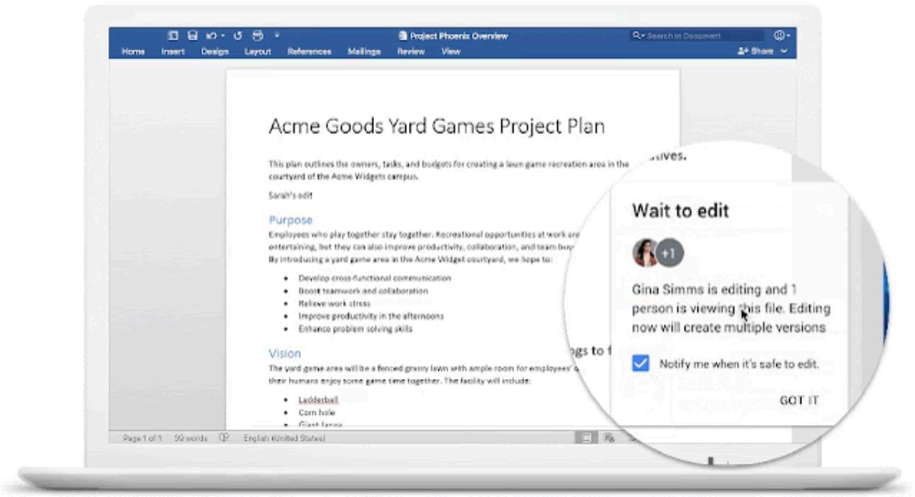
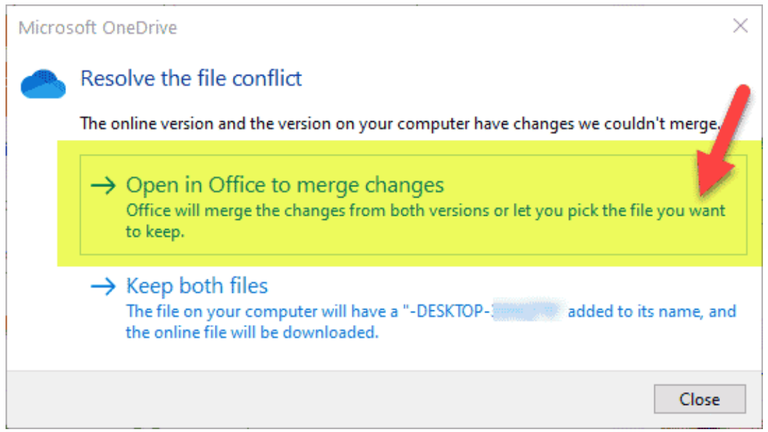
CONCURRENCY CONTROL AND AWARENESS SUPPORT FOR MULTI-SYNCHRONOUS COLLABORATIVE EDITING

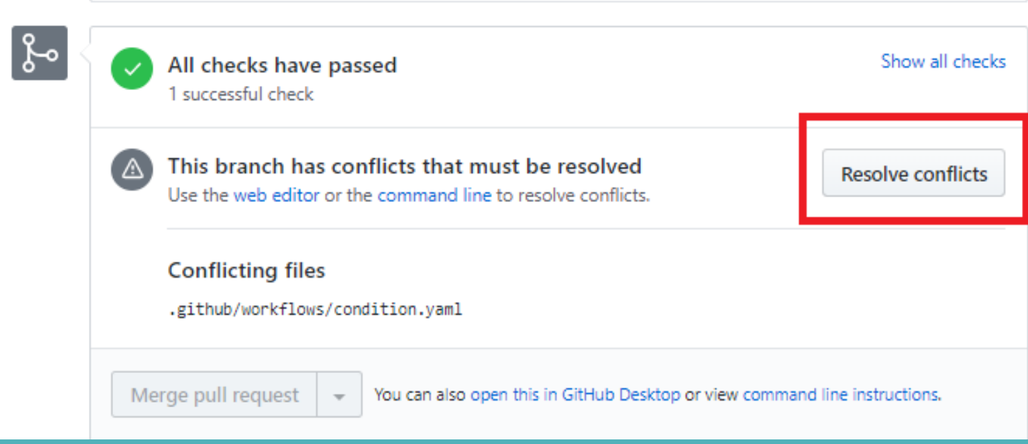
Created by Ana-Maria Cristina Hognogi

WHY COLLABORATIVE EDITING BREAKS



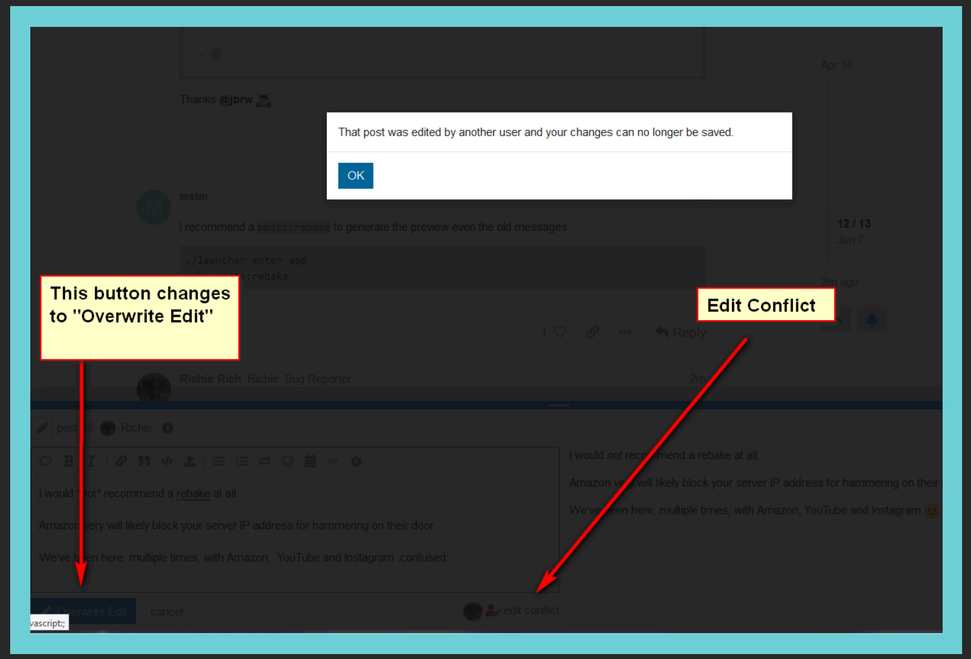
Most editors, like Google Docs, try to support both real-time and offline editing. But when users reconnect, the merge can be messy. Text disappears, changes mix in strange ways, and sometimes the cursor jumps unpredictably. These tools were never built for multi-synchronous editing, so issues are common.

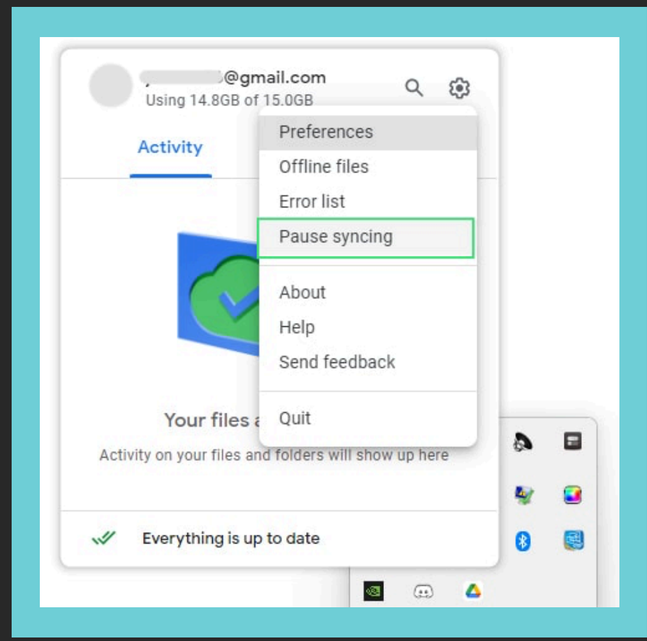


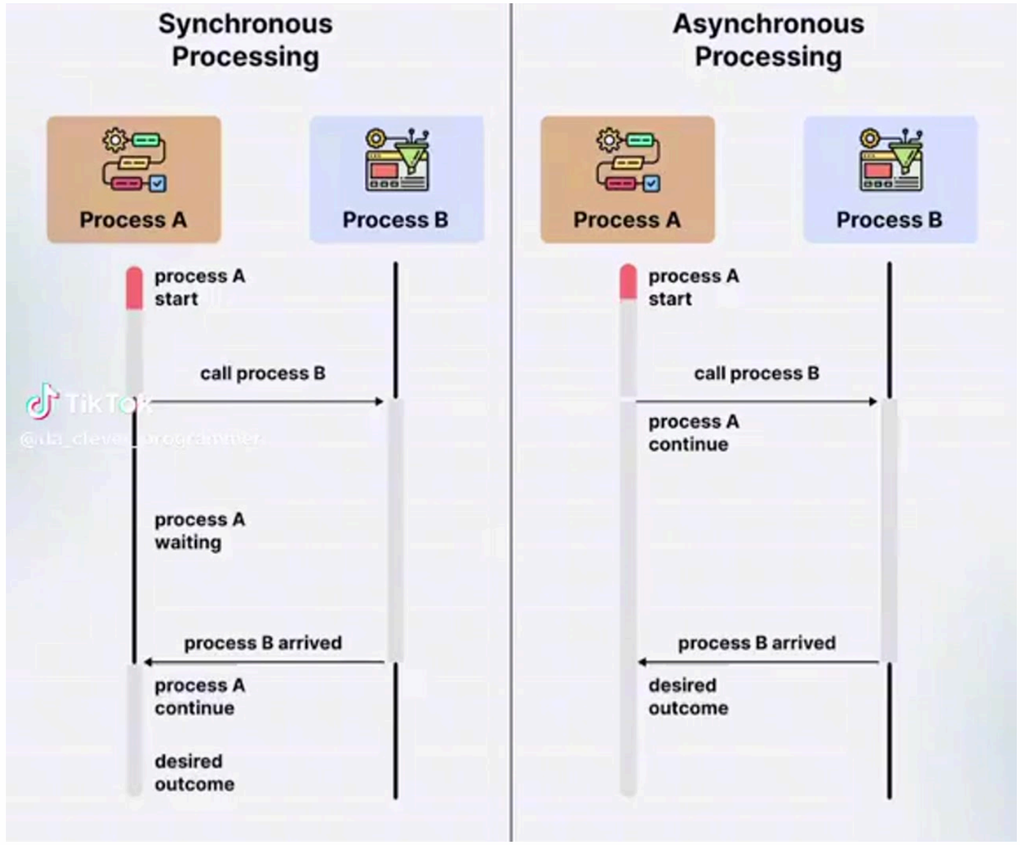


REAL MERGE PROBLEMS

Several merge problems are: duplicated fixes, overwritten text, incorrect revision histories, and even disappearing lines. Most of this happens because the algorithms used today were meant for real-time editing, not disconnected work.





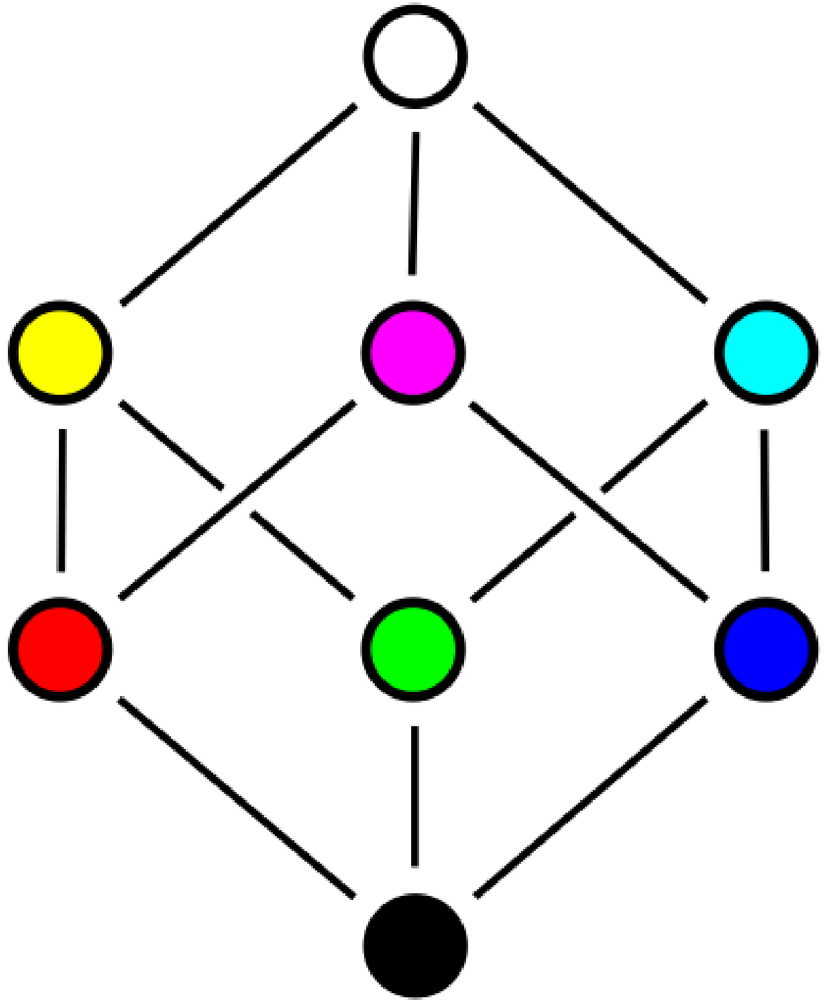


SOLUTION WANTED

The solution wanted is to handle both synchronous and asynchronous editing, prevents hidden data loss, and ensures that merges keep the meaning and intention behind edits. At the same time, improve awareness so users understand what actually happened after a merge.

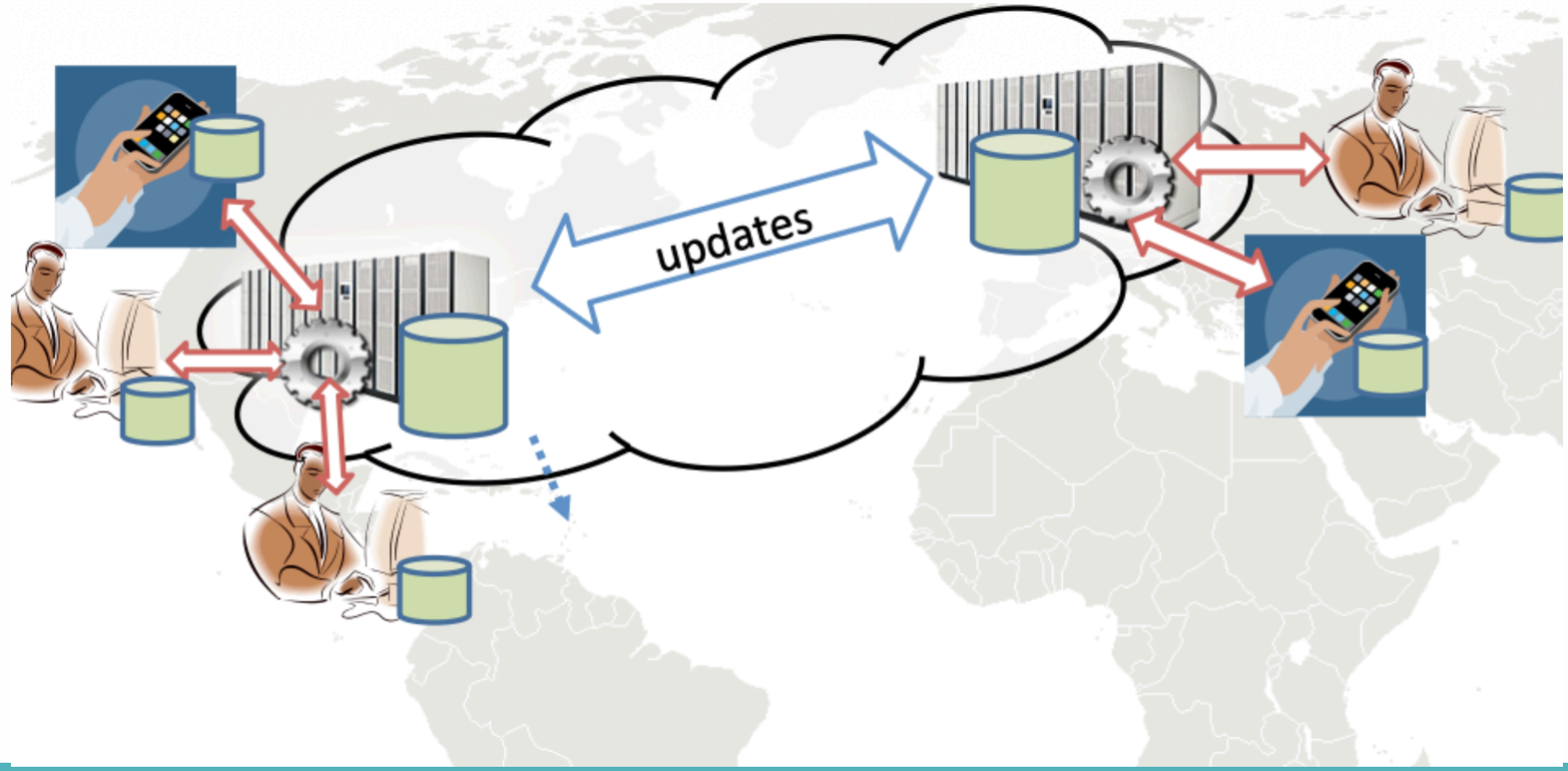
WHY CRDTS ARE THE FOUNDATION

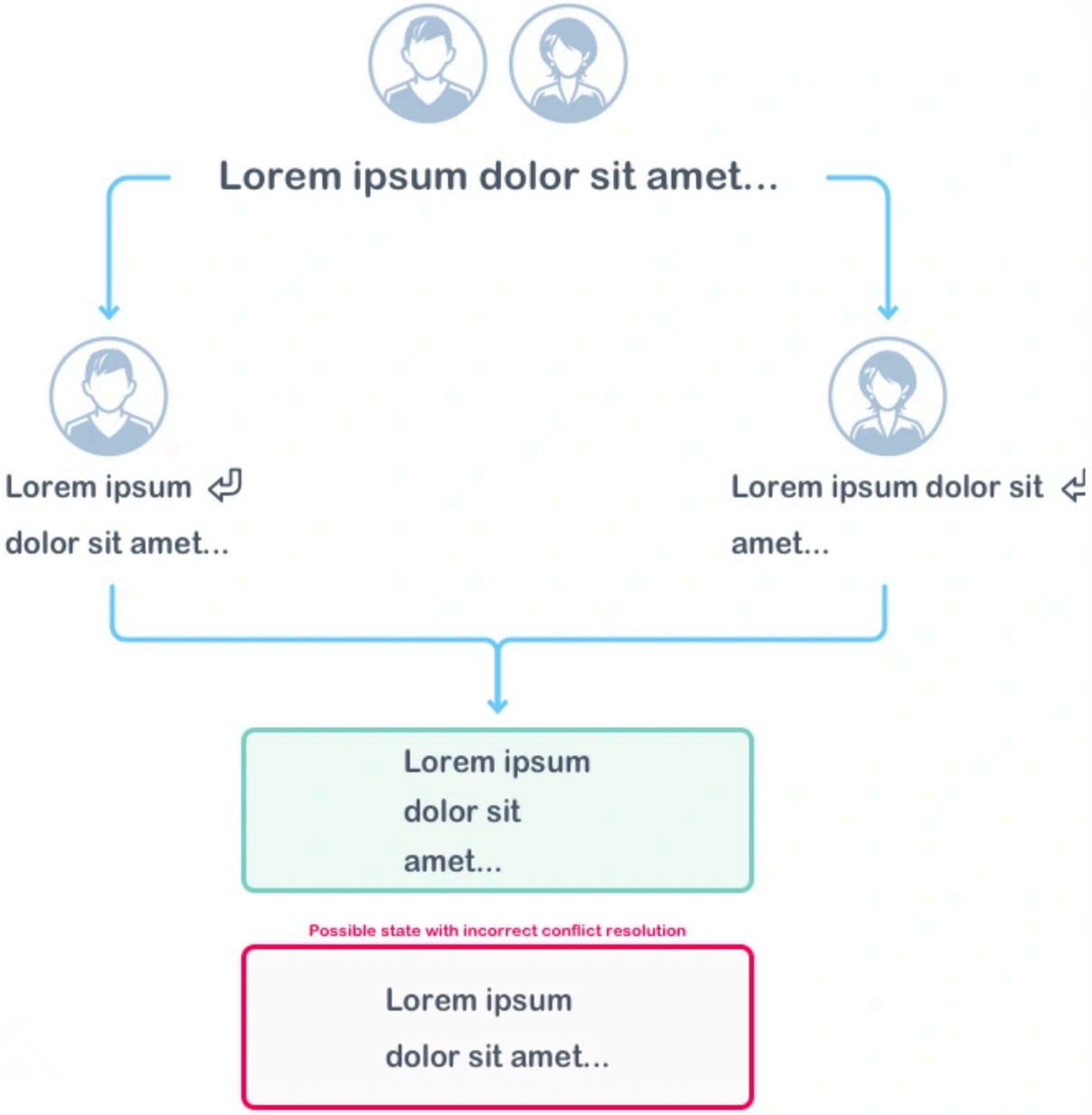
A Conflict-free Replicated Data Type (CRDT) is a data type designed so that concurrent operations commute and all replicas converge to the same state if all operations are delivered in causal order. This avoids complex transformation logic and works well in distributed settings.



NEW OPERATIONS: UPDATE & MOVE

Most editors only support insert and delete at the character level. So a new higher-level operations are introduced: update and move. These operations help preserve meaning when merging offline edits. For example, if one user updates a paragraph while another moves it, the system moves the updated version rather than losing edits.





AWARENESS: HELPING THE USER UNDERSTAND

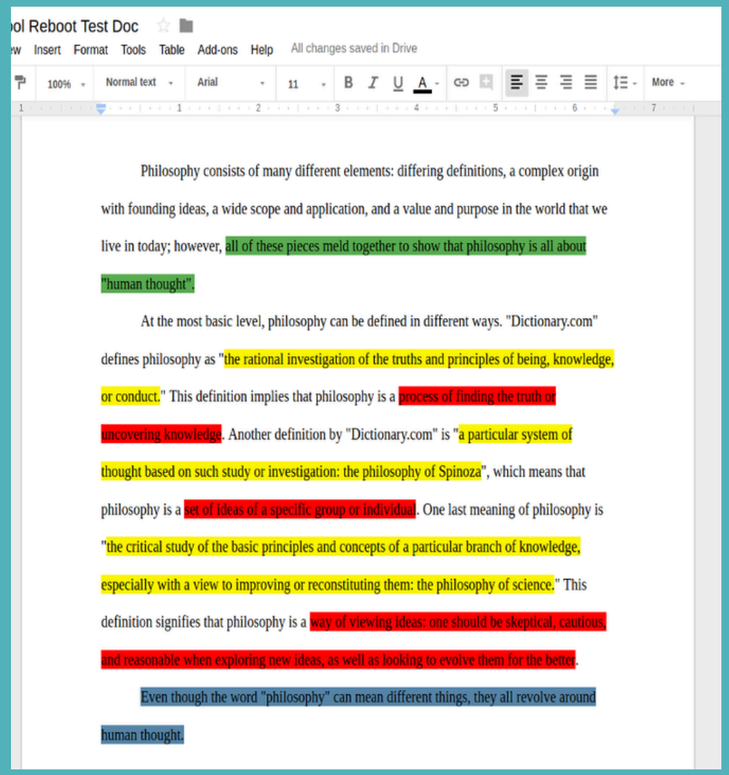
The system doesn't hide conflicts. It highlights updates, duplicates, and alternative versions so the user can decide what to keep. This is a huge improvement over silent merges where users don't know what was overwritten.

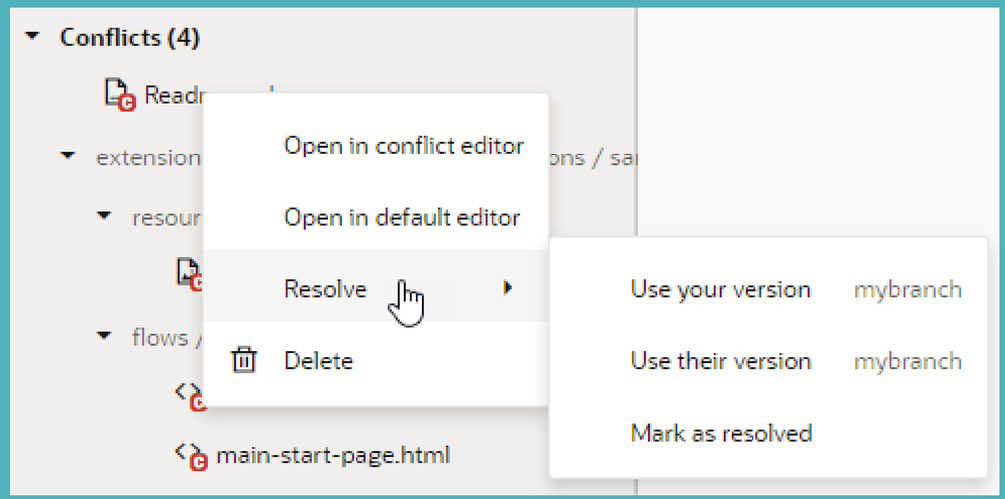
POS: a set of pairs (elementId, positionId).
An element can appear in multiple positions (clones) and thus have several pairs in POS.

VAL: a set of pairs of the form ((elementId, timestamp), value).
Each element can have multiple versions (e.g. from concurrent updates).
Timestamps allow selecting a default version and detecting concurrent versions.

CONFLICT RESOLUTION POLICY

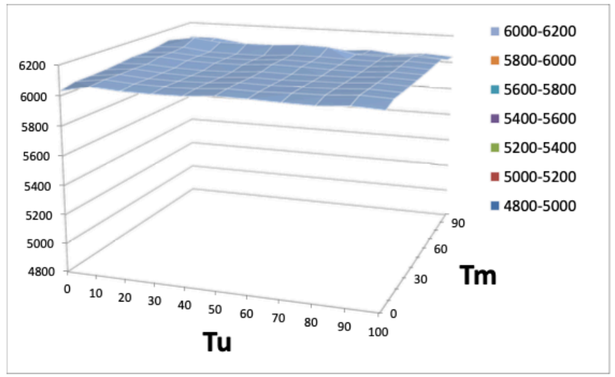
1. Insert vs Insert (Same Position)
   - system keeps both elements
   - two consecutive paragraphs
   - highlight paragraphs
2. Update vs Update
   - multiple versions in VAL
   - one version becomes the default
   - highlight the element
3. Update vs Delete
   - the element is deleted.
   - updated version is kept in VAL
   - awareness elements through a conflict panel
4. Move vs Delete
   - the element disappears from the visible document
5. Update vs Move
   - moved version is the updated one
   - highlight that this element was moved and updated concurrently
6. Move vs Move
   - two positions (clones)
   - POS now has multiple position entries



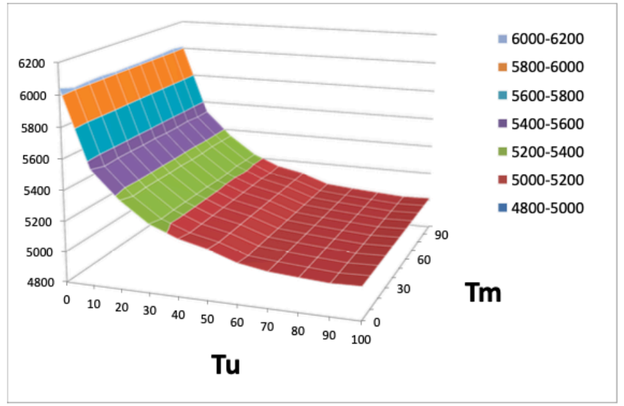


EVALUATION ON REAL GIT REPOSITORIES

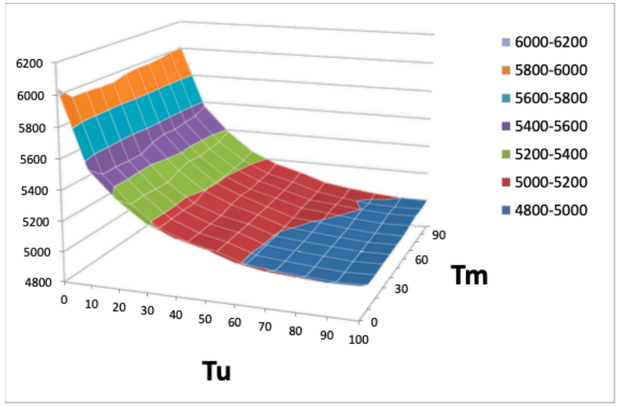
Testing the approach using real Git project histories, replacing edits with CRDT approach and comparing the results against actual developer-made merges. The model produced merged documents that were closer to the real, intended results, about 18% better than the baseline algorithm.



(a) TreeDoc

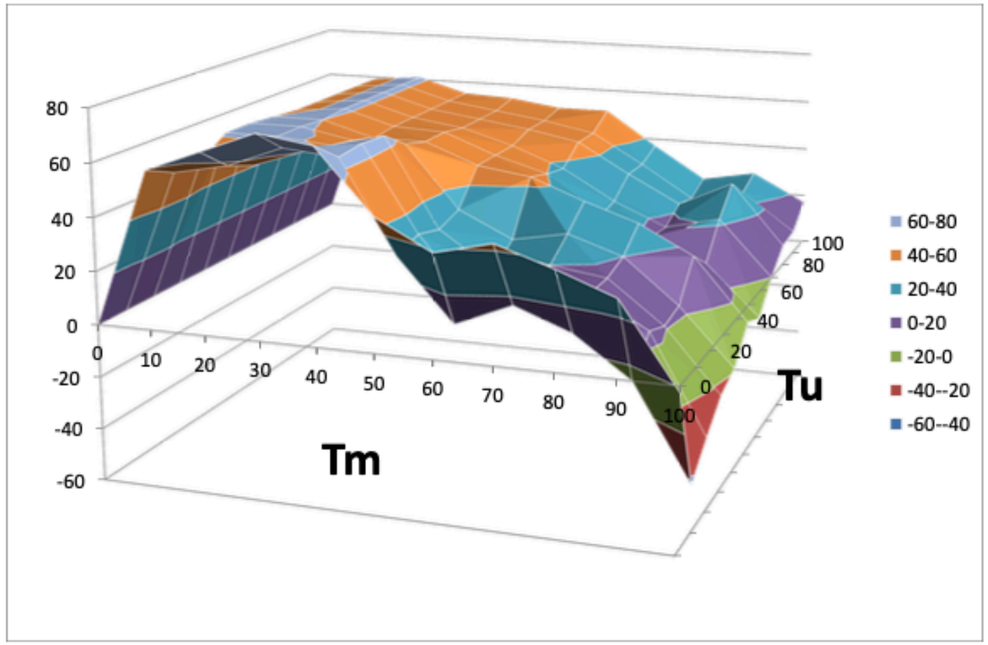


(b) Update only



(c) Update and move

DETECTING UPDATE AND MOVE OPERATIONS



Update detection

For lines in a replace block:

- Compute a normalized Levenshtein distance between old and new lines.
- If this distance is below threshold Tu, treat the change as an update (the same line, modified) instead of a pure delete + insert.

Move detection

After handling updates, look for remaining lines that were deleted in one place and inserted elsewhere:

- Search for matching sequences of lines of length at least 2.
- If similarity is above a threshold Tm, treat the change as a move.
- This avoids counting a move as independent delete and insert operations.
- Vary Tu and Tm between 0 and 1 to find good values.

CONCLUSION

In conclusion, the proposed system improves both merge quality and user understanding. It avoids silent data loss and supports both online and offline editors consistently. This approach is promising for next-generation collaborative tools.

THANK YOU