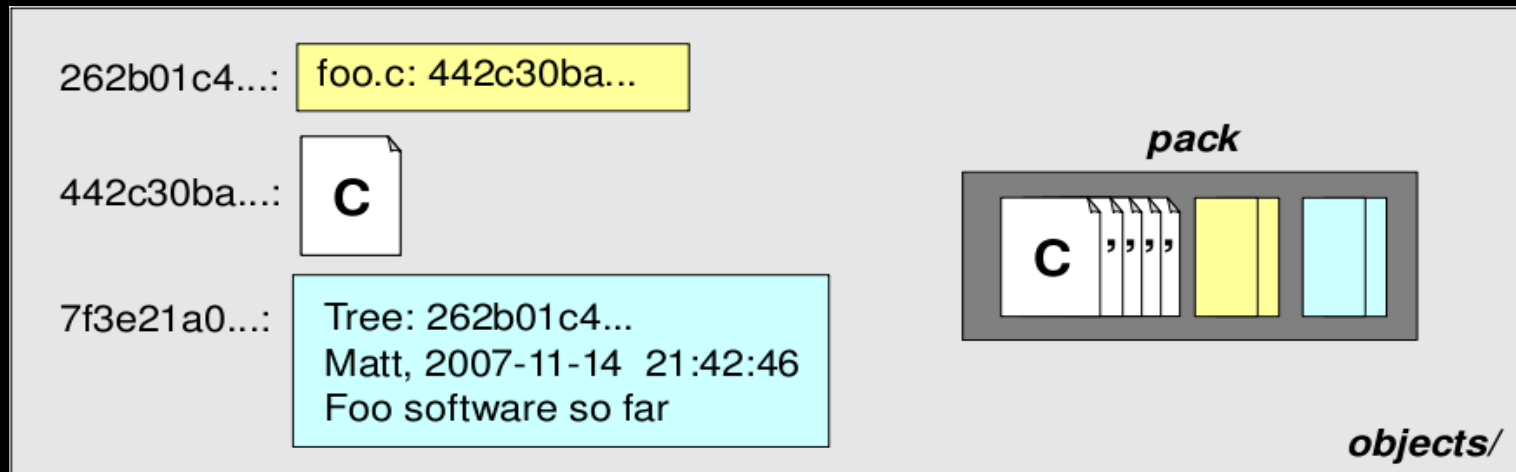
Git 目標 (2)

- 高效能、經濟的空間使用
 - Branching 與 merge 是低度衝擊的動作
 - 快速的 Diff：通常只要 1 秒！
 - KDE tree 的比較
 - Git：小於 2 Gb 的空間
 - SVN：約 8 GB 的空間



Git 目標 (3)

- 高度可靠
 - 無須顧慮檔案、記憶體毀損等議題
 - Git 追蹤整個 repository 的檔案，並非只考慮檔名
 - 採用 SHA1 hash 以確保一致性：
 - file, commit, repo... (object)



Git 目標 (4)

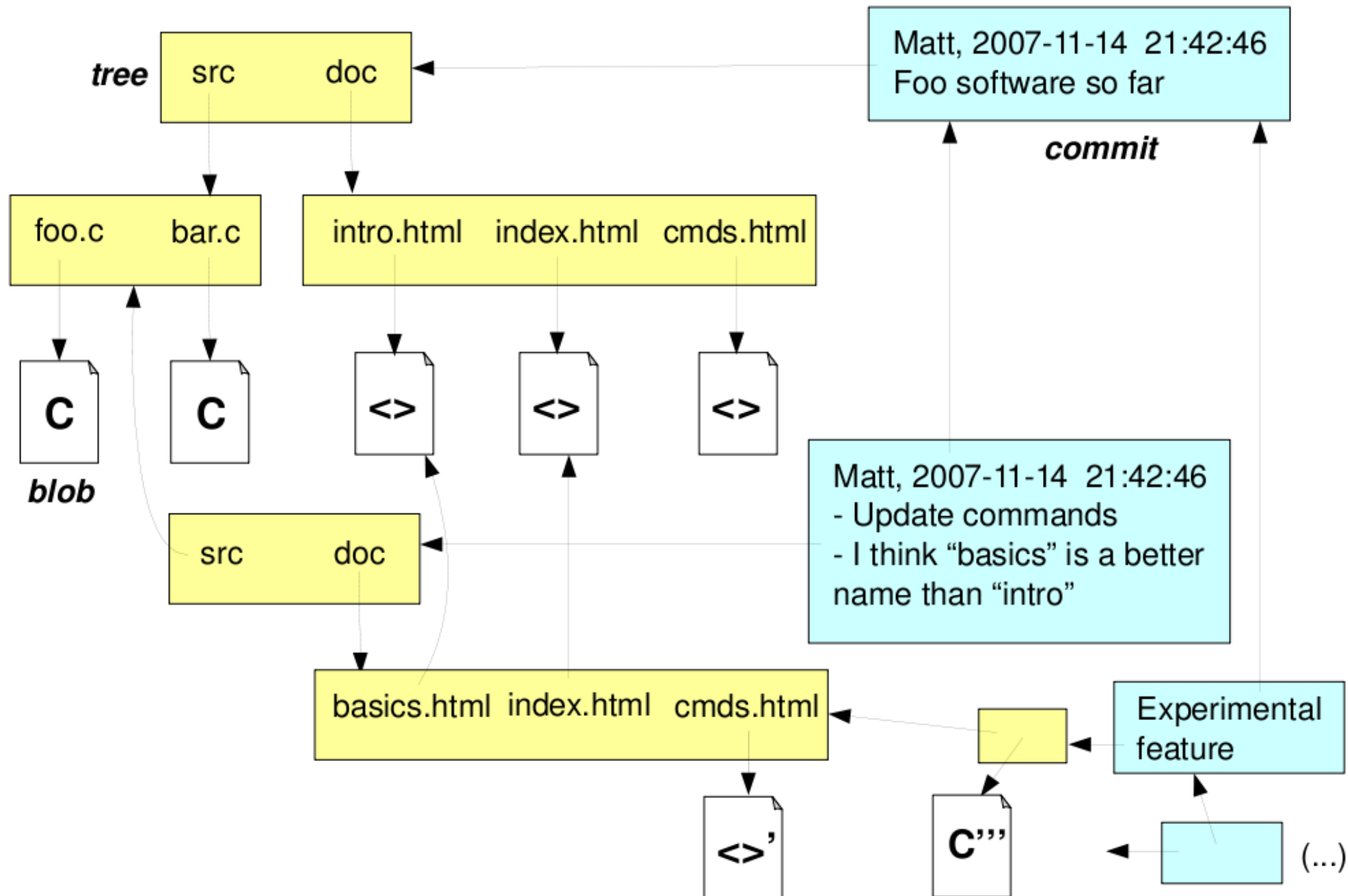
- 良好的工具整合性
 - 與其他 SCM 互通
 - 「清理」 repository – fsck, prune, gc, ...
 - 與檔案管理、郵件、數位簽章等工具整合

\$ git-

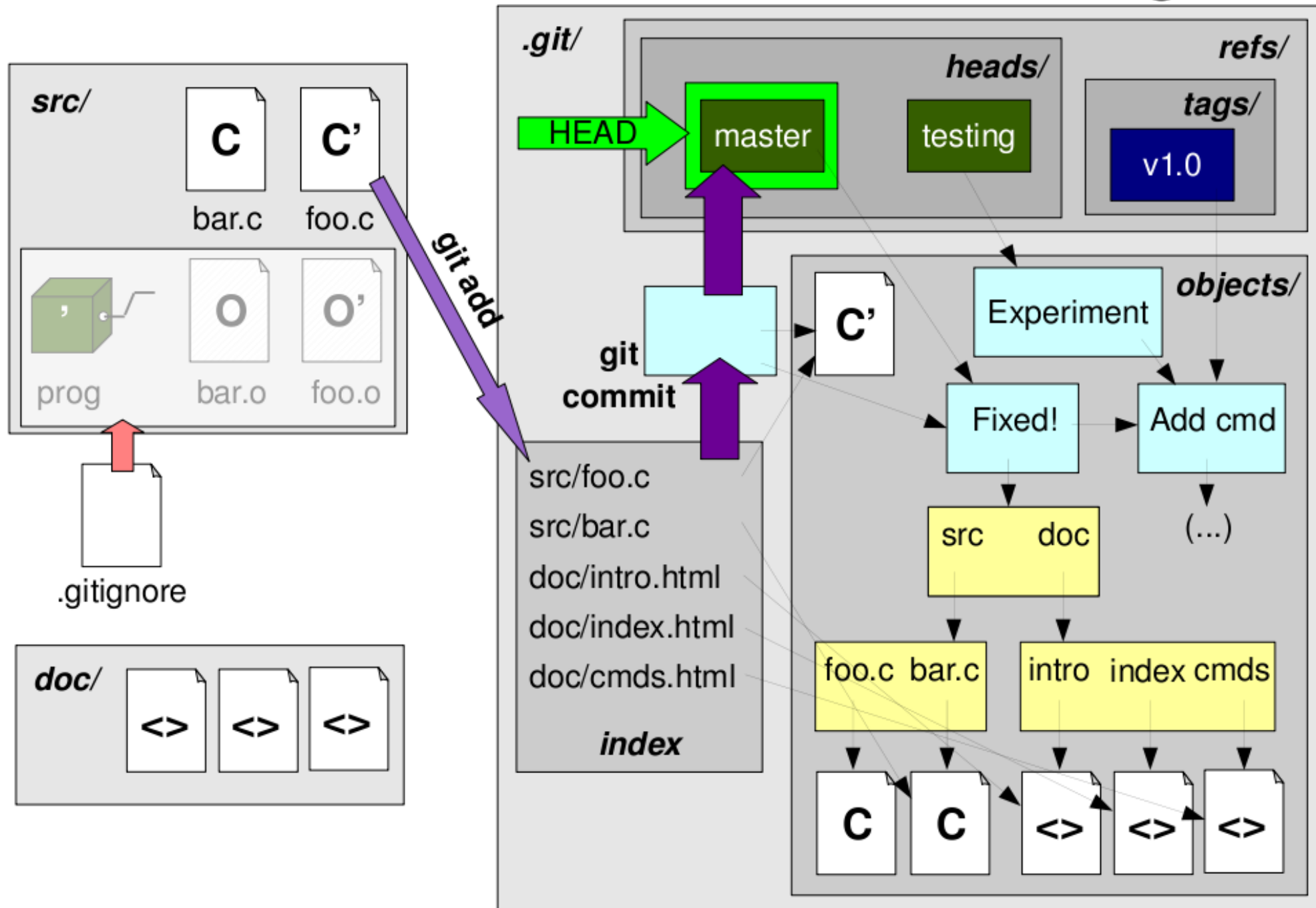
Display all 131
possibilities?



History representation: “objects”



Refs, the index, and committing



Git 相關工具

- 低階、內建
 - git-write-tree
 - Git-commit-tree
- 高階、外部
 - StGit – quilt for git
 - qgit, git gui, gitk – graphical tools
 - Gitweb
 - Cogito – cvs command like syntax (depricated)



Git 快速上手

- 設定個人資訊 (郵件、簽章整合)
 - `git config --global user.name "Jim Huang"`
 - `git config --global user.email "jserv.tw@gmail.com"`
- 建立 Repository
 - `git init`
 - `clone`
 - `git clone git://git.kernel.org/scm/git/git.git`



Git 快速上手

- HEAD: the commit that you are working on
- HEAD^, HEAD^^, HEAD~3 = HEAD^^^,
- HEAD^1, HEAD^2



Git 快速上手

- 建立 branch:
 - `git branch <name>`
 - `git branch <name> <commit-id>`
- 移除 branch
 - `git branch -d <name>`
- 顯示 branch 列表
 - `git branch`
- 採用 /Jump 到某個 commit
 - `git checkout <commit-id>`
 - `git checkout -b <name> <commit-id>`



Git 快速上手

- 給予遠端 Repository 清楚的識別 (如 min)
 - `git remote add min`
`ssh://<username>@git.kernel.org/scm/git/git.git`
- 自指定的識別抓取修改
 - `git fetch min`
- 整合 min repository 到 master branch
 - “get merge min/master”
- 或者: `git pull min`



Git 快速上手

- 顯示 commit 的 SHA1
 - Git rev-list HEAD^..HEAD
- 顯示紀錄
 - git log
 - git log HEAD~4..HEAD
 - git log --pretty=oneline v1.0..v2.0 | wc-l
 - git log --raw -r --abbrev=40 --pretty=oneline origin..HEAD
 - git archive --format=tar --prefix=project/ HEAD | gzip >latest.tar.gz
 - git blame <filename>



Git 快速上手

- 產生 commit(於 local)
 - git diff –cached
 - git add .
 - git diff HEAD
 - git commit



Git 快速上手

- Git merge

- git pull min 或 git fetch min 搭配 git merge min
- 解決衝突
 - \$ git show :1:file.txt # the file in a common ancestor of both branches
 - \$ git show :2:file.txt # the version from HEAD, but including any
 - # nonconflicting changes from MERGE_HEAD
 - \$ git show :3:file.txt # the version from MERGE_HEAD, but including any
 - # nonconflicting changes from HEAD.
- 解決 conflicts / reset 並尋求協助



Git 快速上手

- Reset the conflicted merge: use git-reset
- `git reset --mixed <commit-id>`
 - Reset the index database to the moment before merging
- `git reset --hard <commit-id>`
 - Reset the index database and the working data
- `git reset --soft <commit-id>`
 - 此命令可在不觸及工作區資料與 index 資料庫的前提下，進行錯誤修正



Source Control Solutions

- Subversion <http://subversion.tigris.org>
- CVS <http://www.nongnu.org/cvs/>
- GIT <http://git.or.cz/>
- Github <http://github.com/>
- Bazaar <http://bazaar-vcs.org/>
- Visual Source Safe

參考資料

- Comparative Development Methodologies, Dell Zhang
- Git: a modern version control system, Matt McCutchen

