

Exercise walk-through

12.4

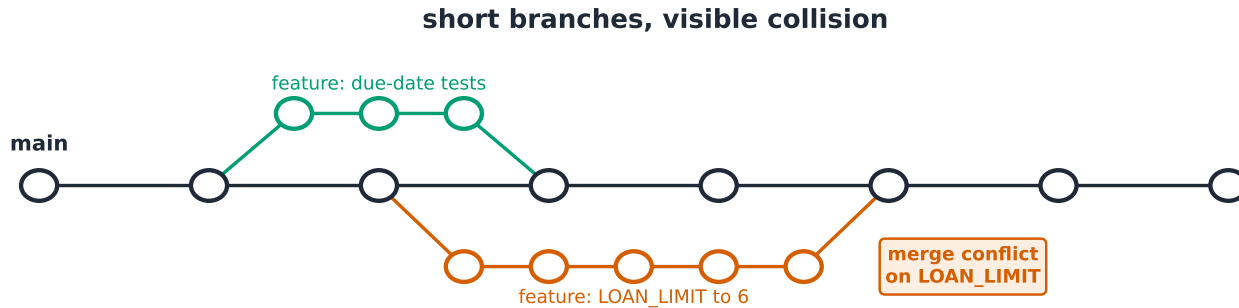
Version control and teamwork

University Computer Science

Version control: the memory that answers why

- 1 **Model** Short-lived branches, reviewed pull requests; commit dumps kill bisect - 12 runnable commits narrow a regression in at most 4 checkouts.
- 2 **Conflict** Two branches edit LOAN_LIMIT (6 vs config): the resolution is a decision recorded in the merge message.
- 3 **Review** Diff + tests + requirement link under ~200 lines - review against the acceptance test, not taste.

Branch graph



one commit = one defensible change, linked to its requirement; 12 commits bisect a regression in at most 4 checkouts

PR contract: diff + tests + requirement link; under ~200 lines

conflict resolution is a decision - record it in the merge message

review against acceptance tests, not taste

Two features: one merges clean, one collides on LOAN_LIMIT.

Exercises 2.1-2.3

2.1 What does a commit dump (“week of work”) break? [3]

2.2 Where does a conflict resolution belong? [3]

2.3 The course PR size rule is roughly what? [3]

Exercises 2.1-2.3

2.1 What does a commit dump (“week of work”) break? [3]

2.1

- Bisect – the history stops narrowing to one defensible change ✓.

2.2 Where does a conflict resolution belong? [3]

2.2

- In the merge commit’s message (and the diff itself) – the decision lives with the code ✓.

2.3 The course PR size rule is roughly what? [3]

2.3

- Under about 200 changed lines per pull request ✓.

Exercise 3.1

3.1 The repo has 16 runnable commits, all tests green at each.

State the bisect bound (number of checkouts) and justify with the logarithm. A regression appears; describe the bisect run until one commit is isolated (states of good/bad). State the two properties the history needs for bisect to work. [8]

Exercise 3.1

3.1 The repo has 16 runnable commits, all tests green at each.

State the bisect bound (number of checkouts) and justify with the logarithm. A regression appears; describe the bisect run until one commit is isolated (states of good/bad). State the two properties the history needs for bisect to work. [8]

3.1

- Bound: $\lceil \log_2 16 \rceil = 4$ checkouts ✓.
- Run: mark the last known good and the first known bad; bisect checks the mid-point commit (runs its tests), narrows good/bad halves; 4 checkouts later exactly one commit is the first bad ✓.
- Properties: every commit is runnable (tests can execute at any point) and every commit keeps the suite green when it was made – dumps or broken middles defeat the narrowing ✓.

Exercise 3.2

3.2 Two branches collide on `LOAN_LIMIT`: feature A sets it to 6 for the pilot program; feature B extracts it to a config file (value 5) so librarians can tune it.

Resolve the collision as a decision. State the merge commit's message content. Give the review contract for each PR before the merge. State the risk of resolving silently. [7]

Exercise 3.2

3.2 Two branches collide on `LOAN_LIMIT`: feature A sets it to 6 for the pilot program; feature B extracts it to a config file (value 5) so librarians can tune it.

Resolve the collision as a decision. State the merge commit's message content. Give the review contract for each PR before the merge. State the risk of resolving silently. [7]

3.2

- Decision: adopt the pilot value 6 via B's config mechanism – the rule stays tunable and the pilot is a config entry, not a code fork ✓.
- Message: names the collision, the decision (6 through config), and the reviewers of the call ✓.
- Contract per PR: diff + tests + requirement link, under ~200 lines, review against the acceptance test ✓.
- Silent resolution risk: the decision is lost – the next reader sees one value with no record of why or who chose it ✓.