SYNCHRONOUS COLLABORATION

Gerald Weber

Collaborative Work

- **Synchronous collaboration**
  - WYSIWIS, screen sharing
  - Operational Transformation
  - History rewriting

Synchronous collaboration: Motivations

- □ Team members work on different locations.
- □ The team wants fast joint development of a document.
- □ There is steady process, so there is no particular focus on access to earlier versions. Fits well to text documents.
- □ Explicit commit of new versions would be too heavyweight. Collaborators make frequent small changes: shared spreadsheet.

Application Sharing

- A collaborative approach on a low technology layer ('low' doesn't mean 'bad') .
- WYSIWIS: What you see is what I see.
- MS Netmeeting.
- Shares a single application such as an office application.
- Begole, J., Rosson, M. B., and Shaffer, C. A. 1999. Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems. *ACM Trans. Comput.-Hum. Interact.* 6, 2 (Jun. 1999), 95-132. DOI= http://doi.acm.org.ezproxy.auckland.ac.nz/10.1145/319091.319096

Screen Sharing

- Also a WYSIWIS technology.
- can be based on desktop/windowing framework.
- Several persons see a single desktop.
- Single input screen sharing:
- There is still a single mouse cursor and a single text cursor:
- Easy to implement: Compatible with all applications.

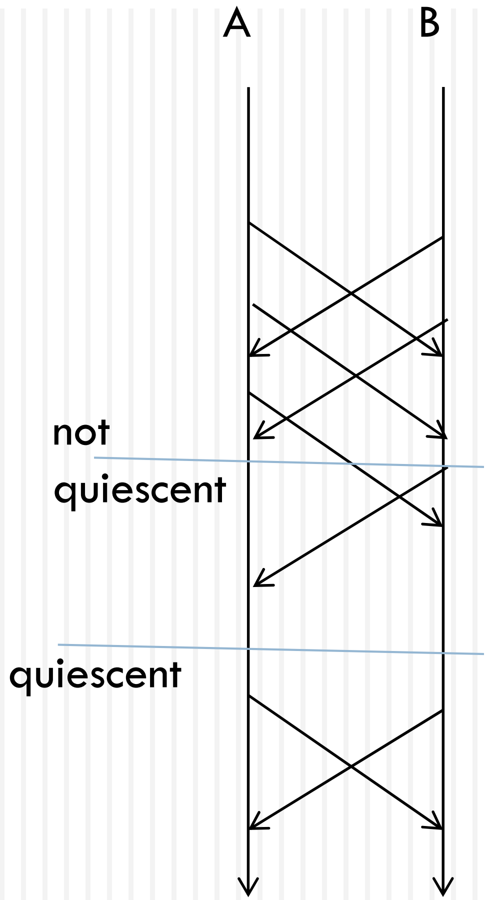
Operational Transformation

- A theory for building synchronous collaborative applications.
- Operations of collaborators are broadcast to other collaborators. At the remote locations, operations might have to be executed in slightly different form.

- Ellis, C. A. and Gibbs, S. J. 1989. Concurrency control in groupware systems. *SIGMOD Rec.* 18, 2 (Jun. 1989), 399-407. DOI=http://doi.acm.org.ezproxy.auckland.ac.nz/10.1145/66926.66963

Operational Transformation Terminology:

- □ A groupware system is *quiescent* if all operations have been executed at all sites.
- □ Convergence property for groupware systems: The state of the artifact should be the same at all sites at quiescence.



Operational Transformation model

□ Classical text editor:

□ Text is modeled as String, the characters are numbered with running numbers.

□ Operations have character positions as parameters. (Caveat: This community starts with1, not with 0!);

□ delete(2) : delete character at position 2.

□ Insert(3,’b’): insert a ‘b’ before the character at position 3.

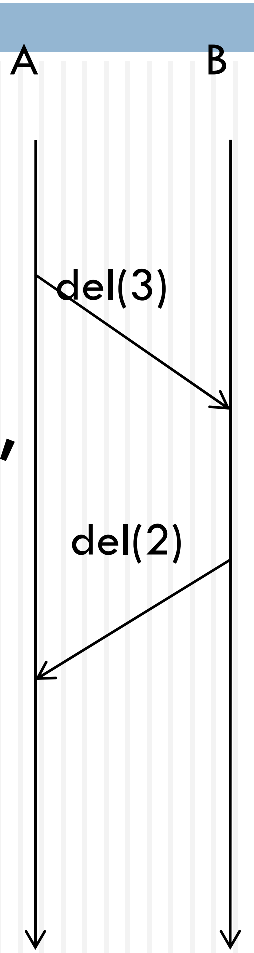
□ “abcd”.delete(2).insert(3,‘g’)= ?

Operational Transformation model

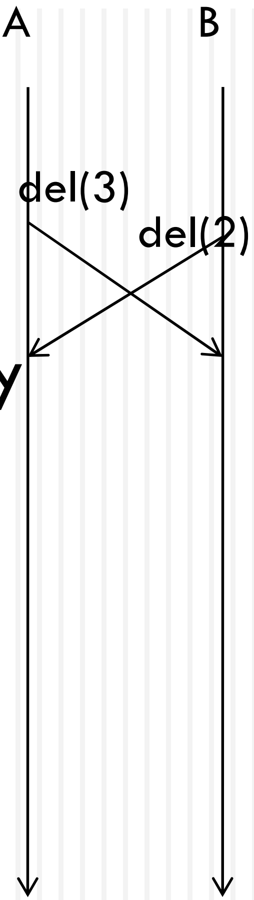
- Single-character operations suffice because the focus is on fast synchronization: every keystroke is immediately looked at by the synchronization framework.
- Example demonstrating the problem that operational transform is addressing:
- Site A executes delete(3) and site B executes delete(2)

Distributed messaging of operations

- □ Precedence property
- □ If operations don't overlap, then they should be executed in the resulting order



- □ But Operations can overlap!
- □ In case they are executed at every site in a different order:
- □ convergence property might be violated.



Operational Transformation problem

- Consider the following example problem:
- Initial state is “abcd”.
- Site A executes delete(3) the other site B executes delete(2). Then they send the operation to the other site.
- A: “abcd”.delete(3).delete(2) = “ad”
- B: “abcd”.delete(2).delete(3) = “ac”
- Convergence property would be violated

Operational transformation approach:

□ The sites exchange enough information so that A can see that B has not executed A's op (delete(3)) before executing B's op (delete(2)) and vice versa.

□ Can A simply apply B's op? *yes*

□ Can B simply apply A's op? *no*

Solution for B

- B cannot simply apply A’s delete(3).
- The outcome would violate the convergence property.
- It is B which would be giving an incorrect result, because “acd”.delete(3) violates the intention of A’s delete(3), namely to delete “c”.
- Solution: delete(3) is transformed at B into delete(2).

Transformation matrix

- To solve the problem, operations have to be applied to other operations.
- Operational transformation uses a transformation matrix.
- Each entry in the matrix tells how one operation o1 must be transformed by another operation o2.

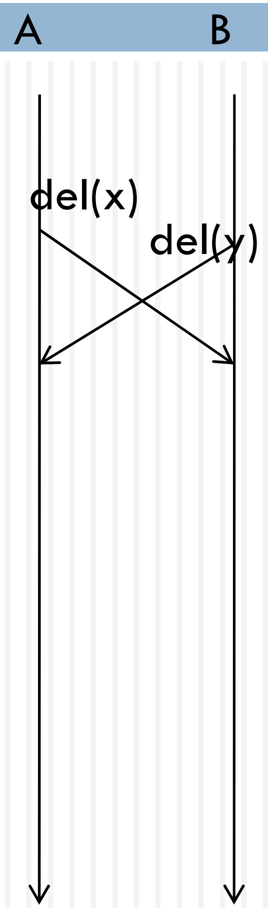
	del()	ins()
del()	m1	m2
ins()	m3	m4

Example transformation

- Each side detects that operations have overlapped.
- For that purpose sufficient auxiliary information must be transferred.
- Each side applies the transform, but at site A this will result in an unchanged operation.
- Hence the operation will have a conditional outcome.

Example transformation (m1 in the matrix)

- Transform at B
  - for del(x) coming from A
  - overlapping with del(y) at B
- if x<y del(x) -> *del(x)*
- if x=y del(x) -> *no operation*
- if x>y del(x) -> *del(x-1)*

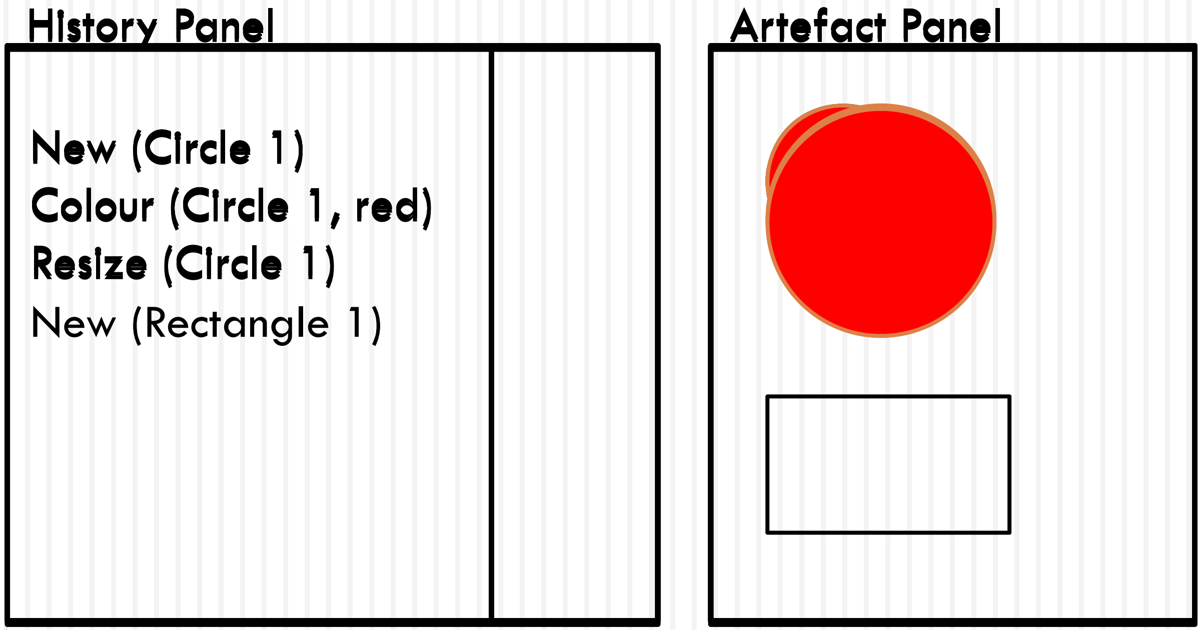


History-Based Editor

- An alternative to operational transform.
- The model-based editors some of you are developing.
- Main difference to OT for text: In our editors operations have object identities as parameters.
- They don't change if other operations are applied to other objects.
- Carlo Bueno, Sarah Crossland, Christof Lutteroth and Gerald Weber. *Rewriting History: More Power to Creative People, OZCHI 2011*

Writing History

- Editors record the *history* of *user operation applications* (called *operations* for short)
- **Operation:** An action of the user, e.g. creating a new shape
- **History:** A list of operations



New (Circle 1)
Colour (Circle 1, red)
Resize (Circle 1)
New (Rectangle 1)

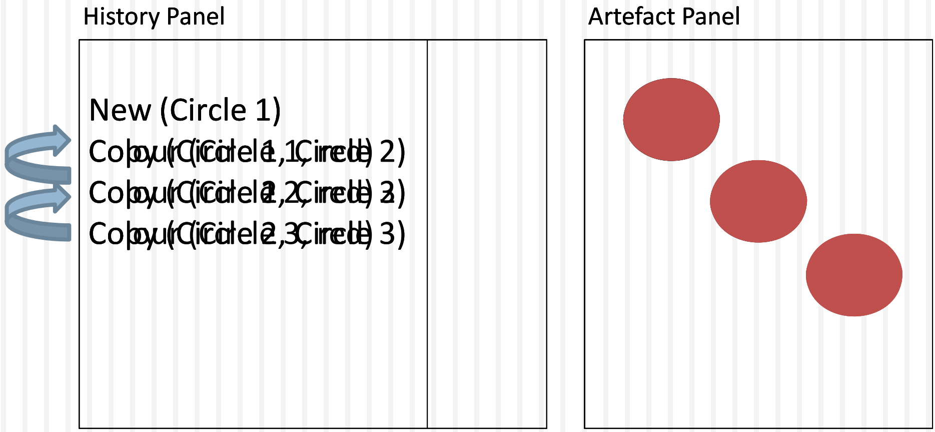
History Operations: Generalizing and Specializing

Generalising

Apply an operation to a superset of shapes

Specialising

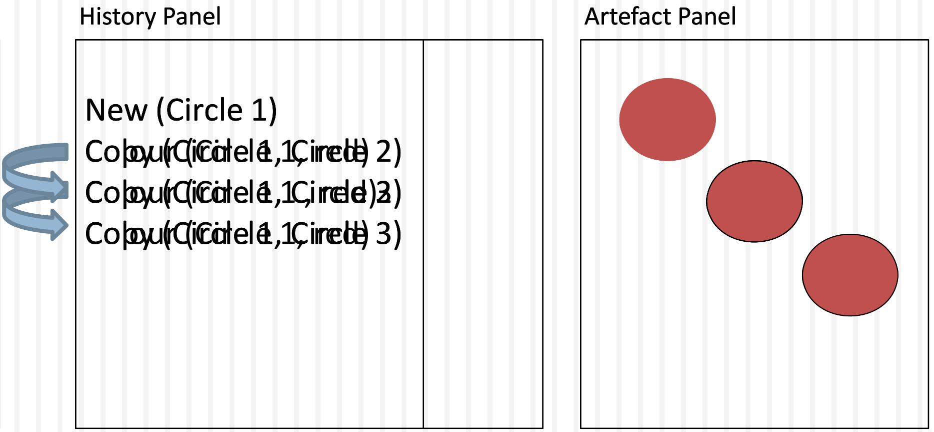
Apply an operation to a subset of shapes



History Panel

New (Circle 1)
Copy(Circle 1, Circle 2)
Colour(Circle 1, red)
Copy(Circle 2, Circle 3)
Colour(Circle 2, red)
Copy(Circle 2, Circle 3)
Colour(Circle 3, red)

Artefact Panel



History Panel

New (Circle 1)
Copy(Circle 1, Circle 2)
Colour(Circle 1, red)
Copy(Circle 1, Circle 3)
Colour(Circle 1, red)
Copy(Circle 1, Circle 3)
Colour(Circle 1, red)

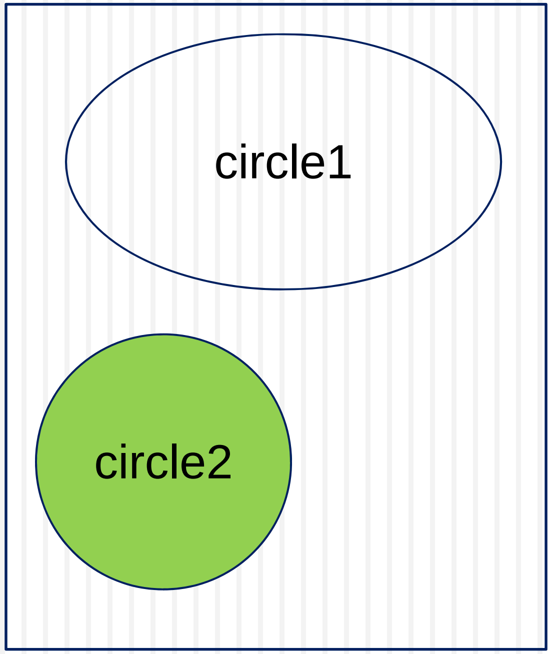
Artefact Panel

History Operations: Deleting, Merging

History pane

copy(CIRCLE, circle1)	Bob
copy(circle1, circle2)	Ann
color(circle2, green)	Ann
stretch(circle1, 1.7)	Bob

Artifact pane

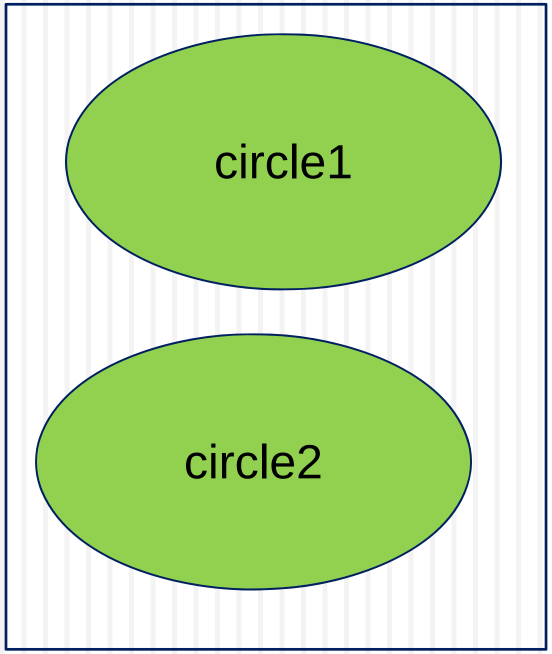


History Operations: Deleting, Merging

History pane

newCircle(circle1)	Bob
color(circle1, green)	Ann
stretch(circle1, 1.7)	Bob
copy(circle1, circle2)	Ann

Artifact pane



Commutativity 1

Two operations a and b are commutative if their order of execution does not change the resulting diagram: xaby = xbay

Shape Disjointness

- □ If two operations do not refer to the same shapes, we call them *shape disjoint*
- □ If two operations are shape disjoint, then they are commutative

Type Disjointness

- □ If two operations have different types, they are *type disjoint*
- □ In our tool (except for copy): if two operations are type disjoint then they are commutative
- □ …because operations with different types affect independent shape properties

Difference to OT

- OT uses a model, where most operations are not commutative.
- Even if team members work on different parts of the document, operations semantically influence each other:
- The data model of the artifact is partly responsible for the problem.
- History-based editors uses datamodels where many operations are commutative.
- Why are more operations commutative?

Difference in data models

- □ OT uses a model, where objects are addressed with changeable identifiers.
- □ Users mean to delete a certain character, but delete operation is encoded by position.
- □ Users obviously give identity to characters.
- □ Position is affected by other operations.
- □ History-based editors give objects immutable identities.

Commutativity 2

Commutativity of Operations Is Not Transitive

- Def. Transitivity: If A is commutative with B and B is commutative with C, then A is also commutative with C
- Counter example:
  - color(circle1, red) and move(cirlce2, pos1) are commutative
  - move(cirlce2, pos1) and color(circle1, blue) are commutative
  - But: color(circle1, red) and color(circle1, blue) are not

Commutative Neighborhood of an Operation A

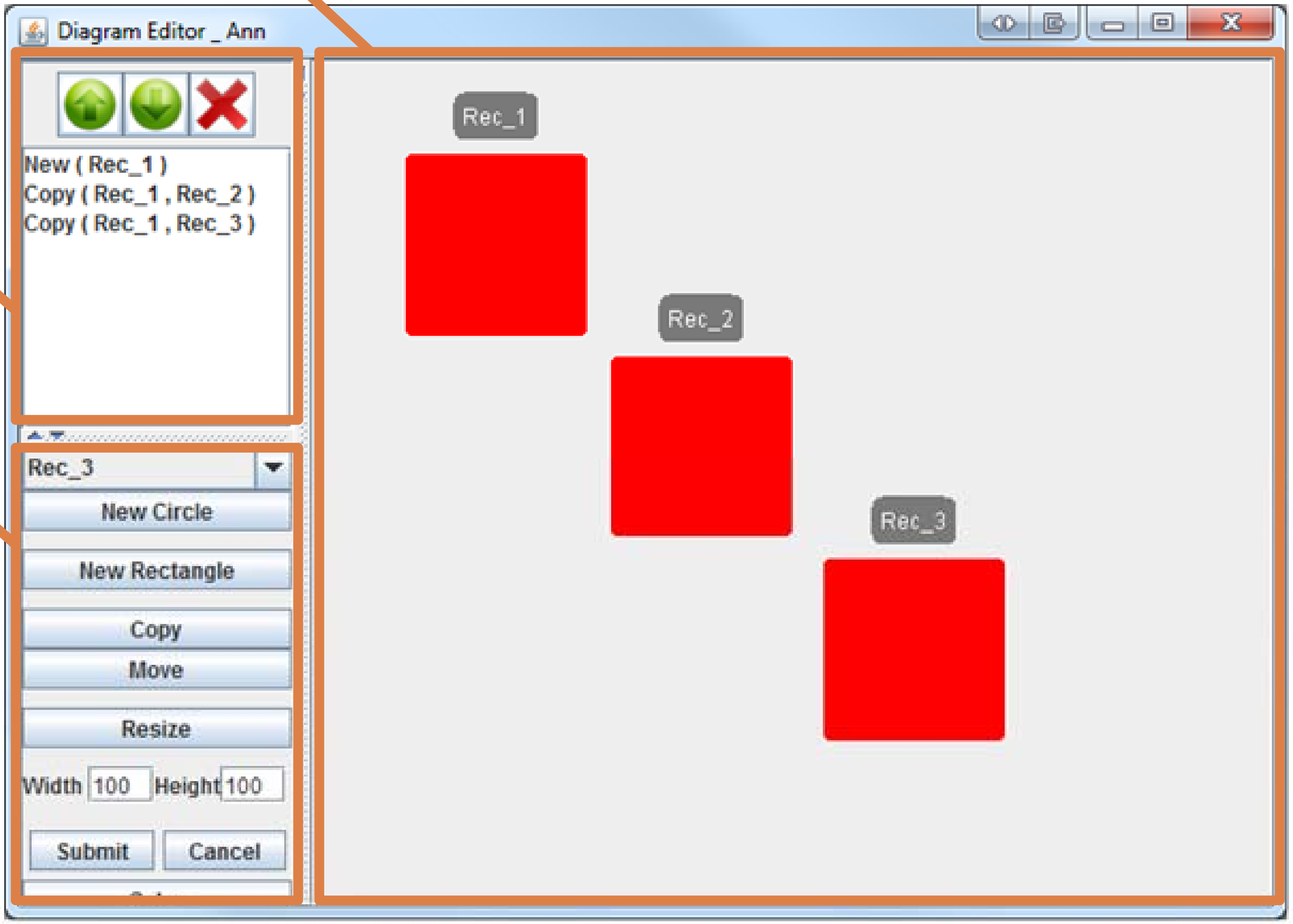
- Neighboring operations that are commutative to A
- Application: swap operation to the next position where it will produce a change in the diagram

Prototype

Artefact Panel

History Panel

Operation Panel



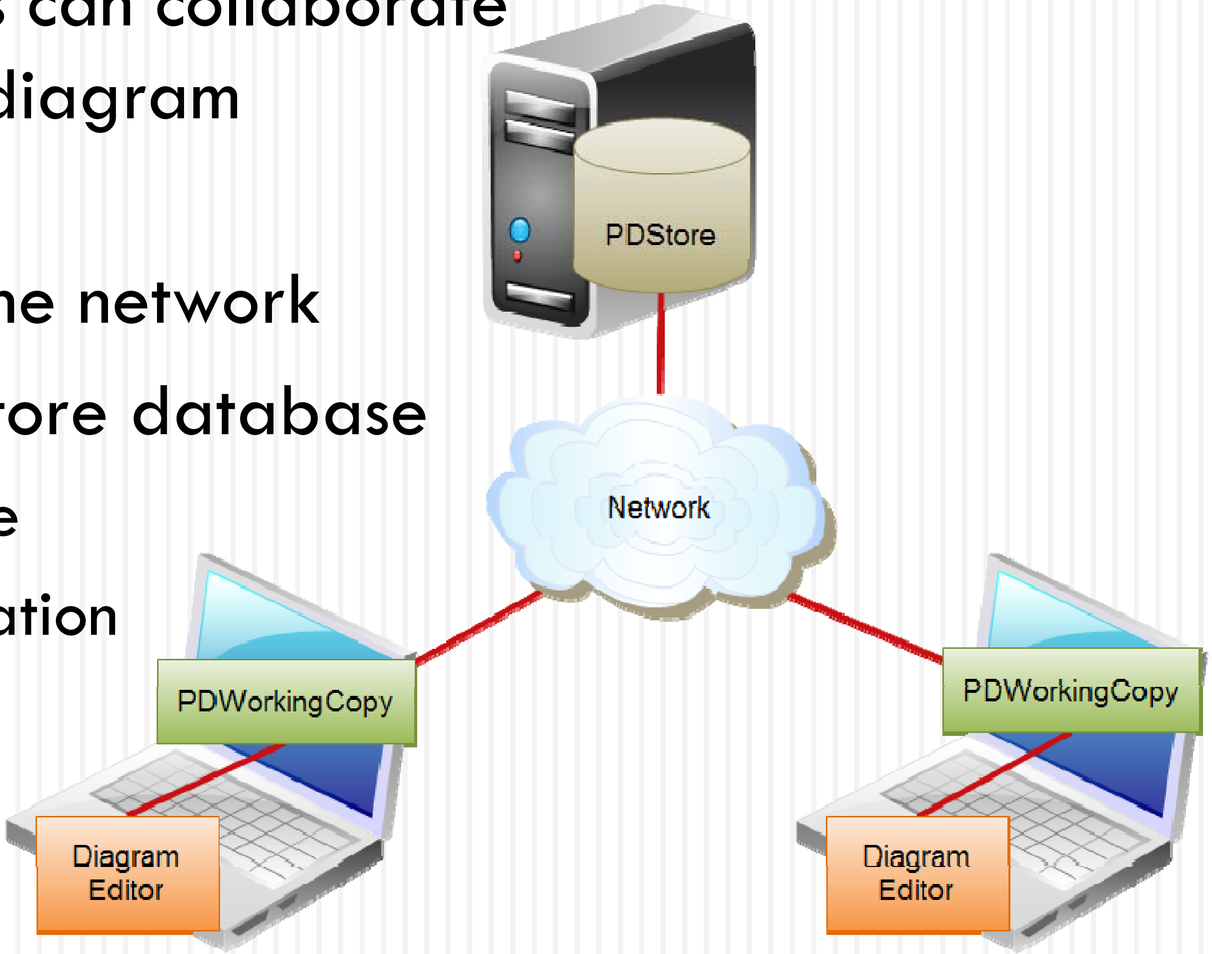
New (Rec_1)
Copy (Rec_1 , Rec_2)
Copy (Rec_1 , Rec_3)

Rec_3

Width 100 Height 100

Prototype Design

- □ Multiple users can collaborate on the same diagram in real-time
- □ Works over the network
- □ Uses the PDStore database
  - Data storage
  - Event notification



User Study

Research Questions

1) Is history rewriting easy to understand?
2) Do users have a preference for history rewriting?

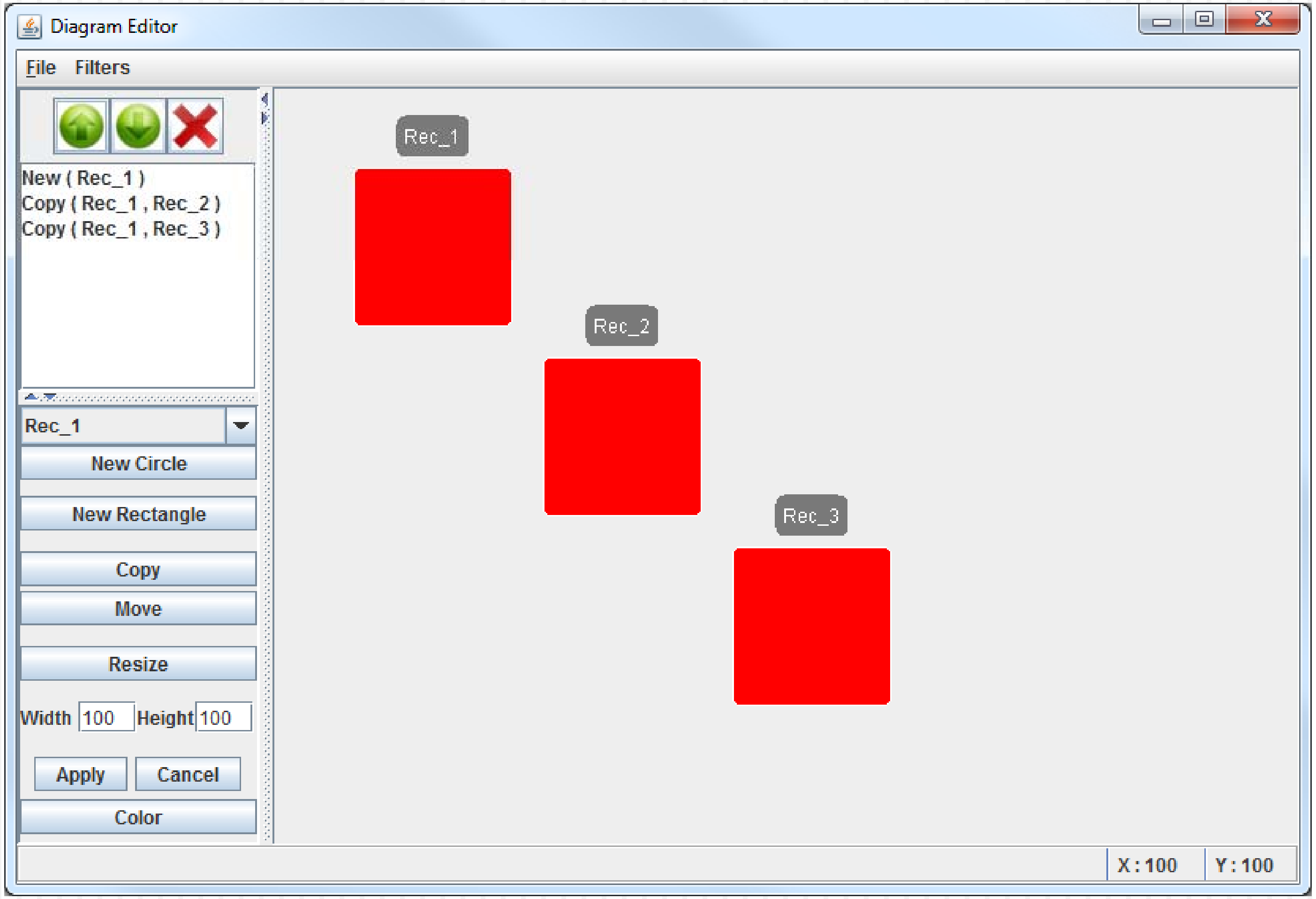
Study Design

1) Short tutorial with prototype
2) 3x “how to” questions to see if history would be used
3) 7x “what if” questions to see if history is understood
4) 2x 5-point-Likert-scale preference questions
5) 4x open questions about preference and suggestions

11 participants – primarily 4th year SoftEng students

Example 1: Will Users Prefer Generalization over Repetition?

3) Refer to Figure 3. How would you resize all three rectangles to have a width of 250 and a height of 70?

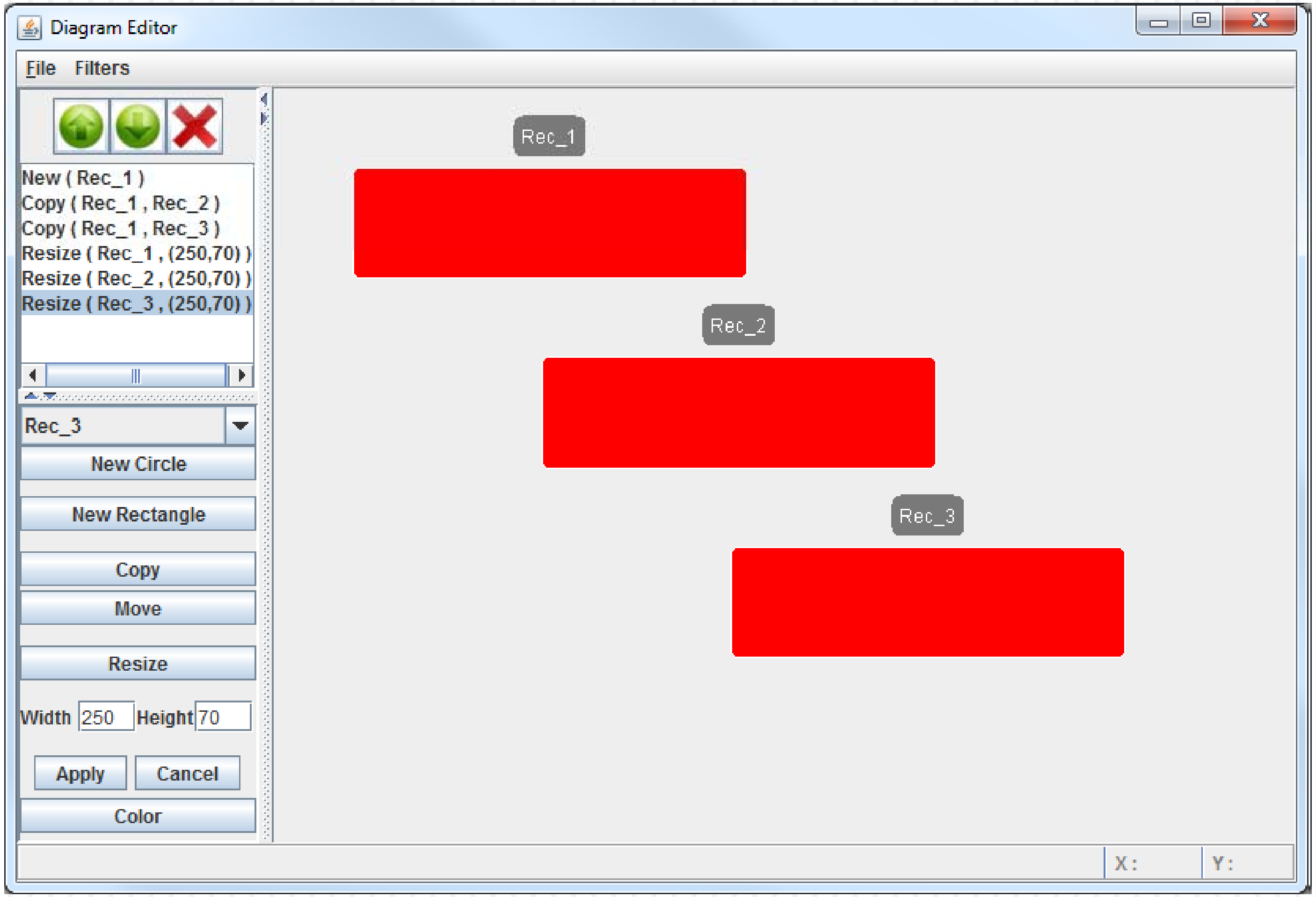


New (Rec_1)
Copy (Rec_1 , Rec_2)
Copy (Rec_1 , Rec_3)

Width 100 Height 100

Example 1: Solution with Repetition

3) Refer to Figure 3. How would you resize all three rectangles to have a width of 250 and a height of 70?



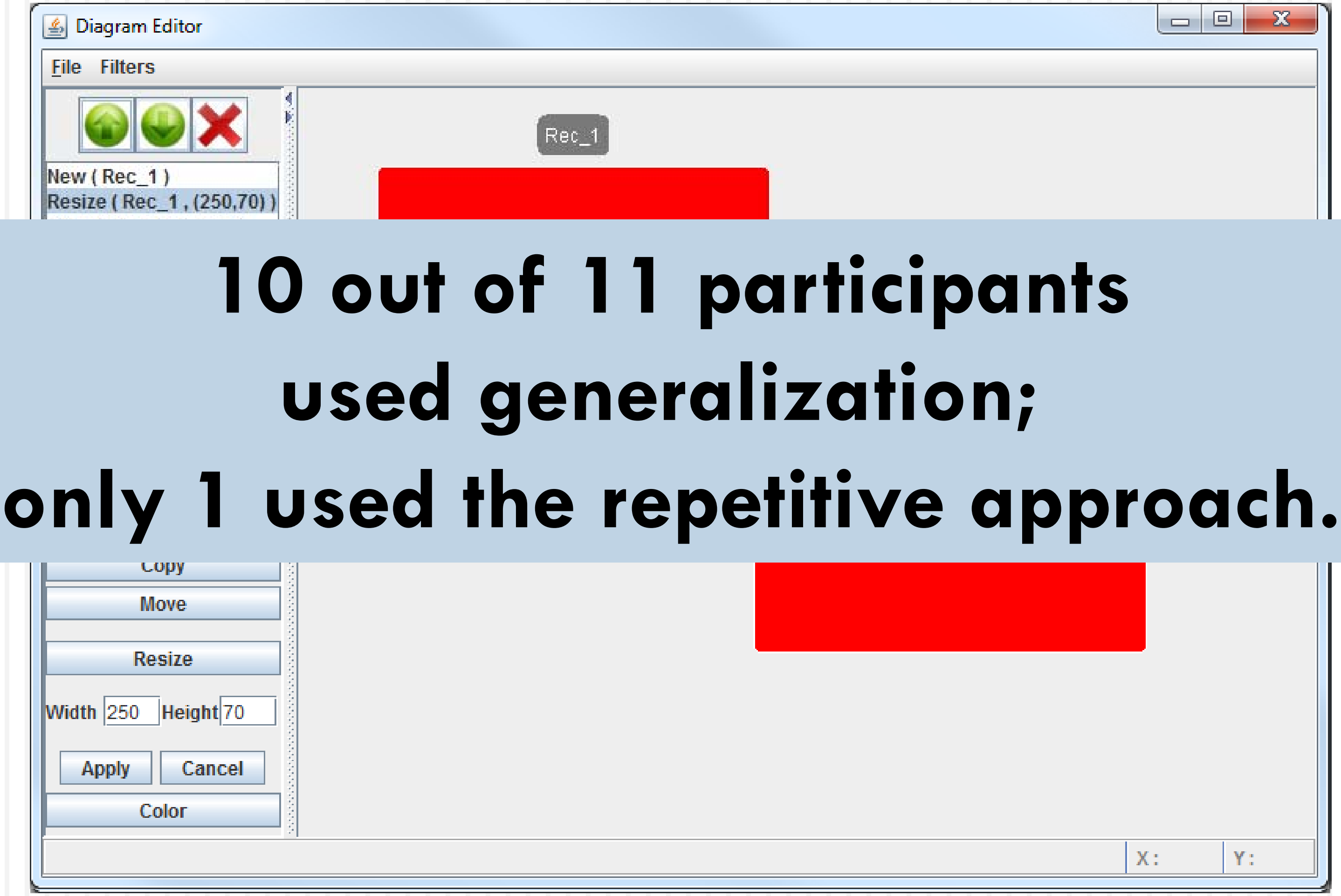
New (Rec_1)
Copy (Rec_1 , Rec_2)
Copy (Rec_1 , Rec_3)
Resize (Rec_1 , (250,70))
Resize (Rec_2 , (250,70))
Resize (Rec_3 , (250,70))

Rec_3

Width 250 Height 70

Example 1:
Solution with Generalization

3) Refer to Figure 3. How would you resize all three rectangles to have a width of 250 and a height of 70?



New (Rec_1)
Resize (Rec_1 , (250,70))

10 out of 11 participants used generalization; only 1 used the repetitive approach.

Example 2: Do Users Understand History?

9. Assuming the default color for a rectangle is red,
   what would happen if you delete the second Color operation (green)?

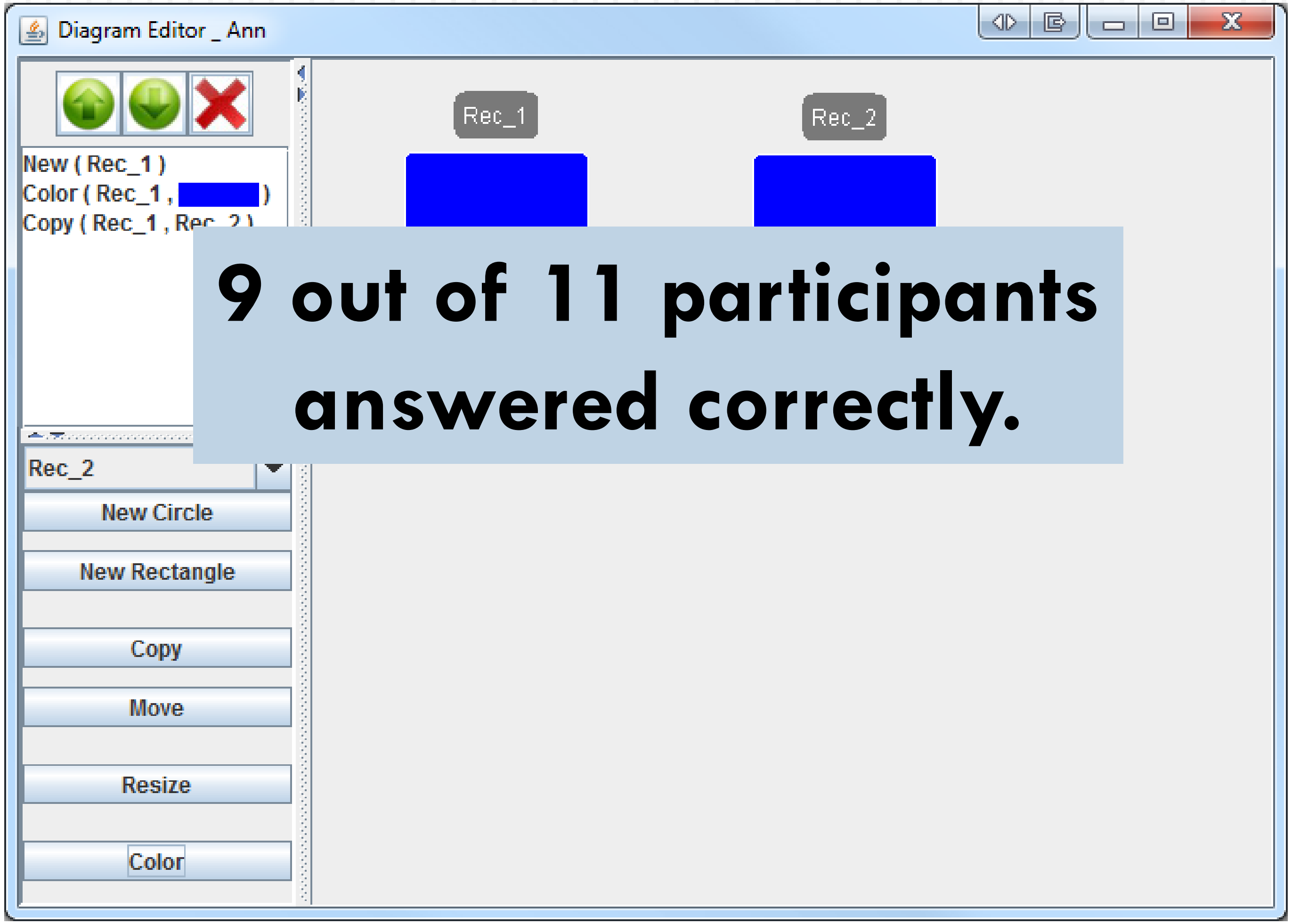


Diagram Editor _ Ann

New (Rec_1)
Color (Rec_1 , 🟦)
Copy (Rec_1 , Rec_2)

9 out of 11 participants answered correctly.

User Study – Results 1

Issues Evaluated	Results	95% Binomial Proportion Central Confidence Interval
1. Applying generalization for non-repetitive case	8/11 used history	[0.43, 0.90]
2. Applying specialization for non-repetitive case	8/11 used history	[0.43, 0.90]
3. Applying generalization for repetitive case	10/11 used history	[0.62, 0.98]
4. - 7. Understanding generalization	11/11 correct	[0.74, 1]
8. Understanding history	10/11 correct	[0.62, 0.98]
9. Understanding history	9/11 correct	[0.52, 0.94]
10. Understanding cascading delete	11/11 correct	[0.74, 1]

User Study – Results II

Likert-Scale Questions

- 10 of 11 participants "*find editing the history of operations a useful feature*" and "*would use this feature if it was included in a drawing application*"
- 95% confidence interval for proportion of sampled population that prefers to use history editing is [0.62, 0.98]

Open Questions

- Showed an unexpected creativity and effort of all participants
- Feedback generally positive with many suggestions,
  e.g. better visualization of history

Conclusion

- History rewriting…
  - gives users more flexibility
  - saves time in merging, generalizing and specializing use-cases
  - leads to new theory
- User study
  - indicates that it is understandable
  - indicates that work in this area is valuable
- Future work:
  better history visualization, more validation