The value of data abstraction and transformation of provenance data for visual analysis

Francis Nguyen[1], Jude Shamsi[2], Joseph Wonsil[1], Shabab Khan[2], Margo Seltzer[1], and Tamara Munzner[1]

University of British Columbia, Vancouver B.C., Canada
[1]{frnguyen,jwonsil,mseltzer,tmm}@cs.ubc.ca
[2]{judems,skhan91}@student.ubc.ca

Most work in provenance visualization has focused on directly visualizing the graph structure inherent in provenance data [1, 3, 2, 6, 8]. However, this graph-centric visualization approach has clear limits when provenance graphs become very large, and it is not clear exactly what end-user tasks it enables.

We call for discussion of an alternative: rather than simply visually encoding the topological structure of the provenance graph itself, we advocate exploring novel use cases that provenance might facilitate, where both provenance and visualization are yoked together in service of a specific application goal. The information contained within provenance graphs could be transformed into new derived forms, to support tasks identified as necessary to target specific domain problems. This approach draws on the nested model of visualization [7], where domain-specific tasks and data are translated into domain-agnostic language at the *abstraction* level. The base data is often transformed to create new derived data that is more applicable to the underlying goals. The *visual encoding* of the transformed data is designed at the next level, which is computationally instantiated at the final *algorithm* level.

Our initial exploration of these ideas suggests that data provenance may be most helpful in visualization tools when it is used to derive further data in service of an end-user task. In other words, we advocate transforming the provenance into a representation that is semantically meaningful in the context of a specific end-user task and visually encoding that transformed data.

We have begun to investigate these ideas in two different problem domains: *intrusion detection* and *program comprehension*. Some intrusion detection systems use sophisticated analysis of provenance graphs to identify anomalous system behaviour [4], but mapping such an anomaly back to a root cause remains difficult. The data representation in which anomalies can be detected is several transformations removed from the original provenance, which is itself an abstract representation of system behavior. We use visualization to help users navigate between these different representations to connect anomalies to specific system actions. In the context of program comprehension, we use language level provenance to enhance code comprehension [5] with *data comprehension* by visually encoding the relationships between different data items and different versions of the same data item.

References

1. M. A. Borkin, C. S. Yeh, M. Boyd, P. Macko, K. Z. Gajos, M. Seltzer, and H. Pfister. Evaluation of filesystem provenance visualization tools. IEEE Trans. Visualization and Computer Graphics, 19(12):2476–2485, 2013. doi: 10.1109/TVCG.2013.155
2. S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In Proc. ACM SIGMOD Intl Conf. on Management of data (SIGMOD '08). 1345–1350, 2008. doi: 10.1145/1376616.1376772
3. H. A. Duru, M. P. Çakir, and V. Isler How does software visualization contribute to software comprehension? A grounded theory approach. International Journal Human Computer Interaction, 29:743-763, 2013.
4. X. Han, T. Pasquier, A. Bates, J.W. Mickens, and M.I. Seltzer. UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats. ArXiv, 2020. abs/2001.01525.
5. A.J. Ko and B.A. Myers. Designing the Whyline: a debugging interface for asking questions about program failures. ACM Conf. on Human Factors in Computing Systems (CHI 2004). 151-158, 2004.
6. B. Lerner and E. Boose. Rdatatracker: Collecting provenance in an interactive scripting environment. 6th USENIX Workshop on the Theory and Practice of Provenance, 2014.
7. T. Munzner. A Nested Model for Visualization Design and Validation. IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 2009), 15(6):921–928, 2009.
8. R. Yin and R. K. Keller. Program comprehension by visualization in contexts. Proc. International Conf. Software Maintenance, pp. 332–341, 2002. doi: 10.1109/ICSM.2002.1167789