UNIVERSITÄT DUISBURG ESSEN

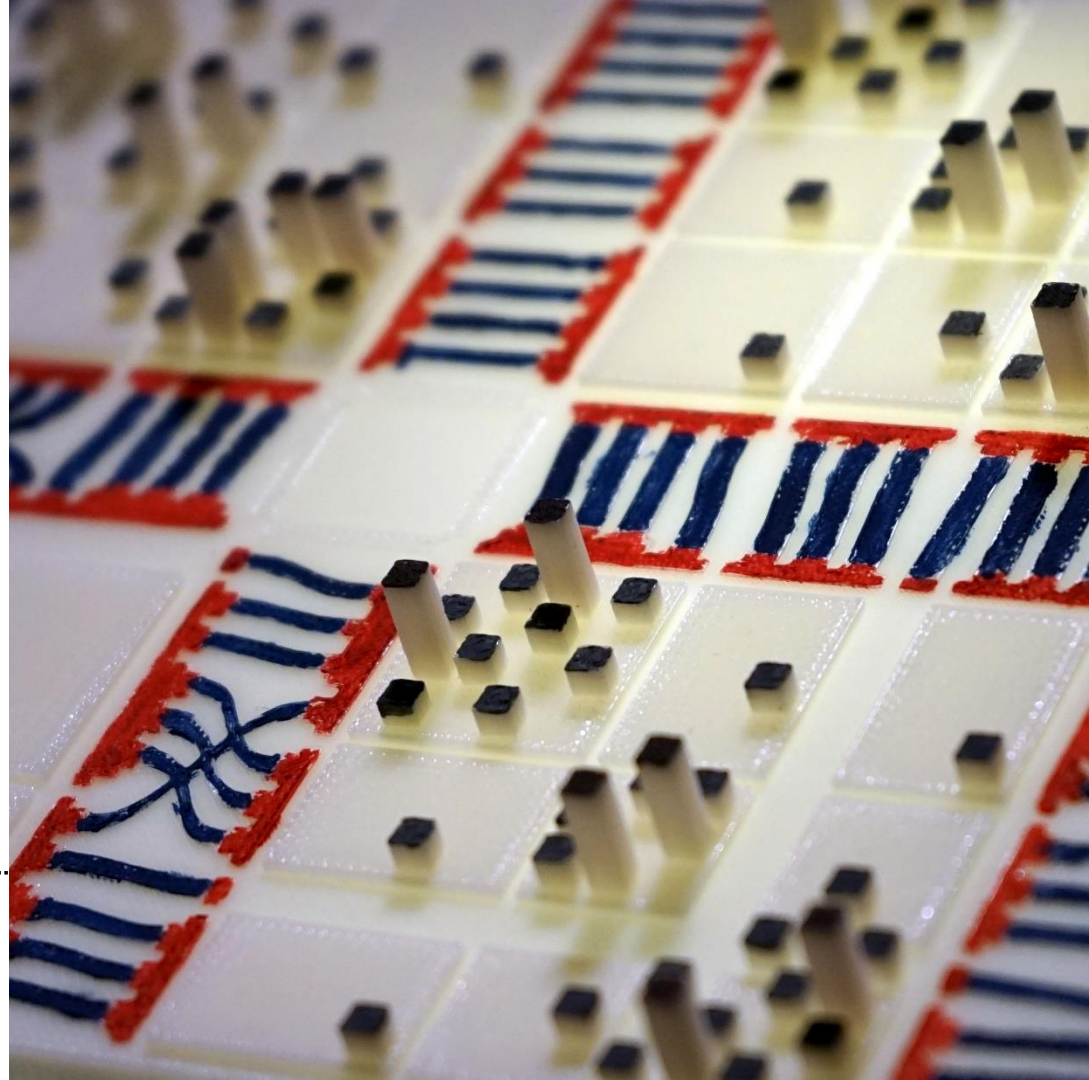
Open-Minded

PALUNO
The Ruhr Institute for Software Technology

Visualizing Software Dynamics

Fabian Beck

Keynote at the
8th Symposium on Software Performance 2017,
Nov 9, Karlsruhe, Germany



Architecture Modularization

Software Engineering

Evolution Coupling

Comprehension

Visualizing Systems and Software Performance

GI-Dagstuhl seminar for young researchers, July 9-13, 2018

https://vssp.github.io/

My Background

Software Visualization

Performance Engineering

Graphs

Visualization Hierarchies

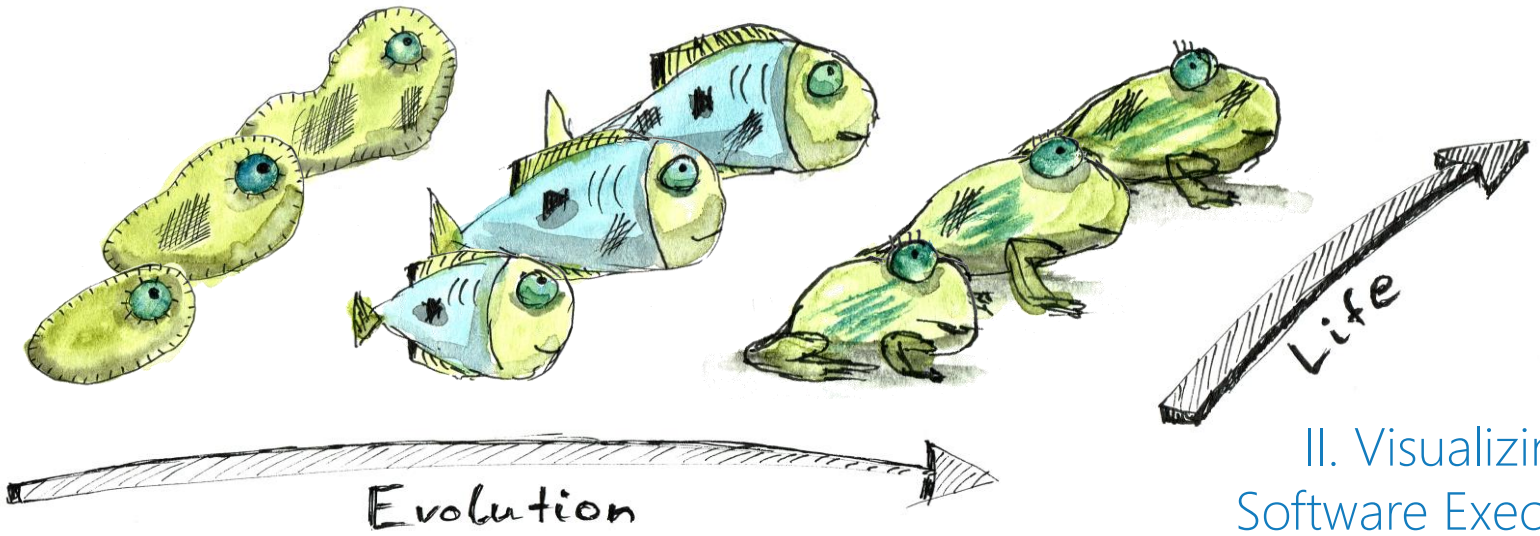
Temporal data

Visual comparison

Performance data

Dynamic analysis

III. Visualizing the Evolution of Executions



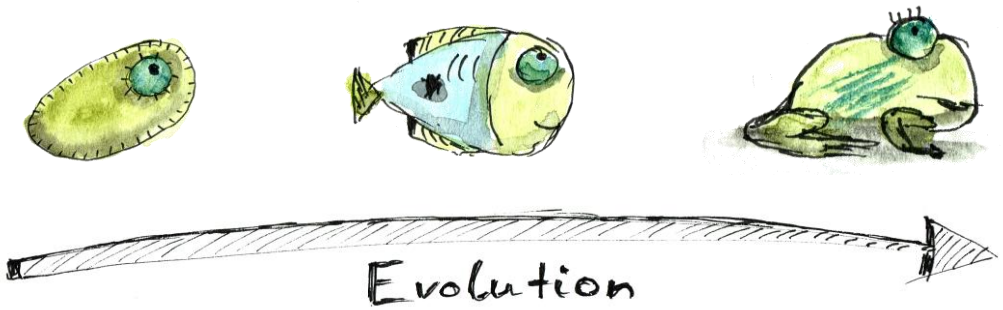
II. Visualizing Software Execution

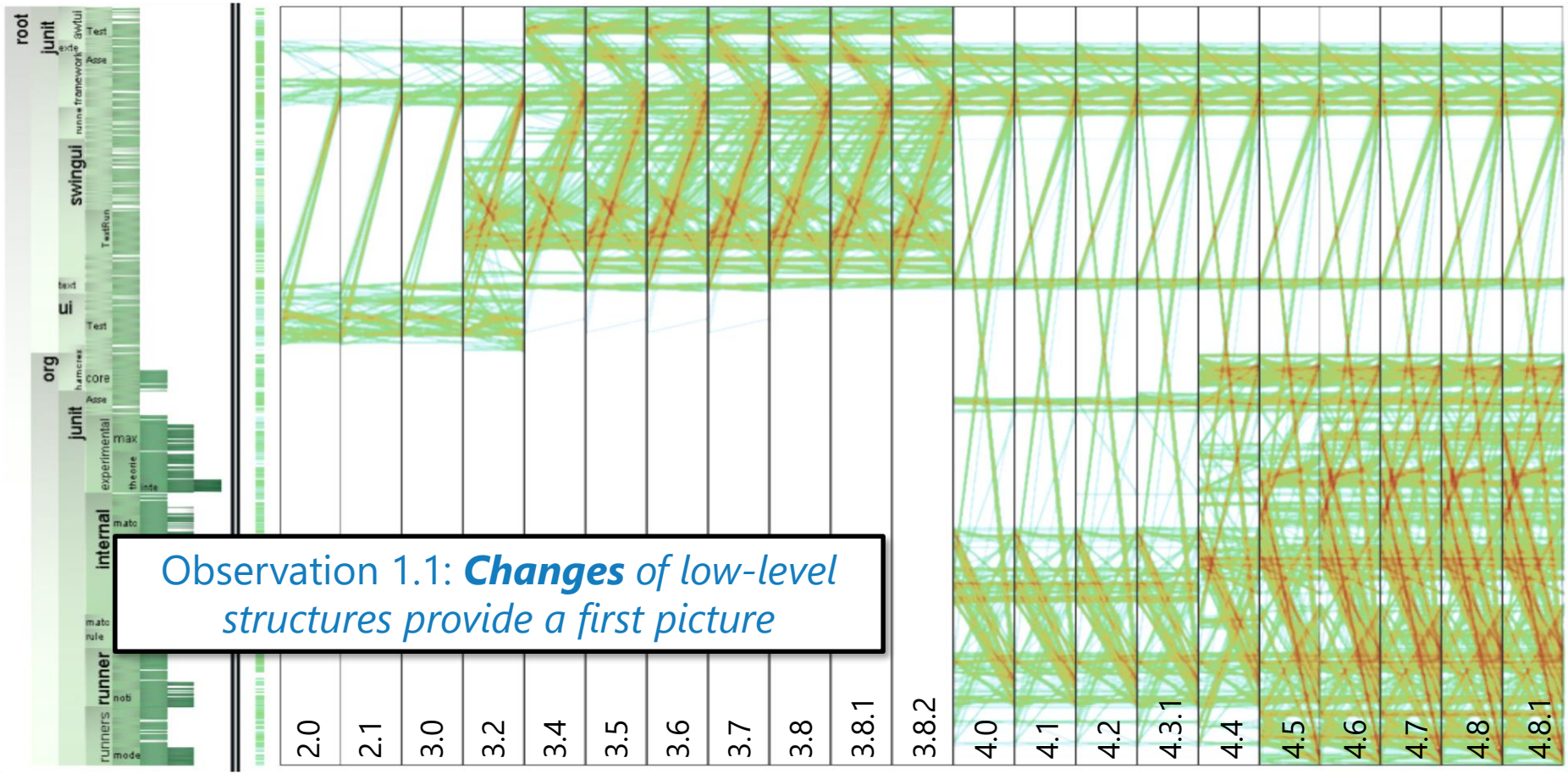
I. Visualizing Software Evolution

IV. Challenges

Software Dynamics

I. Visualizing Software Evolution

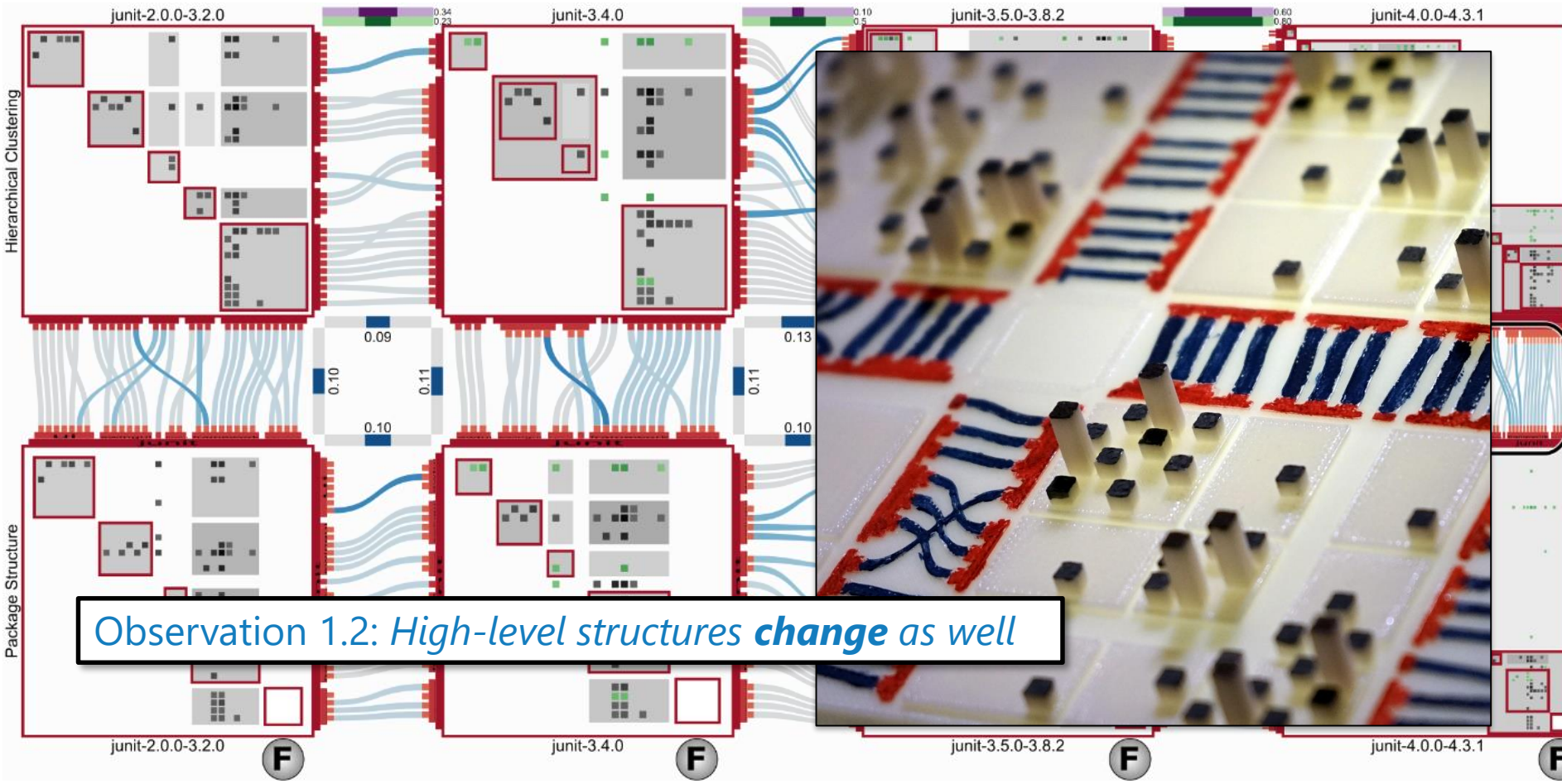




Observation 1.1: ***Changes*** *of low-level structures provide a first picture*

Evolution of Call Graphs

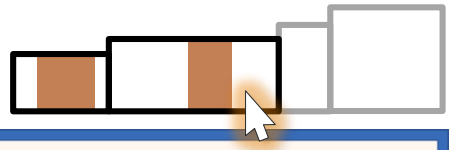
Burch, M., Vehlow, C., Beck, F., Diehl, S., and Weiskopf, D., 2011. **Parallel Edge Splatting for Scalable Dynamic Graph Visualization.** In *IEEE Transactions on Visualization and Computer Graphics* (Vol. 17, No. 12, pp. 2344–2353). DOI: 10.1109/tvcg.2011.226.



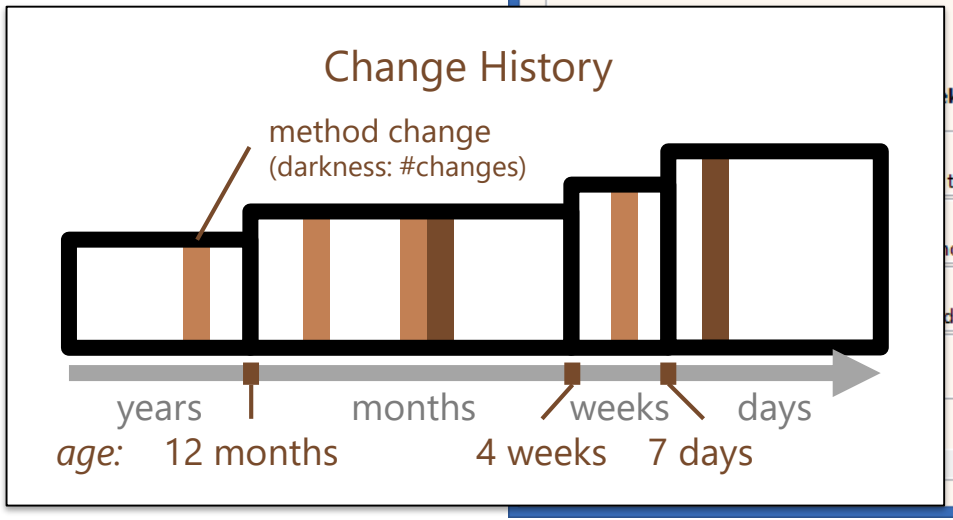
Dynamic Hierarchies

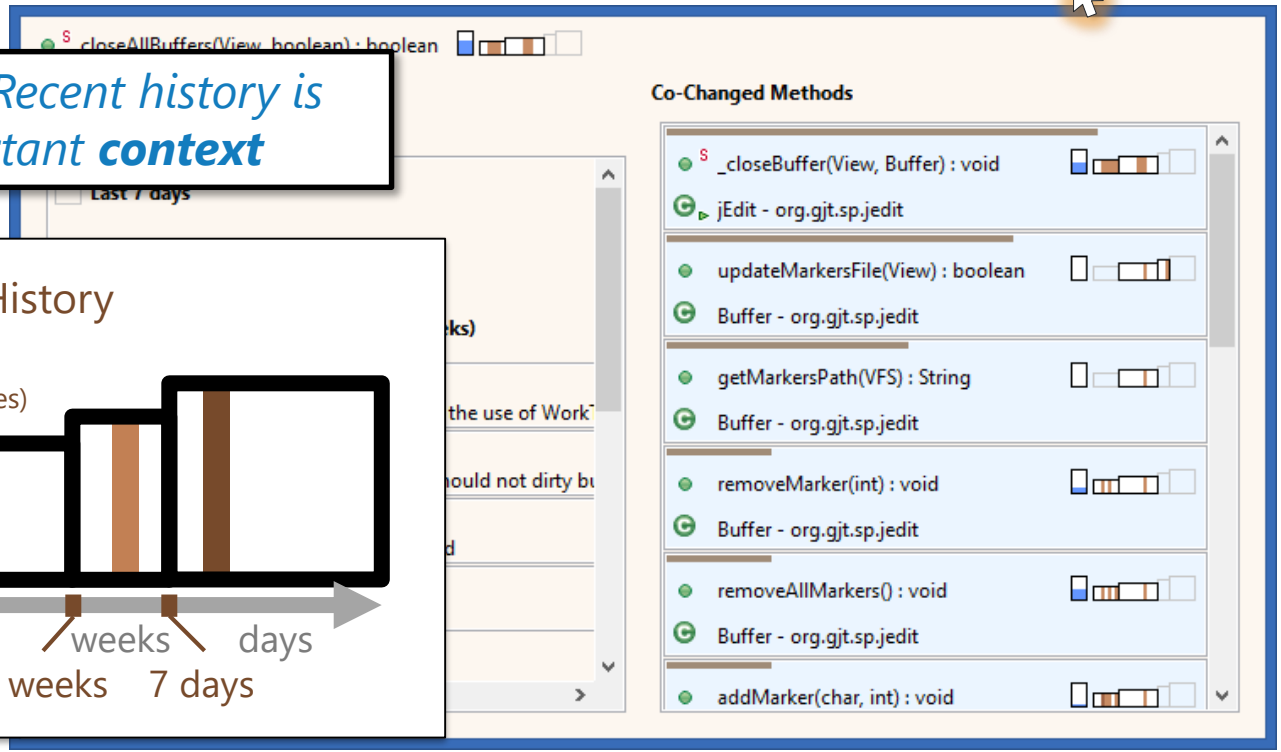
Vehlow, C., Beck, F., and Weiskopf, D., 2016. **Visualizing Dynamic Hierarchies in Graph Sequences**. In *IEEE Transactions on Visualization and Computer Graphics* (Vol. 22, No. 10, pp. 2343–2357). DOI: 10.1109/TVCG.2015.2507595.

`closeAllBuffers (…)`



Observation 1.3: *Recent history is often an important* ***context***

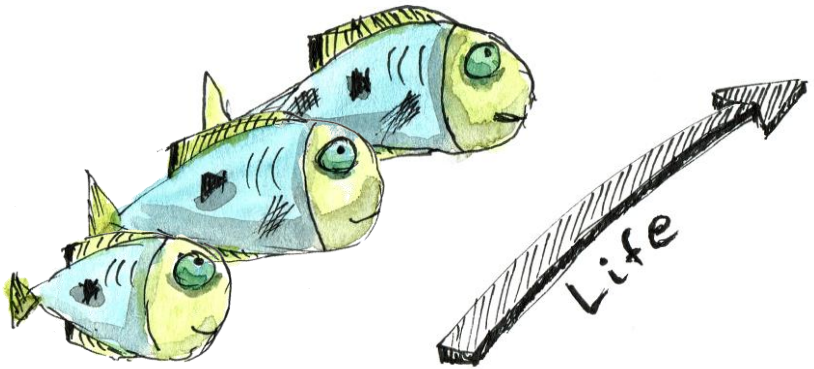


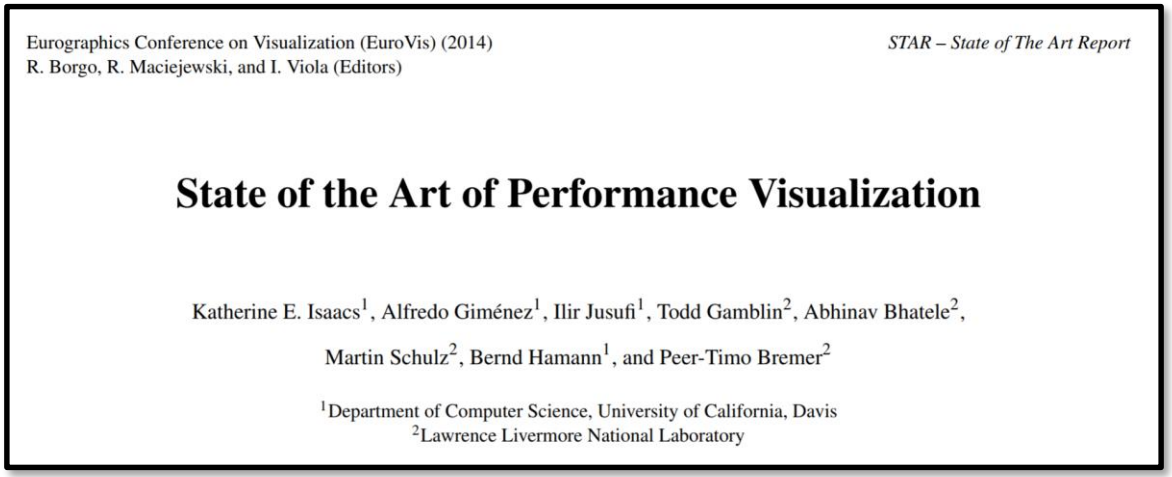


Embedding Evolutionary Context

Beck, F., Dit, B., Velasco-Madden, J., Weiskopf, D., and Poshyvanyk, D., 2015. **Rethinking User Interfaces for Feature Location.** In *Proceedings of the 23rd IEEE International Conference on Program Comprehension* (pp. 151–162). DOI: 10.1109/ICPC.2015.24.

II. Visualizing Software Execution





Eurographics Conference on Visualization (EuroVis) (2014)
R. Borgo, R. Maciejewski, and I. Viola (Editors)

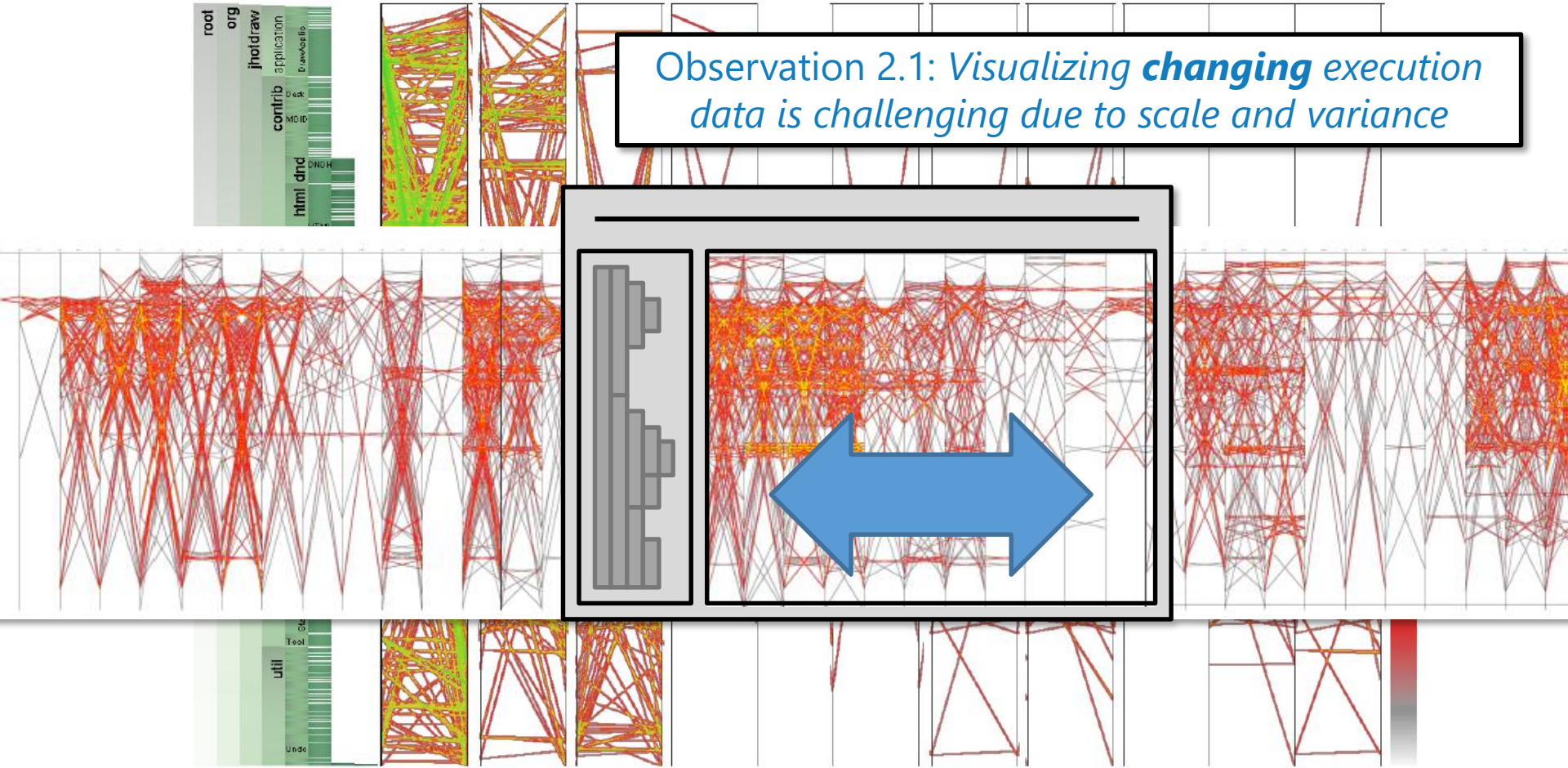
STAR – State of The Art Report

State of the Art of Performance Visualization

Katherine E. Isaacs[1], Alfredo Giménez[1], Ilir Jusufi[1], Todd Gamblin[2], Abhinav Bhatele[2],
Martin Schulz[2], Bernd Hamann[1], and Peer-Timo Bremer[2]

[1]Department of Computer Science, University of California, Davis
[2]Lawrence Livermore National Laboratory

Isaacs, K. E., Giménez, A., Jusufi, I., Gamblin, T., Bhatele, A., Schulz, M., Hamann, B., and Bremer, P.-T., 2014. **State of the Art of Performance Visualization**. In *EuroVis - STARs* (pp. 141–160). DOI: 10.2312/eurovisstar.20141177.



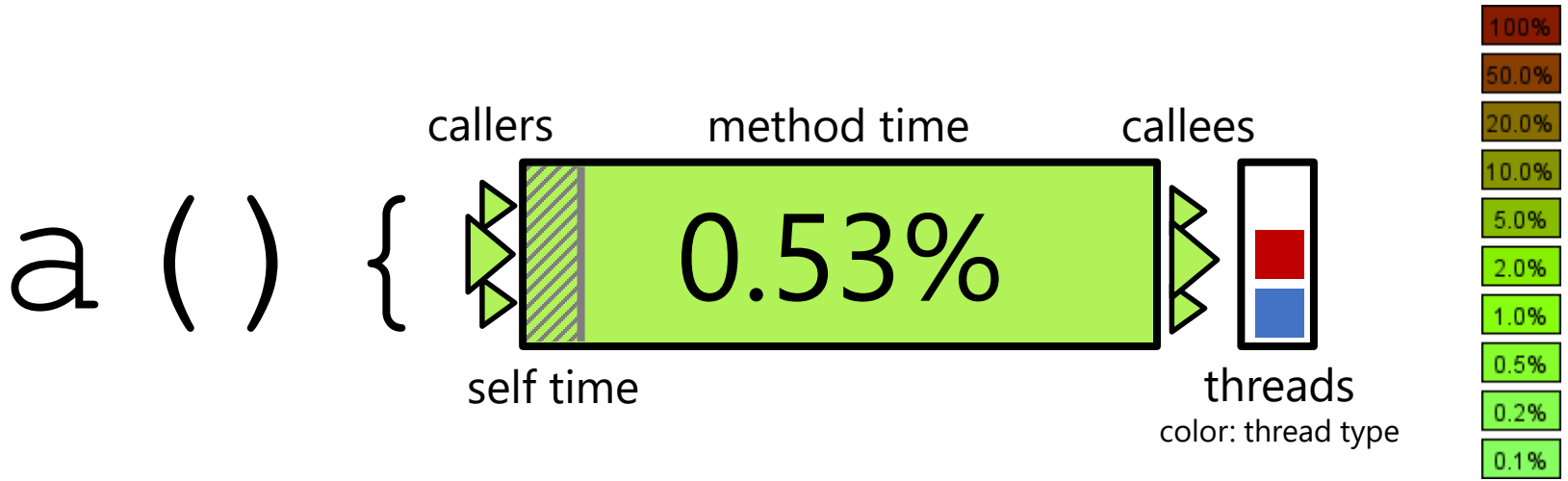
Observation 2.1: *Visualizing **changing** execution data is challenging due to scale and variance*

Dynamic Call Graphs

Beck, F., Burch, M., Vehlow, C., Diehl, S., and Weiskopf, D., 2012. **Rapid Serial Visual Presentation in Dynamic Graph Visualization**. In *Proceedings of the 2012 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 185–192). DOI: 10.1109/vlhcc.2012.6344514.

```
public EntitySet getIncludedEnt
    EntitySet entitySet = new E
    entitySet.addEntitySet(dire
    for (Cluster cluster : subClusters) {
        EntitySet includedEntitySet = cluster.getIncludedEntities();  63.48%
        entitySet.addEntitySet(includedEntitySet);  71.81%
    }
```

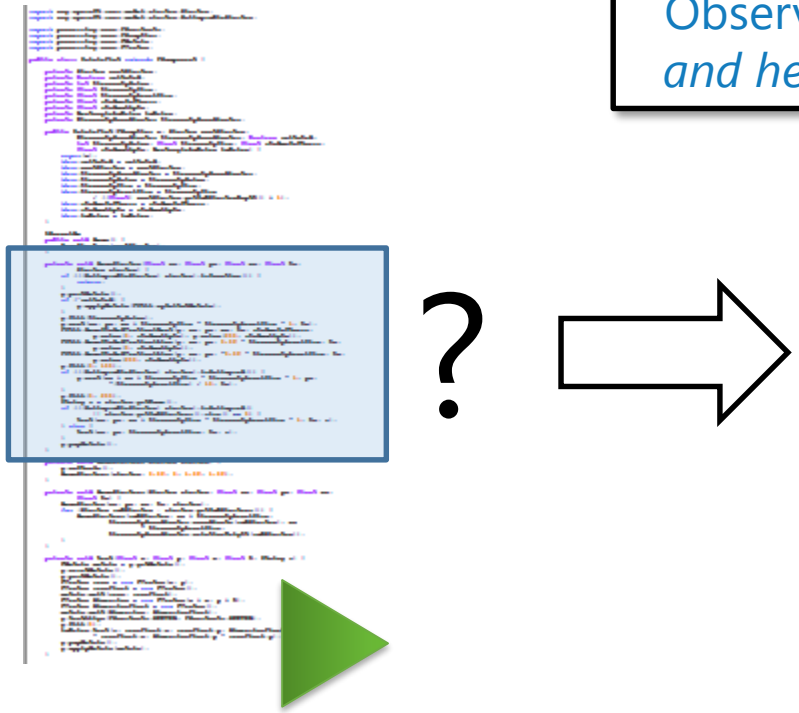
Observation 2.2: *The code is necessary **context** to **comprehend** performance information*



Embedding Performance Context

Beck, F., Moseler, O., Diehl, S., and Rey, G. D., 2013. **In Situ Understanding of Performance Bottlenecks through Visually Augmented Code.** In *Proceedings of the 21st IEEE International Conference on Program Comprehension* (pp. 63–72). DOI: 10.1109/ICPC.2013.6613834.

Method Execution Reports



Observation 2.3: *Natural-language text can explain and help better **comprehend** complex runtime data*

TreeMapPanel.paintEntries

Summary Method Calls Time Consumption

Method Calls

Method paintEntries is a direct recursive method which was called 1846 times 100% out of which 1845 times 99.9% it was recursive. It was called non-recursively only 1 time 0.1% by method paint.

Method paintEntries made 25683 calls 100% out of which 1845 calls 7.2% were direct recursive to itself. It made 23838 calls 92.8% to 17 methods. It made maximum 3691 calls 14.4% to method getScaledSize.

Recursion depth of method paintEntries went up to 11 levels. It reached depth level 9 maximum times which was 498.

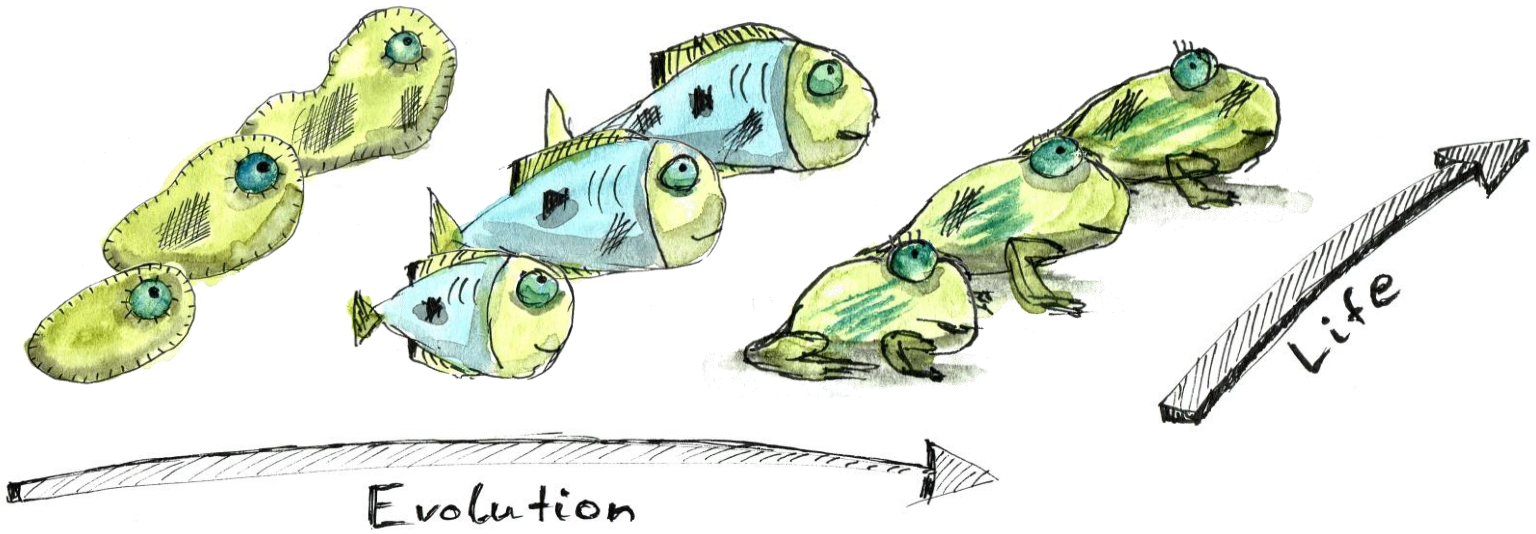
Time Consumption

Method paintEntries took 65.60 ms 100%, out of which 33.03 ms 50.4% were consumed as self time and 32.57 ms 49.6% by outgoing calls made to 17 methods. Outgoing calls made to method drawRect took maximum time which was 24.11 ms 36.8%. Please note the measurements are uncertain due to short runtime of outgoing calls.

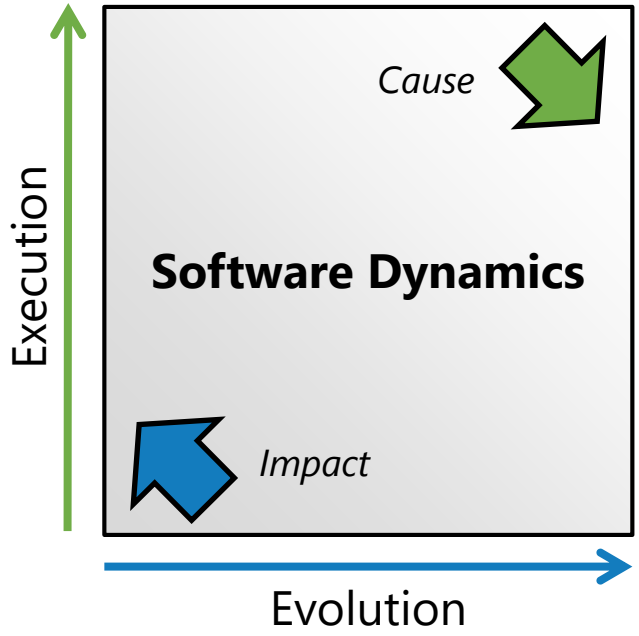
```
/**
 * Computes the position of entries and paints them on the canvas.
 */
private void paintEntries(Entry entry, boolean splitHorizontal, Graphics g) {
    if (entry.getScaledSize() == 0) return;
    try {
        SingleFile sf = (SingleFile) entry;
        drawRect(g, sf);
    } catch (ClassCastException e) {
        Dir dir = (Dir) entry;
        float x = dir.getX();
        float y = dir.getY();
```

Beck, F., Siddiqui, H. A., Bergel, A., and Weiskopf, D., 2017. **Method Execution Reports: Generating Text and Visualization to Describe Program Behavior.** In *Proceedings of the 5th IEEE Working Conference on Software Visualization,* to appear.
https://fabian-beck.github.io/Method-Execution-Reports/

III. Visualizing the Evolution of Executions

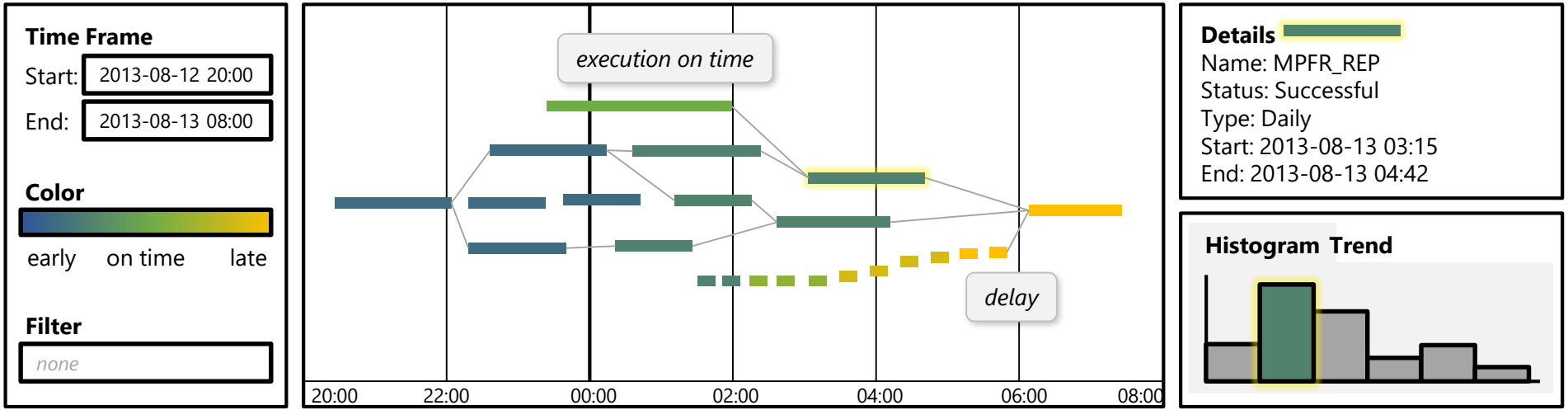


Execution
- Time: ms – min
- Data:
  - Call graph
  - Object graph
  - Performance
  - Memory



Evolution
- Time: h – years
- Data:
  - Changes
  - Commits
  - Developers
  - Issues

Observation 3.1: *History provides additional **context** for interpreting performance information*



Stored Procedures of a Data Warehouse

Meyer, M., Beck, F., and Lohmann, S., 2016. Visual monitoring of process runs: An application study for stored procedures. In *Proceedings of the 2016 IEEE Pacific Visualization Symposium* (pp. 160–167). DOI: 10.1109/PACIFICVIS.2016.7465264.

Observation 3.2: *Diverse **context** is necessary to **comprehend** performance regressions*

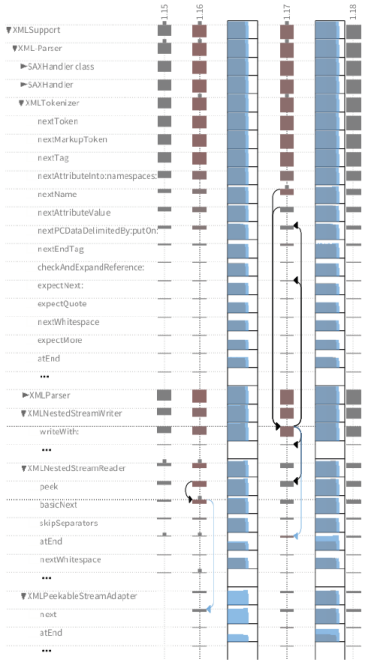
Evolution of code

Performance Regression

Benchmark

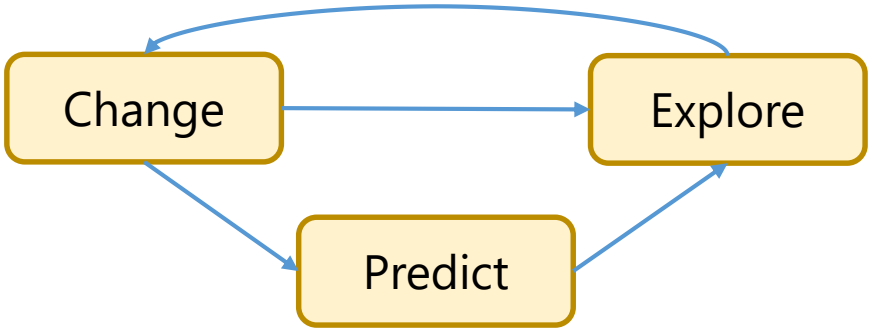
Context:

- Modularization
- Dynamic calls
- Execution timelines
- Code diffs

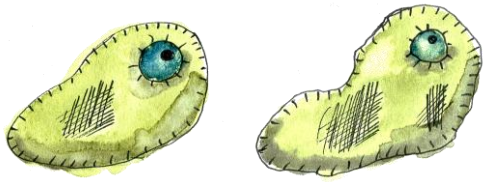


Ongoing collaboration with Juan Pablo Sandoval Alcocer and Alexandre Bergel

Observation 3.3: *Interactively **changing** a system and exploring the (predicted) effects of changes is desired*



IV. Challenges

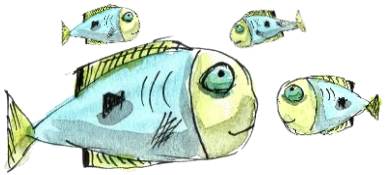


Change

- History: What happened?
- Future: What if?

Context

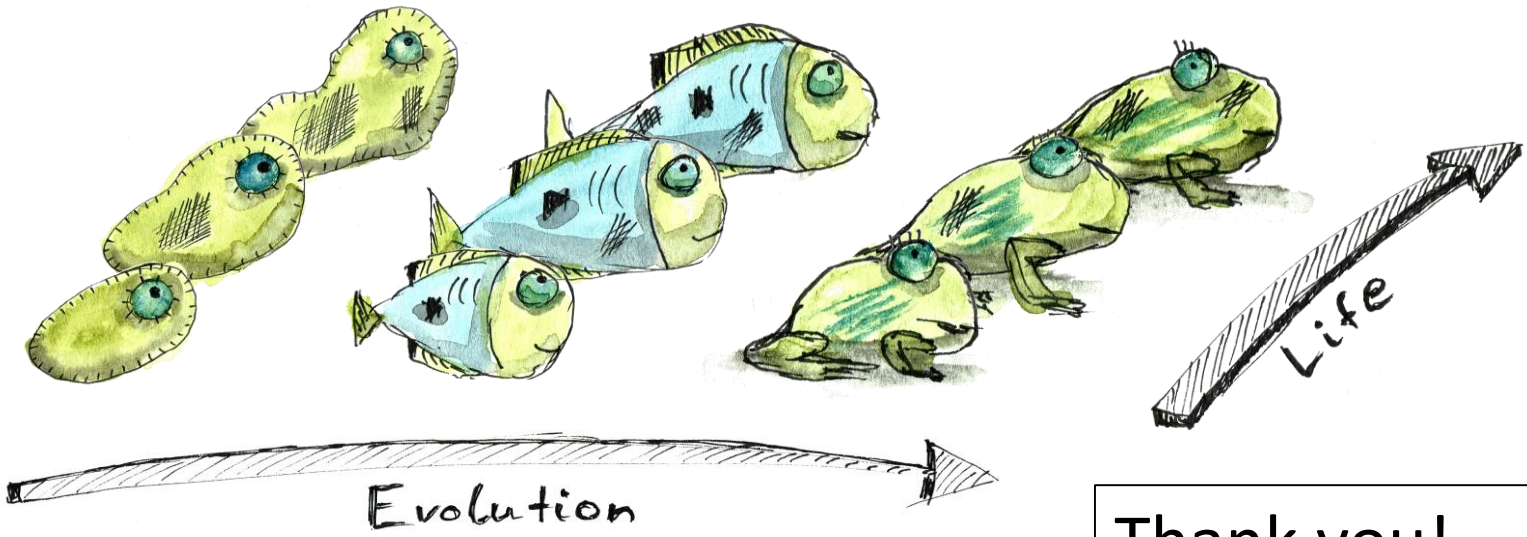
- What context information is required?
- How to integrate required data sources and analysis methods?



Comprehension

- How to present the analysis in a understandable and self-explaining way?

Visualizing Software Dynamics



Change Context Comprehension

Thank you!

Fabian Beck
fabian.beck@paluno.uni.due.de
@beck_fabian

Visualizing Software Dynamics

Abstract: Software is not just a static set of code fragments, it is dynamic – programs show dynamic behavior when being executed and software systems evolve over time. This talk introduces how visualization helps better understand and analyze these two dimensions of software dynamics. I present both architecture-centric overview visualizations and detailed code-centric visual representations to support various software maintenance and performance engineering tasks. Finally, I discuss challenges for leveraging the two dynamic dimensions within integrated visualizations.

Biography: Fabian Beck is assistant professor at the University of Duisburg-Essen, Germany. He received the Dr. rer. nat. (PhD) degree in computer science from the University of Trier, Germany in 2013. He worked as a postdoctoral researcher at the University of Stuttgart Visualization Research Center (VISUS) until 2016. His research focuses on methods for visualizing and comparing large and dynamic graphs and hierarchies, often in the context of software systems, their evolution, and execution behavior. He also investigates visual analytics systems and the integration of visualizations into text documents.