I Don't Know What You Did Last Summer: The Missing Role of Humans in Systems Research

Thomas Davidson
MPI-SWS

Jonathan Mace
MPI-SWS

Abstract

Human-in-the-loop tools are a common fixture of systems research. Yet the human-facing aspects of these tools are frequently sidelined: in a survey of 822 recent systems papers, we find that the majority pay no attention and give no explanation to the human-facing components. We propose that the impact and reach of human-in-the-loop systems research can be greatly enhanced when exposition is given to the human-facing components, and propose a concrete checklist of steps for authors to realize this opportunity.

1 Introduction

Humans are an integral part of building, testing, deploying, operating, and troubleshooting computer systems. Modern systems are large, complex, and always-on. They have given rise to disciplines such as site reliability engineering and a blurring of boundaries between development and operations, with devops roles and on-call duties commonplace in industry.

For increasingly many research problems, humans thus play a pivotal role as *users* of the proposed tools and techniques. Consider, as a running example, distributed tracing. A key use case of this is debugging anomalous requests in distributed systems. While distributed tracing research primarily focuses on how to capture trace data across machines, taking the technical pieces in isolation elides a critical step: a human user is ultimately responsible for exploring and interpreting the recorded traces to intuit an anomaly's root cause [34].

User interfaces and visualizations thus determine how *effectively* humans can perform their tasks, of which many are time-critical and complex, and rely on humans because automation is intractable. Many tools produce complex data as output, and interpreting raw data is unintuitive and error prone. In distributed tracing, visualizations have repeatedly surfaced in the research literature for this reason, from the event timelines in Magpie [2], to span waterfalls in Dapper [40], to aggregations in Canopy [16] (cf. Figure 1).

However, visualization and user interaction goes unexplained and unpresented in the majority of systems research. Our earlier examples from distributed tracing are outliers. In §3 we survey 822 research papers published over the last 4 years at top systems conferences. We found that at least 8% of all papers relied on users, usually in the form of a human user interacting with system-produced output. However, only around 33% of these papers show an example, or explain the output their systems produce. Furthermore, only 14% of the papers requiring interaction from users give any motivation behind the design of their output. These figures would be easier to understand if there was a wealth of systems visualization literature, however, over the 4 years covered by our survey out of roughly 800 papers published at IEEE Vis (the premier visualization conference) only 8 held relevance to systems visualization problems.

This demonstrates a *missed opportunity* for systems research: practitioners seek *usable* tools and therefore *require* a solution for the UI and visualization components. Without explicit answers, the challenge is pushed to practitioners, raising the barrier for adoption and hindering potential impact. Within distributed tracing, prior research on trace comparison [35, 36] has influenced later features in open-source tools [38]; however, longstanding problems such as accessible visualization for non-experts [39] have received little attention and remain a major pain point for practitioners [21, 39].

The essence of the problem is missing exposition. Systems research often omits details of UI and visualization, even when those components exist. By doing so, it forgoes an opportunity to disseminate new problems and domain insights to a wider audience of visualization researchers and practitioners. A proof of concept, no matter how unrefined, can still have surprising impact: in our running example, *all* distributed tracing tools we know of base their visualizations on the span swimlanes first presented by Dapper [40].

In this paper we seek to answer how systems research should present the "interface" to human users – that is, the key concepts, interactions, and expectations that bridge human users with tools. This notion is analogous to programmatic APIs that separate concerns and define mutual expectations between independent software components. For human users, the details needed are often simple, such as prototype screenshots and interaction

overviews; yet they make UI and visualization requirements concrete, offer a problem definition, and highlight subtleties and unexpected challenges that can arise.

We outline our solution in §4: a checklist of 5 questions adapted from the "what, why, how" principle for designing and presenting visualizations [25]. In following the checklist, systems researchers methodically (yet concisely) produce a description of the interface to humans, the problem definition for the user-facing components, challenges already encountered, and insights into possible visualization approaches. We hope that by shining the spotlight on visualization and user interaction, we can increase the impact of systems research by extending its reach to a broader audience of practitioners, visualization researchers, and systems researchers.

2 Motivation

2.1 HL-Tools: Human-in-the-Loop Tools

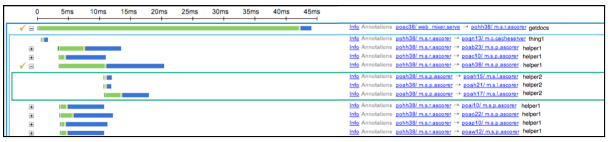
In this paper we are interested in how *human-in-the-loop* tools in systems research present their user-facing visualization and outputs. Our interest in this topic stems from a simple premise: if a user has to perform a non-trivial task – e.g. interpreting complex output – then designing the human-facing components of the tool is important and non-trivial.

Consider, for example, distributed tracing tools such as Dapper [40]. These tools aid operators in understanding and debugging problems in distributed systems. They produce *traces* of requests, comprising numerous logging statements and latency measurements, ordered into a directed acyclic graph. Human operators manually inspect individual traces, typically aided by a swimlane visualization (cf. Figure 1), to gain detailed insight into a request's execution.

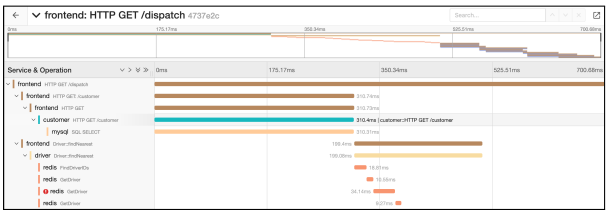
Other example topics include debugging, modelling, and performance analysis, at the application, operating system, and network level. More broadly, we consider any tools with the following characteristics. Firstly, the tools exist to aid human users in a high-level task. They produce output that a human must interpret and reason about; implicitly this is a non-trivial task for it not to be automated. Second, this output may be large, structured, and multi-dimensional; not just a simple line of output, but data that may be difficult to consume in its raw form. Moreover, users may need to correlate the data with other sources (e.g. a source code base), further compounding the task complexity.

We use the term HL-Tools to refer to tools that:

- produce complex, potentially multi-faceted data
- require humans to perform a non-trivial cognitive task, exacerbated by the complexity of the data
- cannot be automated fully, or automation cannot be fully trusted
- benefit when human users perform their task faster and more effectively



(a) Dapper [40] inspired by timelines of Magpie [2] introduced spans, an abstraction for RPC calls, and visualizes them in a hierarchical timeline.



(b) The open-source Jaeger [14] is inspired by Dapper (cf. 1a) and incorporates aspects of Canopy [16] (not pictured).

Figure 1: The canonical "swimlane" visualization found in distributed tracing tools has evolved across multiple works.

2.2 Opportunities

Human users are central to motivating HL-Tools, but thereafter research often focuses on the technical challenges to making the tool a reality. This is commonplace in systems research, as we show in §3, but not surprising: if we cannot overcome the technical challenges then the tool isn't possible in the first place. In the use cases introduced in §2.1, the research primarily focuses on those technical challenges. Distributed tracing, for example, focuses on *how* to causally relate information across machines, the instrumentation needed at the system level, and how to collect and reconstruct recorded information into traces.

The contribution of research that introduces an HL-Tool is therefore primarily demonstrating that such a tool is technically possible. Nonetheless, we argue that research should be careful not to lose sight of the human users of these tools. Concretely, there are three broader opportunities available to research that successfully 'closes the loop' of the HL-Tool back to its human users. Namely, doing so increases the potential impact, by extending the reach of research to practitioners, visualization researchers, and ultimately back to systems researchers.

Practitioners. Practitioners are often the intended audience and motivation for HL-Tools, and thus an important measure of success is whether research ideas proliferate into practice. For a tool to be feasible, it cannot omit the pieces that human users are expected to interact with.

Consider again distributed tracing. Open-source tools Jaeger [14] and Zipkin [46] both leverage designs from Dapper [40] (Figure 1), a tool developed by industrial researchers, where there is clear impetus for practical

tools. By contrast, X-Trace [9], an academic distributed tracing tool, only introduced its visualization in later experience reports [8, 33].

Research that "closes the loop" back to human users is more accessible to practitioners and has the potential for practical impact. The key to seizing this opportunity is to *reduce uncertainty* around what is needed to take a tool from research prototype to usable in practice. If the research does not discuss these pieces, then it introduces doubt about how feasible the tool might be and whether there exist unmentioned or unsolved challenges. On the other hand, research that does discuss user-facing outputs reduces this uncertainty, even if some challenges are unsolved. Screenshots or descriptions of prototype UIs serve both as evidence and as starting points for practitioners, even though these may only be byproducts of the research. Eliminating uncertainty increases the appeal to practitioners who may want to implement ideas. This is even the case for unsolved challenges.

Visualization Research. HL-Tools are human-in-the-loop solutions to systems problems and a natural fit for visualization research [25]. Visualization research can address common challenges shared by different kinds of data and HL-Tools. An example are *dashboards* that report metrics to system operators. Dashboards are commonplace in performance monitoring tools, and in their general form, *decision support* dashboards have received significant attention in the visualization literature [37].

In general, HL-Tools wrangle multi-dimensional, multi-faceted, and structured data – a difficult research challenge that receives attention across a wide range of application domains [17, 20, 41]. Of particular note are medical science and bioinformatics, where a focus on user interaction and visualization has spurred development and adoption of common visualization approaches [27, 29, 30], and so much research interest that dedicated visualization conferences now exist in this field [4]. Similarly, high-performance computing has received attention due to the importance of performance optimization and relatively homogeneous systems [13].

In contrast to these other application domains, HL-Tools receive relatively little direct attention in the visualization literature. In the past 4 years of IEEE Vis, the premier visualization conference, we found only 1 HL-Tool paper (CloudDet [43], centered on cloud performance anomalies) and a further 7 papers with broad similarities (4 software engineering [6, 12, 18, 24] and 3 high-performance computing [19, 26, 42]). Generally, we have observed that when HL-Tools are presented in visualization literature, it is often by the same group responsible for developing the original HL-Tool. For example, Spectroscope [35, 36], Pajé [5, 28] and ShiViz [3, 11] each have tandem contributions in systems and visualization.

We believe this observation demonstrates a missed opportunity for systems research. Interesting visualization challenges exist in HL-Tools, but by default, *only* systems researchers have visibility of those challenges. In the aforementioned examples, researchers were able to pursue visualization research because they were already familiar with the HL-Tool and application domain. For researchers outside of this expert core, a lack of domain expertise and a clear problem statement are *domain* and *abstraction threats* to be avoided [25]. Thus when systems researchers choose *not* to pursue visualization research questions, there is little chance that other researchers will pursue them either. Overall, this situation leads to point solutions for specific HL-Tools, but little technique-driven research that addresses commonalities across many tools.

To benefit from the attention of visualization research, HL-Tools must offer visibility of their challenges. For systems researchers this entails communicating details about the human-facing components. Counter-intuitively, there is substantial value when systems research acknowledges visualization and interaction challenges that it *hasn't* addressed, because this still "closes the loop" back to human users and exposes details, difficulties, and nuances. It highlights an extant need for further research and makes it possible for readers to extract common challenges that transcend individual tools. Lastly, it provides a starting point for visualization researchers to approach problems without requiring a priori application domain knowledge.

Systems Community The needs of human-users can also drive systems research into HL-Tools. In §2.1 we provided several examples of HL-Tool research. These examples have the potential to stem further research, such as that laid out in distributed tracing. These sorts of problems are often motivated by a need from practitioners who have adopted a tool. In general, HL-Tool research has flexibility in choosing the data to capture, and changing use cases can cause researchers to revisit these tools.

Beyond directly motivating new research directions, research that closes the loop back to users provides a foundation for subsequent work to both inherit and enhance. Our example in Figure 1, provides an example of this, where each work builds upon ideas presented in the previous. In areas such as metric dashboards, commonplace in industry today, systems researchers can assume basic functionality that is commonplace and provides a starting point. In many other areas there is a lack of these initial reference points.

Summary A common theme to all three opportunities is *exposition*. Practitioners, visualization researchers, and systems researchers benefit when research closes the loop between the HL-Tool and its human users. These

opportunities only become available when systems research provides details of how output from their tools is presented and how human users are expected to interact with the tool. By explicitly laying out any challenges presented by this, and providing solid details, researchers provide a starting point for future work and adoption in our three highlighted areas.

3 Survey of HL-Tools

In this section we conduct a survey of 822 research papers published between 2017-2020 at five premier systems conferences[1]. The survey design is inspired by the *"what, why, how?"* principle of visualization research [25]. The principle helps designers and researchers effectively present and explain their work by asking: what is shown to the user; why is it shown; and how is it shown. The questions are prompts for describing three important aspects of visualization research: data abstractions (what), task abstractions (why), and visual and interaction idioms (how). We are interested in these aspects: given a tool, can we ascertain the data abstractions relevant to the tool (what); the tasks expected of users (why); and lastly, ideas or justification for presenting data to users (how). Overall we map these aspects to five survey questions. The survey was conducted by the lead author, whose primary expertise is visualization, and cross-validated by three systems researchers[2]. Table 1 summarizes the results.

Q1: Does the paper present an HL-Tool? (Yes/No/Unclear) Of 822 papers surveyed, 8.4% of papers (69) meet the criteria for HL-Tools outlined in §2.1. 81.4% (669) do not present an HL-Tool and are not relevant to our survey; included in this are tools that integrate into some external system and effectively outsource all user interaction, e.g. reporting metrics to Prometheus [31]. The remaining 10.2% of papers (84) could not be categorized. The main reason a paper cannot be categorized is when it motivates or automates a problem for human users, but thereafter does not distinguish the ultimate role of human users or whether they perform a non-trivial task. For the rest of the survey, we present results only for the 69 papers that meet the criteria for HL-Tools; for the 85 unclear papers, the answer to all survey questions is "no".

Q2: Does the paper show a screenshot or a mock-up of the tool? (Yes/No/Partial) We ask this question preemptively because many papers use example screenshots as a vehicle to explain user-facing outputs. Of the 69 HL-Tool papers, 33.3% (23) include a screenshot or mock-up. 49.3% (34) do not include a screenshot or mock-up. We did not consider it sufficient to indicate the existence of a visualization (e.g. in an overview diagram) without providing further detail. 17.4% (12) do not include a screenshot or mock-up, but provide some depiction of the visualization beyond just its existence. For example, Seer [10] depicts stylized Gantt charts in its system overview diagram, thereby indicating a data abstraction used by its visualization and is marked as "partial". Papers that provide an overview diagram with no indication of the form of output, such as TrackIO [7] are marked as "no".

	Q1 (n=822)			Q2 (n=69)			Q3 (n=69)		Q4 (n=69)		Q5 (n=69)	
	Yes	Unclear	No	Yes	Partial	No	Yes	No	Yes	No	Yes	No
%	8.4	10.3	81.3	33.3	17.4	49.3	39.1	60.9	14.5	85.5	14.5	85.5

Table 1: Full Survey Results

Q3: Does the paper explain the user-facing output? ("*What*") (Yes/No) This asks whether papers describe the data abstractions output by the tool. A paper can satisfy this question regardless of including a screenshot; however, most papers make reference to a screenshot. We accepted any explanation provided: detailed description in the text body; a brief figure caption; or annotations directly in the screenshot. Most papers only provide one-line statements or figure captions. Several, such as tpprof and SelfStarter [44] [15], provide substantial detail. Of the 69 HL-Tool papers, 39.1% (27) explain the user-facing output and 60.9% (42) do not provide any explanation.

Q4: Does the paper motivate the user-facing output? ("*Why*") (Yes/No) This question ties the user-facing output back to the tasks the user is expected to perform. We look for an explanation of why the specific output is the right fit for the task at hand. For example, Rex [23] is a tool for preventing misconfigurations when developers update code but don't update related configuration files. The tool uses association rule mining, but developers struggled to interpret this output, so instead the tool presents concrete explanations and examples from prior code commits. In general we accepted any explanation provided, including anecdotal, intuitive, or empirical evidence. Of the 69 HL-Tool papers, 14.5% (10) motivate the user-facing output and the remaining 85.5% (59) provide no justification.

Q5: Does the paper design user-facing components? ("*How*") (Yes/No)

This question stretches the expectations of systems research, yet we found that several papers dedicate significant attention to explaining how an effective visualization could be implemented, by proposing and/or implementing visual idioms. For example, tpprof [44] discusses how visually aligning network state heat-maps with network state subsequences enables users to draw detailed conclusions about network traffic patterns. We accepted any explanation of the visual parameters used to portray the data. Of the 69 HL-Tool papers, 14.5% (10) provide an explanation and 85.5% (59) provide no explanation.

[1] ASPLOS, NSDI, OSDI, SIGCOMM, SOSP

[2] Scripts and results will be made available upon publication

3.1 Takeaways

Most HL-Tool research omits user-facing details. Screenshots, descriptions, and designs are absent for the majority of HL-Tool research beyond an initial problem motivation. Q1 identified a conservative lower bound of 8.4% of papers presenting HL-Tools; a further 10.2% of papers not included in subsequent survey questions *potentially* include human-facing components, but this wasn't clear from the paper alone.

Most HL-Tool descriptions focus on "what" and not "why". Most research papers focus on making a HL-Tool a reality, under the assumption that outputs can be made useful for users. Though they may provide screenshots and descriptions of *what* is provided as output (Q2, Q3), information is often missing about *why* the data is needed and how it maps to users' tasks (Q4).

Industry papers are more user-centric. Of the 69 HL-Tool papers, 32 were published by industrial research groups. These papers had significantly higher focus on user-facing details: 47% provide screenshots (Q2); 50% explain the user-facing output (Q3); and 19% motivate the user-facing output and design user-facing components(Q4, Q5). We expect that industrial researchers more directly draw from practical evidence and needs.

HL-Tools are Datacenter-centric. Several common themes emerge when we stratify our survey results by key topics. We observe that HL-Tools often target datacenters and clouds, they operate at the network and application level, and common challenges include monitoring, debugging, analysis and configuration. The prevalence of HL-Tools in these areas is likely due to increased complexity, "always-on" systems, and a need to touch live systems when investigating problems.

HL-Tools are on the rise. The number of HL-Tool papers published increases each year: 4.2% (10) in 2017; 5.1% (13) in 2018; 8.4% (19) in 2019; and 6.9% (27) in 2020. We attribute this growth to a continuing shift in software engineering practices towards a tighter integration of development and operations and larger, continuously-deployed systems.

Multi-modal Data is Commonplace. Combining data sources is commonplace; e.g. debugging, performance optimization, and configuration research often ties runtime measurements back to source code and use paths and walks through source code as data abstractions [22, 45]. Change over time is commonplace due to code changes, workload changes, and self-adaptation; differencing and comparison are common task abstractions. Many HL-Tools today remain myopic, yet live in a broader ecosystem; combining tools or relating information between tools remains an open challenge [1, 16]. We also found that bug-localization work [32] whilst repeatedly identifying the importance of an end-users interaction rarely addressed how this could be achieved. **These are all compelling future directions for HL-Tool visualization research.**

4 Closing the loop

Our goal with this paper is to nudge researchers towards more exposition of the human-facing components of their HL-Tools. In this section, we propose a concrete checklist for researchers to aid this task. The philosophy of our checklist is to overshare. We request only the details, and not qualitative arguments defending choices.

1. Explain the human user's role. State that a user is required for any part of the tool, even if: (1) it's off the critical path; (2) user interaction can be outsourced to some other system; or (3) it requires no changes from prior tools. If the work is not human-in-the-loop, but prior or related work is, state this distinction. If human users motivate the work (introduction, motivation), then clarify this role later (design, implementation).

2. Include a screenshot. State whether a visualization was implemented. If so, screenshots contextualize the work for all audiences and reduce the burden on textual descriptions. If space is a concern, defer to an appendix or link to examples in a webpage or repository. If screenshots aren't possible (e.g. for privacy concerns) consider a stylized mock-up.

3. Describe the user-facing output. Not all readers are domain experts, yet assuming knowledge was a common pitfall in our survey. Describe the data types output by the tool and the outside data sources required to use the tool (e.g. the source code base). Highlight differences between this work and prior or related work, such as subtle changes or additions to the output data. Avoid only an abstract specification: if a tool is designed to be flexible (e.g. *"Dapper supports a map of key-value annotations"* [40]) then describe the common uses (*"Programmers tend to use application-specific annotations as a kind of distributed debug log file"*), the scale of the output (*"70% of all Dapper spans and 90% of all Dapper traces have at least one application-specified annotation"*), and expected outliers (*"In many of the larger systems at Google, it is not uncommon to find traces with thousands of spans"*).

4. Motivate the output from the user's perspective. Explain why the tool's output was the 'right' output for the human user. This justification can be anecdotal, intuition, or empirical evidence. In many cases the output may already be justified by prior work, though this should be revisited for changes or additions to the user-facing output. If possible, decouple the justification from technical limitations of the tool.

5. Explain how you built it. If a visualization was implemented, describe any ideas or intuitions that went into it, such as visual motifs that users found compelling, or

established visualization design patterns that could be leveraged. Conversely, if the visualization was apparently trivial to implement, say so.

References

[1] D. Ardelean, A. Diwan, and C. Erdman. Performance analysis of cloud applications. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 405–417, 2018.

[2] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using magpie for request extraction and workload modelling. In *OSDI*, volume 4, pages 18–18, 2004.

[3] I. Beschastnikh, P. Liu, A. Xing, P. Wang, Y. Brun, and M. D. Ernst. Visualizing distributed system executions. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 29(2):1–38, 2020.

[4] The BioViz Interest Group. Retrieved January 2021 from http://biovis.net/.

[5] J. C. De Kergommeaux, B. Stein, and P.-E. Bernard. Pajé, an interactive visualization tool for tuning multi-threaded parallel applications. *Parallel Computing*, 26(10):1253–1274, 2000.

[6] S. Devkota, P. Aschwanden, A. Kunen, M. Legendre, and K. E. Isaacs. Ccnav: Understanding compiler optimizations in binary code. *IEEE transactions on visualization and computer graphics*, 2020.

[7] A. Dhekne, A. Chakraborty, K. Sundaresan, and S. Rangarajan. Trackio: tracking first responders inside-out. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 751–764, 2019.

[8] R. Fonseca, M. J. Freedman, and G. Porter. Experiences with tracing causality in networked services. *INM/WREN*, 10(10), 2010.

[9] R. Fonseca, G. Porter, R. H. Katz, and S. Shenker. X-trace: A pervasive network tracing framework. In *4th {USENIX} Symposium on Networked Systems Design & Implementation ({NSDI} 07)*, 2007.

[10] Y. Gan, Y. Zhang, K. Hu, D. Cheng, Y. He, M. Pancholi, and C. Delimitrou. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 19–33, 2019.

[11] S. Grant, H. Cech, and I. Beschastnikh. Inferring and asserting distributed system invariants. In *Proceedings of the 40th International Conference on Software Engineering*, pages 1149–1159, 2018.

[12] K. E. Isaacs and T. Gamblin. Preserving command line workflow for a package management system using ascii dag visualization. *IEEE transactions on visualization and computer graphics*, 25(9):2804–2820, 2018.

[13] K. E. Isaacs, A. Giménez, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz, B. Hamann, and P.-T. Bremer. State of the art of performance visualization. In *EuroVis (STARs)*, 2014.

[14] Jaeger: Open Source, End-to-End Distributed Tracing. Retrieved January 2021 from https://www.jaegertracing.io/.

[15] S. K. R. Kakarla, A. Tang, R. Beckett, K. Jayaraman, T. Millstein, Y. Tamir, and G. Varghese. Finding network misconfigurations by automatic template inference. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 999–1013, 2020.

[16] J. Kaldor, J. Mace, M. Bejda, E. Gao, W. Kuropatwa, J. O'Neill, K. W. Ong, B. Schaller, P. Shan, B. Viscomi, et al. Canopy: An end-to-end performance tracing and analysis system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 34–50, 2017.

[17] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):495–513, 2012.

[18] Y. Kim, J. Kim, H. Jeon, Y.-H. Kim, H. Song, B. Kim, and J. Seo. Githru: Visual analytics for understanding software development history through git metadata analysis. *arXiv preprint arXiv:2009.03115*, 2020.

[19] Z. Li, H. Menon, D. Maljovec, Y. Livnat, S. Liu, K. Mohror, P.-T. Bremer, and V. Pascucci. Spotsdc: Revealing the silent data corruption propagation in high-performance computing systems. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[20] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1249–1268, 2016.

[21] D. Luu. A simple way to get more value from tracing. Retrieved February 2021 from https://danluu.com/tracing-analytics/.

[22] J. Mace, R. Roelke, and R. Fonseca. Pivot Tracing: Dynamic Causal Monitoring for Distributed Systems. In *25th ACM Symposium on Operating Systems Principles (SOSP'15)*, 2015.

[23] S. Mehta, R. Bhagwan, R. Kumar, C. Bansal, C. Maddila, B. Ashok, S. Asthana, C. Bird, and A. Kumar. Rex: Preventing bugs and misconfiguration in large services using correlated change analysis. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 435–448, 2020.

[24] H. Mumtaz, S. Latif, F. Beck, and D. Weiskopf. Exploranative code quality documents. *IEEE transactions on visualization and computer graphics*, 26(1):1129–1139, 2019.

[25] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[26] H. T. P. Nguyen, A. Bhatele, N. Jain, S. Kesavan, H. Bhatia, T. Gamblin, K.-L. Ma, and P.-T. Bremer. Visualizing hierarchical performance profiles of parallel codes using callflow. *IEEE transactions on visualization and computer graphics*, 2019.

[27] S. I. O'Donoghue, B. F. Baldi, S. J. Clark, A. E. Darling, J. M. Hogan, S. Kaur, L. Maier-Hein, D. J. McCarthy, W. J. Moore, E. Stenau, et al. Visualization of biomedical data. *Annual Review of Biomedical Data Science*, 1:275–304, 2018.

[28] F.-G. Ottogalli, C. Labbé, V. Olive, B. de Oliveira Stein, J. C. de Kergommeaux, and J.-M. Vincent. Visualization of distributed applications for performance debugging. In *International conference on computational science*, pages 831–840. Springer, 2001.

[29] G. A. Pavlopoulos, D. Malliarakis, N. Papanikolaou, T. Theodosiou, A. J. Enright, and I. Iliopoulos. Visualizing genome and systems biology: technologies, tools, implementation techniques and trends, past, present and future. *Gigascience*, 4(1):s13742–015, 2015.

[30] B. Preim and C. P. Botha. *Visual computing for medicine: theory, algorithms, and applications*. Newnes, 2013.

[31] Prometheus - Monitoring System and Time-Series Database. Retrieved January 2021 from https://prometheus.io/.

[32] F. Ruffy, T. Wang, and A. Sivaraman. Gauntlet: Finding bugs in compilers for programmable packet processing. In *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, pages 683–699, 2020.

[33] R. R. Sambasivan, R. Fonseca, I. Shafer, and G. R. Ganger. So, you want to trace your distributed system? key design insights from years of practical experience. *Parallel Data Lab., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-PDL*, 14, 2014.

[34] R. R. Sambasivan, I. Shafer, J. Mace, B. H. Sigelman, R. Fonseca, and G. R. Ganger. Principled workflow-centric tracing of distributed systems. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 401–414, 2016.

[35] R. R. Sambasivan, I. Shafer, M. L. Mazurek, and G. R. Ganger. Visualizing request-flow comparison to aid performance diagnosis in distributed systems. *IEEE transactions on visualization and computer graphics*, 19(12):2466–2475, 2013.

[36] R. R. Sambasivan, A. X. Zheng, M. De Rosa, E. Krevat, S. Whitman, M. Stroucken, W. Wang, L. Xu, and G. R. Ganger. Diagnosing performance changes by comparing request flows. In *NSDI*, volume 5, pages 1–1, 2011.

[37] A. Sarikaya, M. Correll, L. Bartram, M. Tory, and D. Fisher. What do we talk about when we talk about dashboards? *IEEE transactions on visualization and computer graphics*, 25(1):682–692, 2018.

[38] Y. Shkuro. A Picture is Worth a 1,000 Traces. Retrieved February 2021 from https://www.shkuro.com/talks/2019-11-18-a-picture-is-worth-a-thousand-traces/.

[39] C. Shridharan. Distributed Tracing – we've been doing it wrong. Retrieved February 2021 from https://copyconstruct.medium.com/distributed-tracing-weve-been-doing-it-wrong-39fc92a857df.

[40] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. Dapper, a large-scale distributed systems tracing infrastructure. 2010.

[41] J. Wang, S. Hazarika, C. Li, and H.-W. Shen. Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2853–2872, 2018.

[42] K. Williams, A. Bigelow, and K. Isaacs. Visualizing a moving target: A design study on task parallel programs in the presence of evolving data and concerns. *IEEE transactions on visualization and computer graphics*, 26(1):1118–1128, 2019.

[43] K. Xu, Y. Wang, L. Yang, Y. Wang, B. Qiao, S. Qin, Y. Xu, H. Zhang, and H. Qu. Clouddet: Interactive visual analysis of anomalous performances in cloud computing systems. *IEEE transactions on visualization and computer graphics*, 26(1):1107–1117, 2019.

[44] N. Yaseen, J. Sonchack, and V. Liu. tpprof: A network traffic pattern profiler. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 1015–1030, 2020.

[45] Y. Zhang, S. Makarov, X. Ren, D. Lion, and D. Yuan. Pensieve: Non-intrusive failure reproduction for distributed systems using the event chaining approach. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 19–33, 2017.

[46] Zipkin: A Distributed Tracing System. Retrieved January 2021 from http://zipkin.io/.