Extending analytics for eye tracking data linked to source code based on iTrace

- Dennis Bøgelund Olesen

Issues and Goals

Issue, motivation

- Eye tracking is used to understand how people comprehend text, commercials and other visual stimuli. There is little work done on the process of comprehension using an editor with no fixed image size.

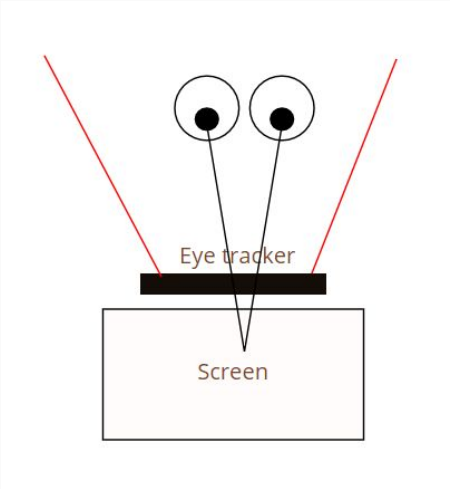
Goal

- Analysis across several users using process mining

Background theory

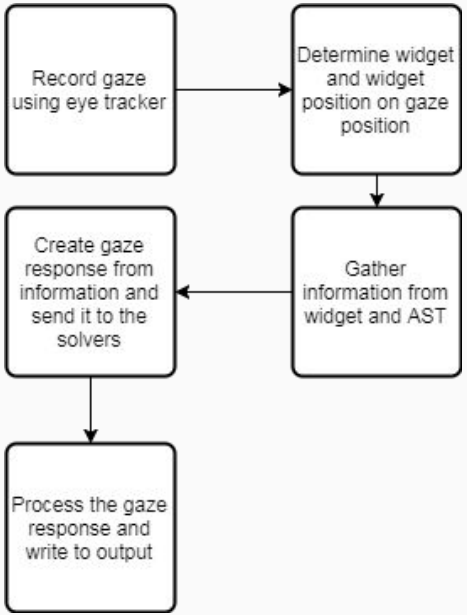
Eye tracking

- Mapping eyes to gaze point
- Eye trackers
- Tobii tracker 4C, laptop
- Gazes, fixations



iTrace

- Eclipse plugin
- Gaze to source code mapping
- XML and JSON output



Process mining

Extracting event logs

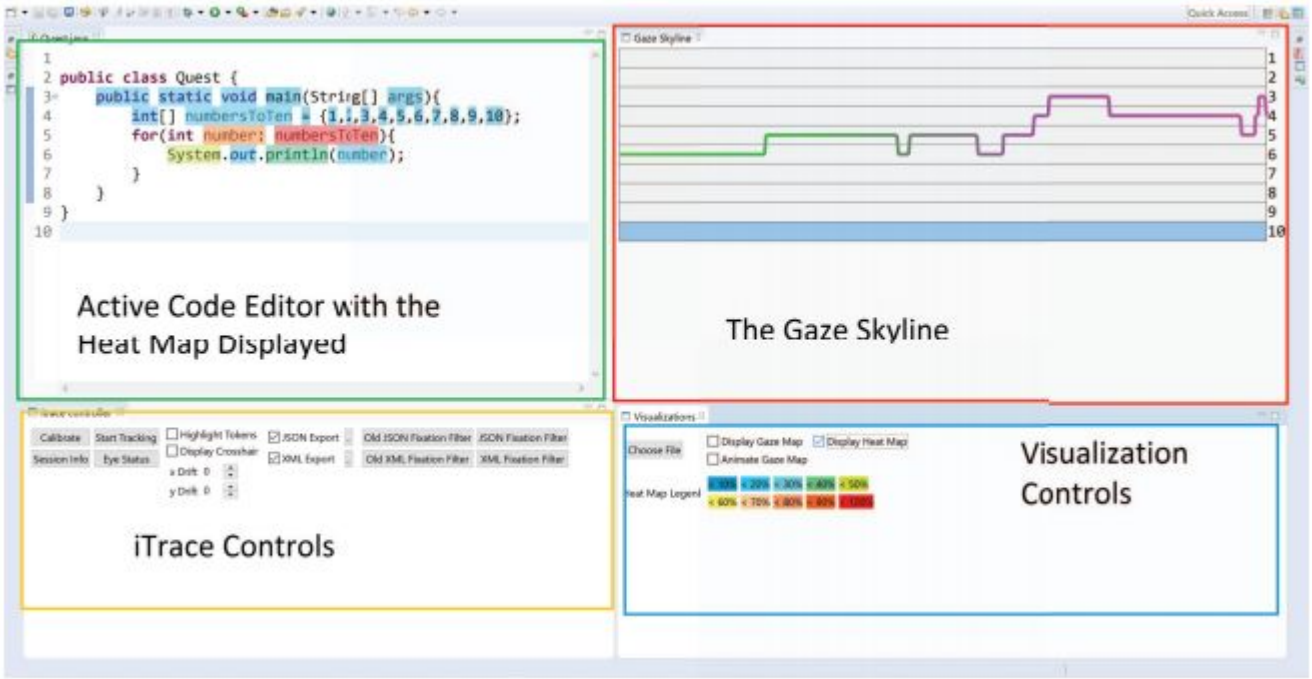
Creates diagrams and statistics

- Most used paths
- Total,avg,mean duration, frequency

Tool: Disco by Fluxicon.

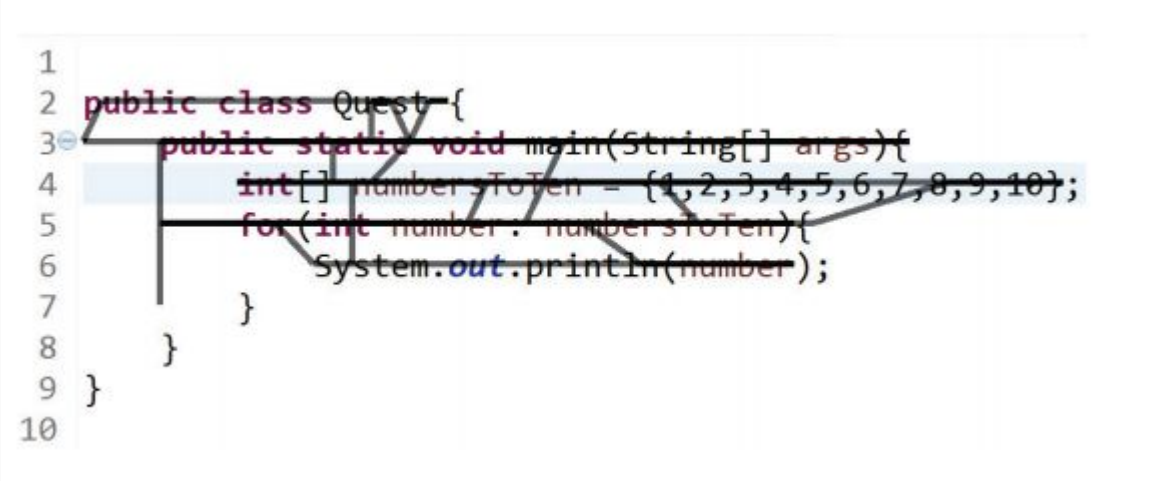
Related work - iTraceVis

- Heatmaps
- Gaze maps
- SkyLines



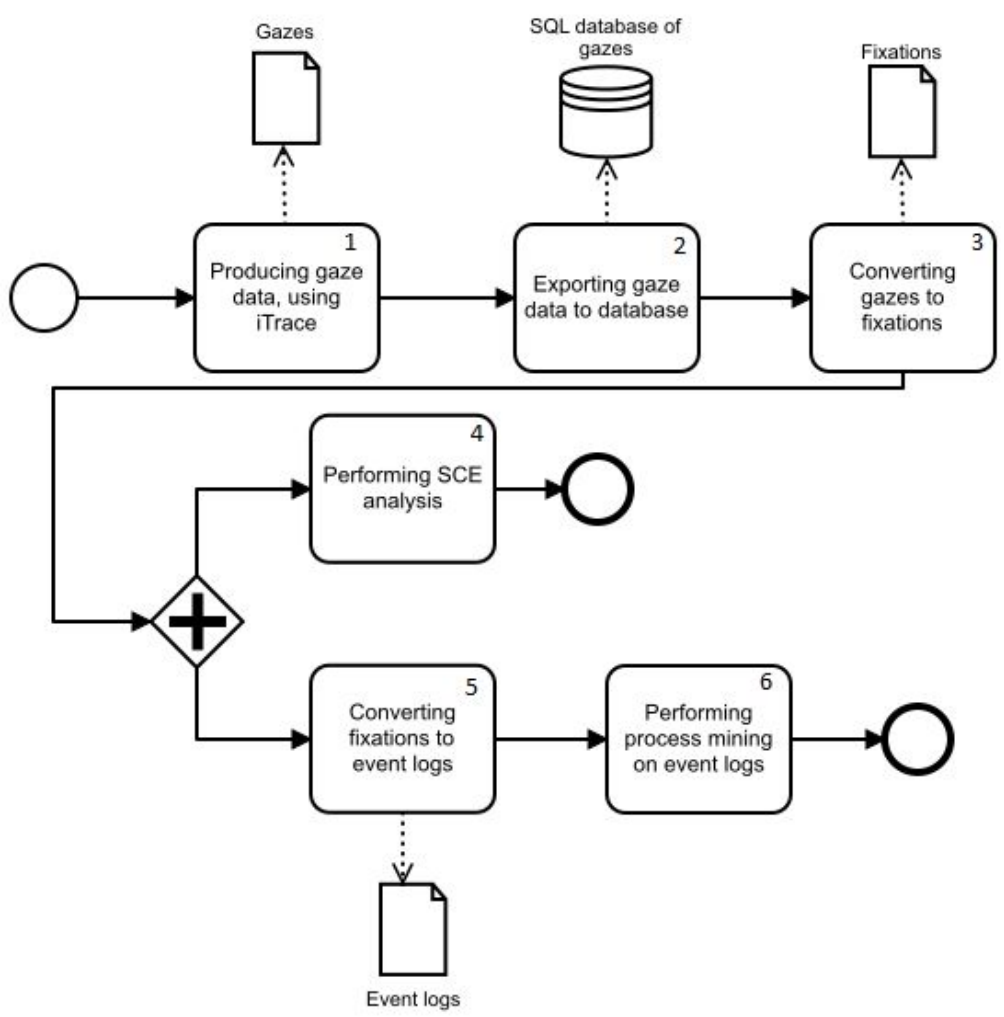
Related work - iTraceVis

- Heatmaps
- Gaze maps
- SkyLines



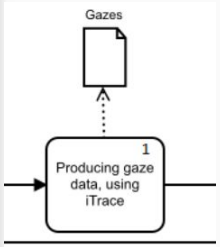
Proposed solution

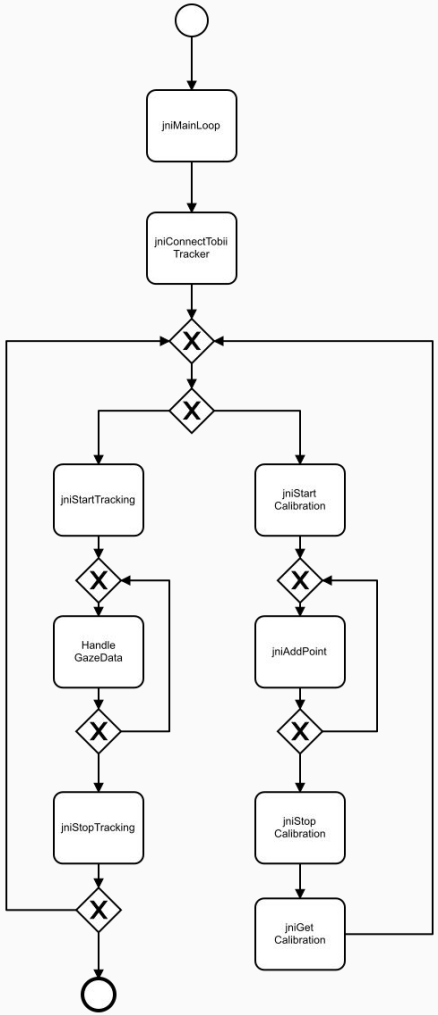
The process


Gazes
SQL database of gazes
Fixations
1
Producing gaze data, using iTrace
2
Exporting gaze data to database
3
Converting gazes to fixations
4
Performing SCE analysis
5
Converting fixations to event logs
6
Performing process mining on event logs
Event logs

Producing gaze data using iTrace

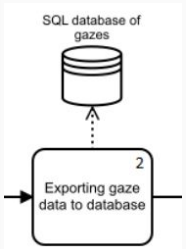
- Java native interface
- Tobii Pro SDK

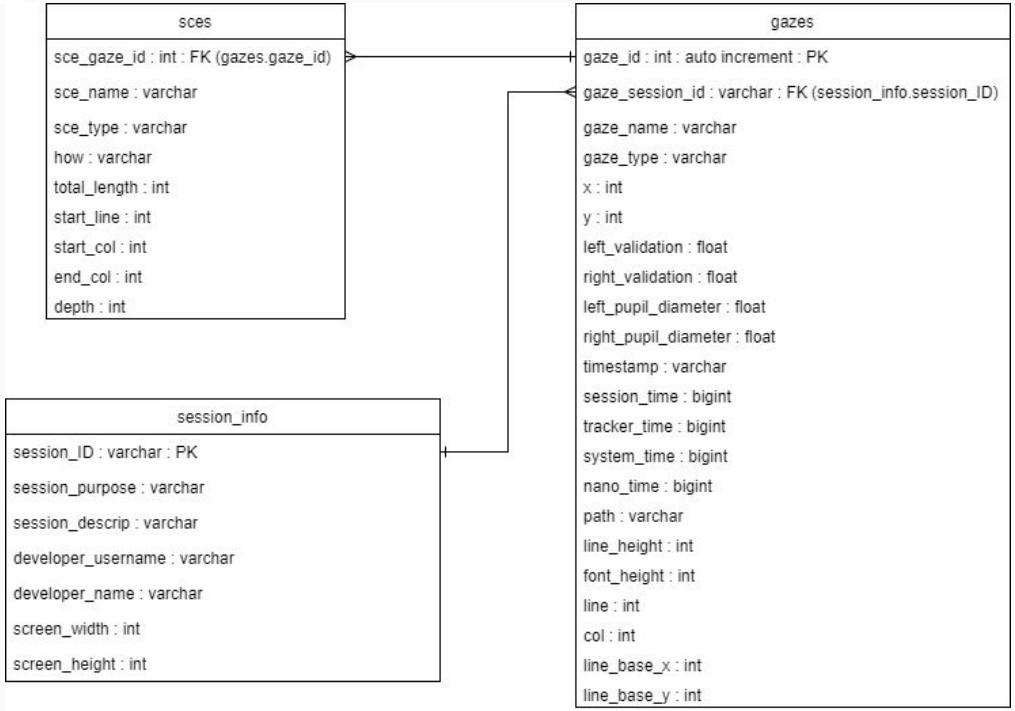




Exporting gaze data to database

- Keep same information
- Old setup used XML or JSON in separate folders.

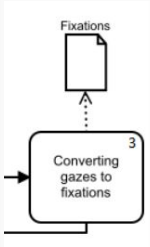




Converting gazes to fixations

Fixation identification algorithms

- Dispersion based, I-DT

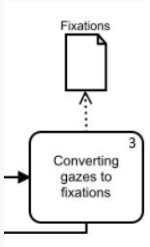



```
Loop over each gaze
  Add points until the duration threshold is covered

  If dispersion of window points <= threshold
    Add next gazes until dispersion threshold > threshold
    Calculate the centroid
    Collapse into a fixation.
  If dispersion of window points > threshold
    remove first point in window.
```

Converting gazes to fixations

Fixation identification algorithms

- Velocity based, I-VT

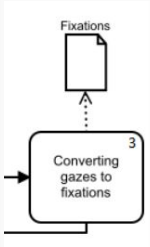



```
Calculate point to point velocity for each gaze

Label velocities under threshold as a fixation point

Collapse consecutive fixation points into a fixation group
Remove saccades

For each fixation group,
  determine the start time and the end time
  find the centroid of the points

Return the fixations
```

Converting gazes to fixations

Fixation identification algorithms

- Result of the dispersion based implementation.



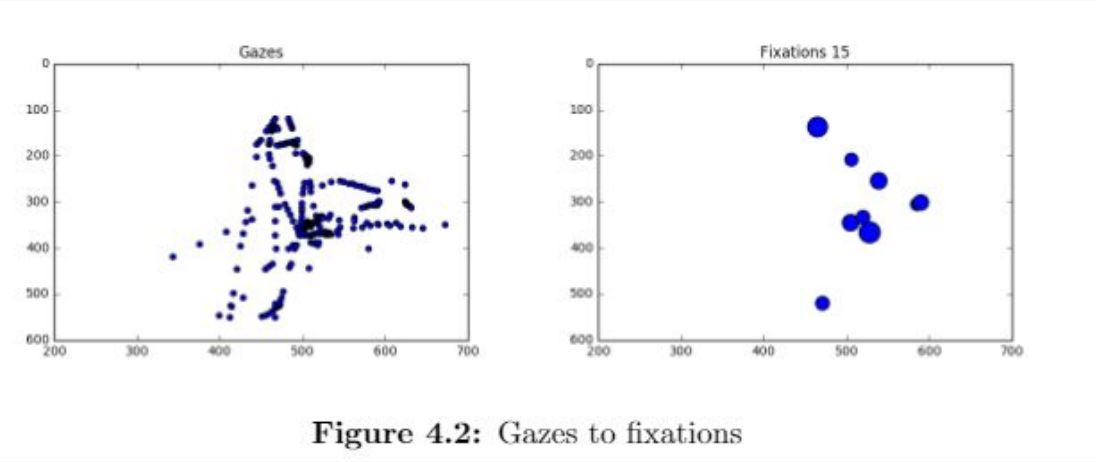


Figure 4.2: Gazes to fixations

Collecting data from participants

What we did:

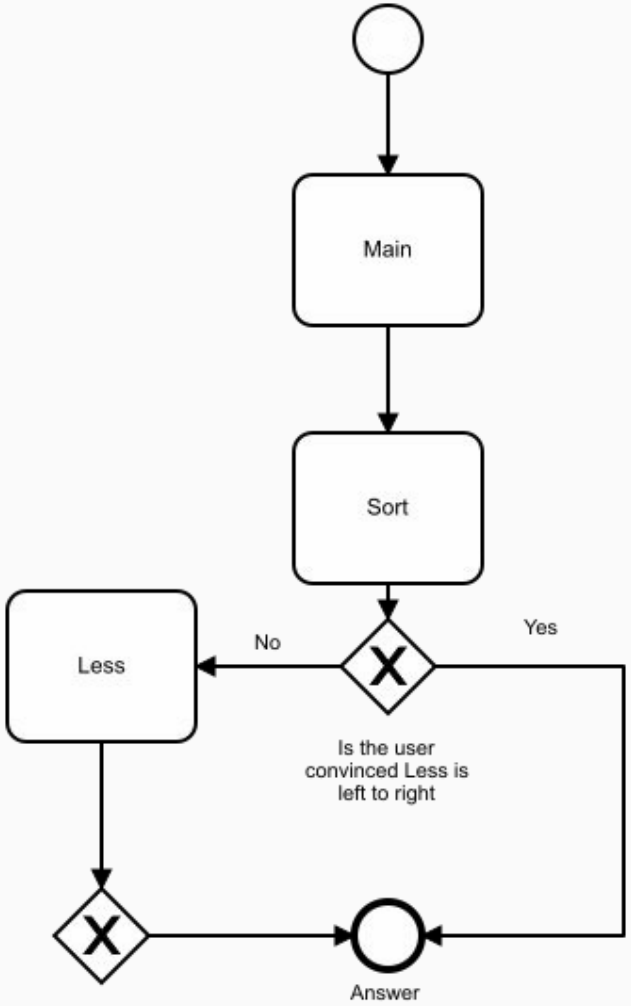
- Insertion Sort
- Five users
- Four finished

Test quality:

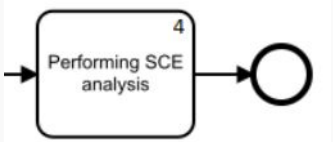
- Calibration within a line accuracy
- Informal, but focused.

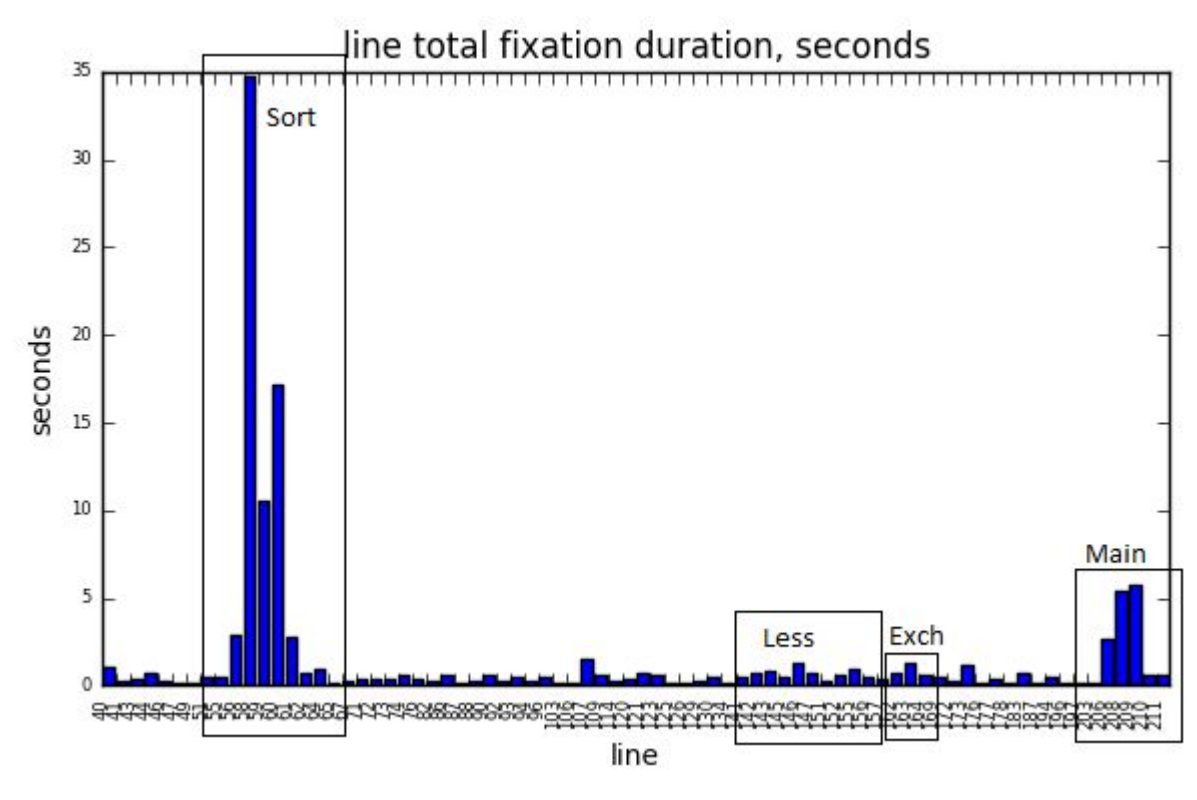
Expected output:

- Main -> Sort -> less (See Model)



Performing SCE analysis



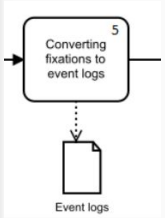


Converting fixations to event logs

Area of interest

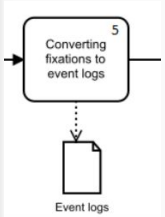
Event log

- Consecutive fixations on the same area are collapsed
- Session ID, area name, start, end



Line by line events

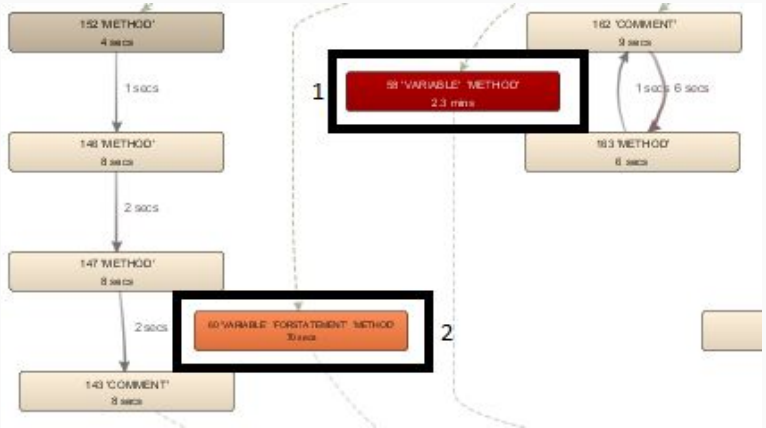
- Generic, works with all single-file code.
- Large and difficult to read for larger files
- Specific, based on small areas of interest.




```
57      */
58      public static void sort(Comparable[] a) {
59          int n = a.length;
60          for (int i = 1; i < n; i++) {
61              for (int j = i; j > 0 && less(a[j], a[j-1]); j--) {
62                  exch(a, j, j-1);
63              }
64              assert isSorted(a, 0, i);
65
66          }
67          assert isSorted(a);
68      }
69
```

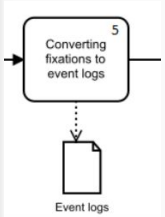
1

2

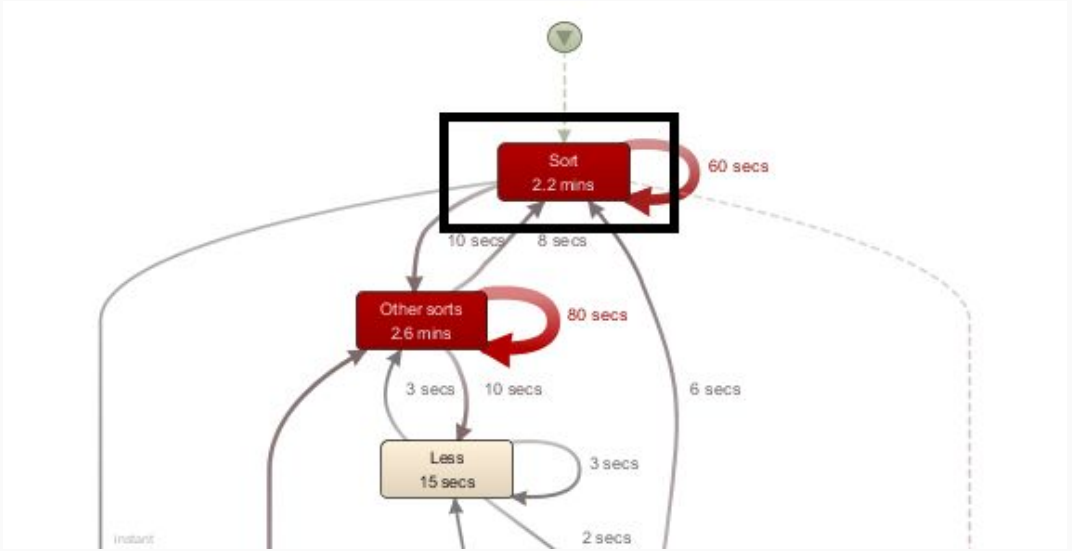


User defined events

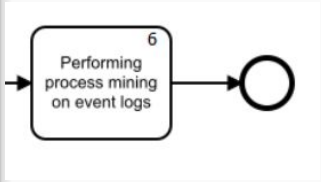
- User defined
- Requires definition of areas of interest.
- Size depends on AoI definitions.
- May be easier to read due to fewer defined areas.

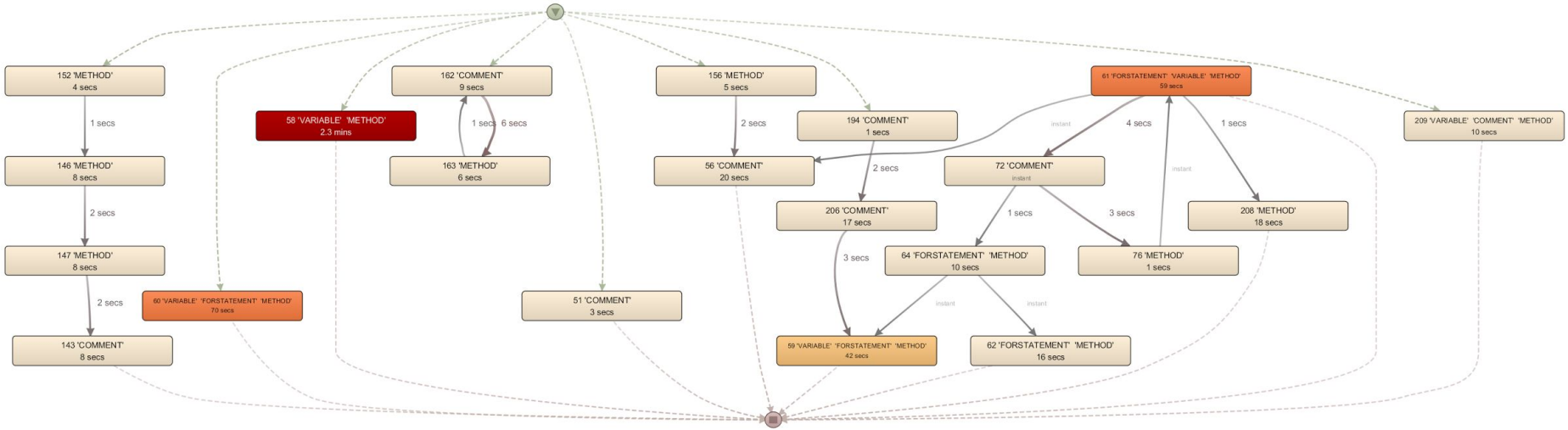



```
57      */
58     public static void sort(Comparable[] a) {
59         int n = a.length;
60         for (int i = 1; i < n; i++) {
61             for (int j = i; j > 0 && less(a[j], a[j-1]); j--) {
62                 exch(a, j, j-1);
63             }
64             assert isSorted(a, 0, i);
65
66         }
67         assert isSorted(a);
68     }
69
```



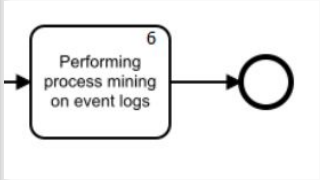
Performing process mining on event logs

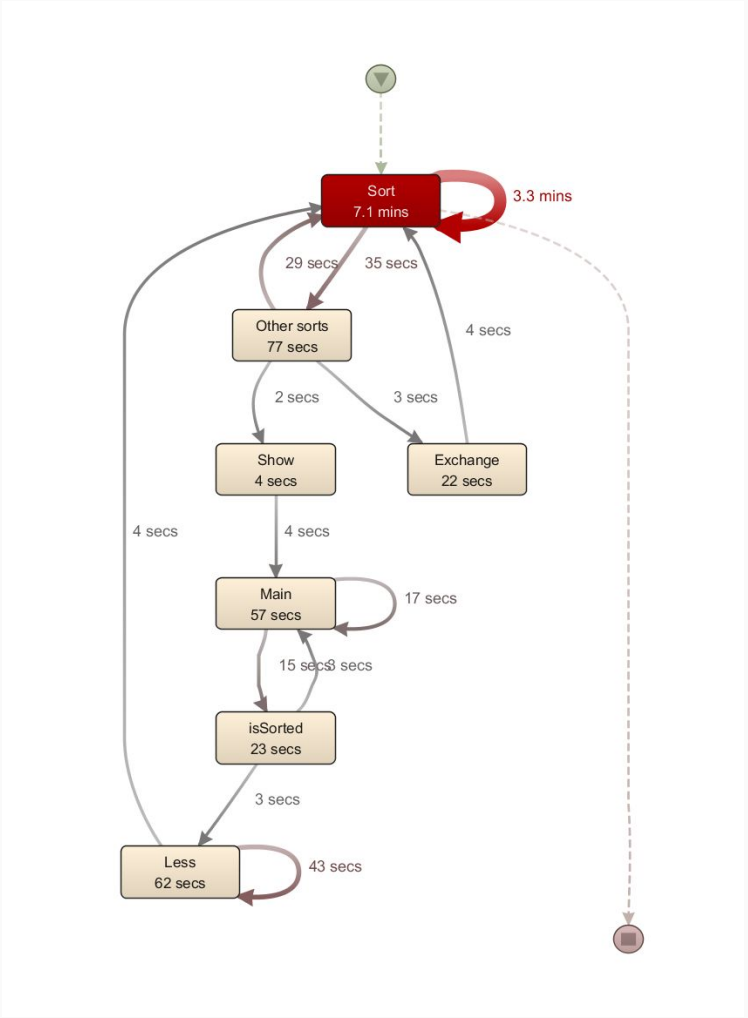




Performing process mining on event logs

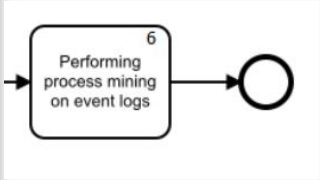
- Sort most viewed
- Exchange independent
- Flow: Top (Sort) -> Main -> Less -> Sort

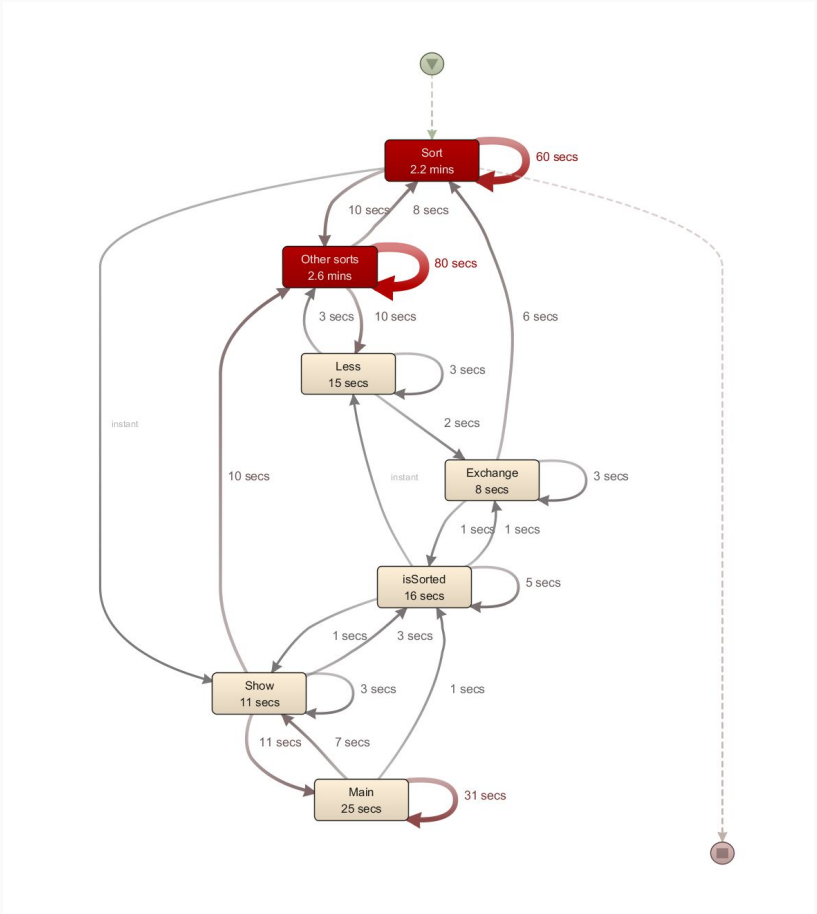




Performing process mining on event logs

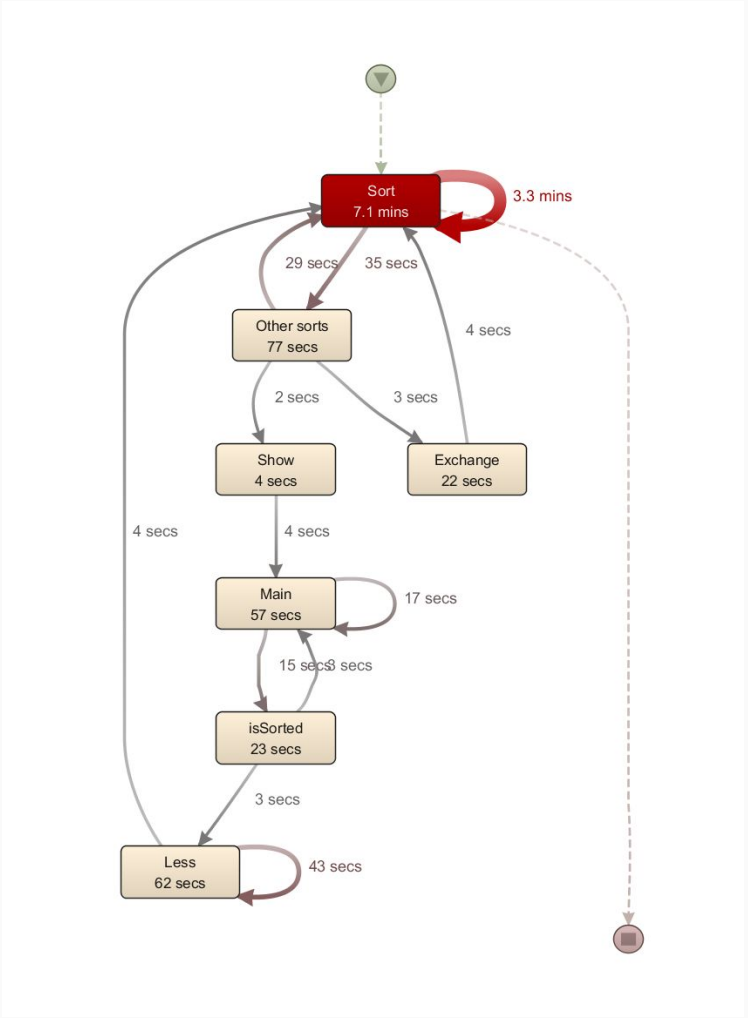
- User who did not finish
- Lots of time spent in the wrong sections
- Read everything in order, with some backtracking.





Conclusion

- The process collects the data and returns event logs readable by process mining tools
- The data presented this way allow for extensive exploration



Future work

- More complex and complete process mining exploration of the data
- Conformance checking against expected subpaths.
- Automatic heuristic scores to detect users not performing as expected

Thanks!

Extending analytics for eye tracking data linked to source code based on iTrace

- Dennis Bøgelund Olesen

Danmarks Tekniske Universitet