See it to Believe it? The Role of Visualisation in Systems Research

Thomas Davidson
Jonathan Mace

Max Planck Institute for Software Systems (MPI-SWS)
Saarland Informatics Campus, Germany

ABSTRACT

A common fixture of computer systems research are *Visualisation-in-the-Loop Tools*: tools that produce complex output data and require a human user to interpret the data visually. However, systems research frequently omits or sidelines details of the visualisation components that were necessary for the tool. In a survey of 1,274 recent systems papers we find that at least 7.7% (98) of them present a visualisation-in-the-loop tool. We also find that the majority of these publications pay no attention and give little explanation to implemented visualisations. We propose that the impact and reach of visualisation-in-the-loop systems research can be greatly enhanced when exposition is given to visualisation, and propose a concrete checklist of steps for authors to realise this opportunity.

CCS CONCEPTS

• **General and reference** → **Surveys and overviews**; • **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

survey, human-in-the-loop, visualisation-in-the-loop, visualisation, computer systems

ACM Reference Format:
Thomas Davidson and Jonathan Mace. 2022. See it to Believe it? The Role of Visualisation in Systems Research. In *SoCC '22: ACM Symposium on Cloud Computing (SoCC '22), November 7–11, 2022, San Francisco, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3542929.3563488

SoCC '22, November 7–11, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9414-7/22/11.
https://doi.org/10.1145/3542929.3563488

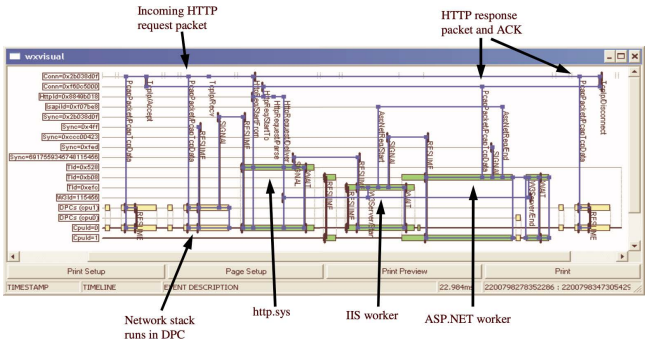
1 INTRODUCTION

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

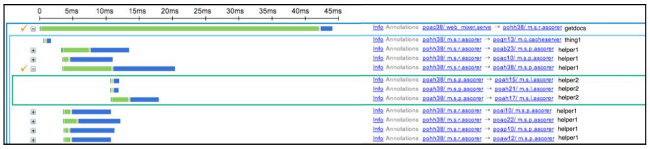
Visualisation-in-the-Loop Tools (VL-Tools) are thus important in practice and commonplace in the systems research literature as they inherently rely on human users to interact with and interpret output data from systems. VL-Tools are software tools to aid the developers and operators of computer systems. VL-Tools typically analyse a system's code, configuration, deployment, or runtime execution, to provide human users insight into the system's behavior and properties. They achieve this by producing complex data which must be interpreted by a human and is often presented in a custom visualisation component of the tool, to aid the analysis. Although other interaction modalities outside of visualisation do exist, visualisation is commonplace and thus we focus on VL-Tools.

Consider *distributed tracing* as a running example; we will revisit this example throughout the paper, but emphasise that the key aspects it highlights are prevalent in all forms of VL-Tools. A key use case of distributed tracing is debugging anomalous requests in distributed systems. While distributed tracing research primarily focuses on how to capture trace data across machines at runtime, taking the technical pieces in isolation elides a critical step: the recorded trace data is complex (an annotated directed acyclic graph); the data is presented to users through a graph or timeline visualisation; and the user is ultimately responsible for exploring and interpreting the visualisation to intuit any anomaly's root cause [42].

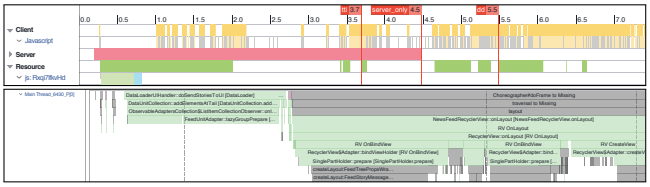
User interfaces and visualisations thus influence how *effective* a VL-Tool is. Human users are needed when tasks cannot be fully automated, and many tasks are time-critical and complex. Visualisation is enormously useful in this setting because interpreting raw data directly can be slow, unintuitive, and error prone. In distributed tracing, visualisations have repeatedly surfaced in the research literature for this reason,



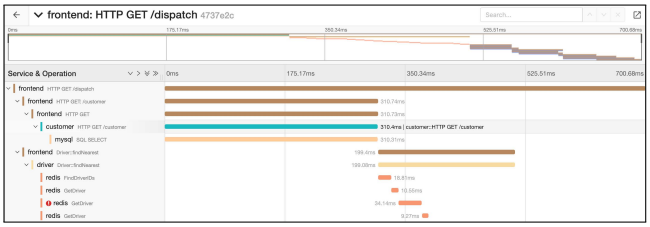
(a) Magpie [6] developed a timeline visualisation that displays resource events grouped by thread and highlights processing time of a request in each thread.



(b) Dapper [48] inspired by timelines of Magpie [6] introduced spans, an abstraction for RPC calls, and visualises them in a hierarchical timeline.



(c) Canopy [24] extends Dapper's swimlanes with flame graphs [18] and context aggregated across many requests.



(d) The open-source Jaeger [22] is inspired by Dapper (cf. 1b) and incorporates aspects of Canopy [24].

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

from the event timelines in Magpie [6], to span waterfalls in Dapper [48], to aggregations in Canopy [24] (cf. Figure 1).

Yet the distributed tracing example is atypical. Many systems research papers introduce tools with complex outputs, but leave the visualisation components and requirements unexplained, unpresented, or only implied. In §3 we survey 1,274 research papers published over the last 5 years at top systems conferences. We found that almost 8% of all papers described tools that relied on human user interpretation of complex outputs, well-suited to visualisation. However, less than 30% of these papers show an example, or explain how or if they visualised the output. Furthermore, only 17% of these papers motivate the design decisions for presenting the outputs. These results are not counterbalanced by systems publications in the visualisation research community either: over the 5 years covered by our survey out of roughly 950 papers published at IEEE VIS (the premier visualisation conference) only 8 held relevance to systems visualisation problems.

This demonstrates a *missed opportunity* for systems research: practitioners *require* a solution for the UI and visualisation components, without which interpreting complex data outputs is cumbersome and error prone. Yet without explicit consideration in the research literature, the challenge of designing and implementing visualisation components is pushed to practitioners, raising the barrier for adoption and hindering potential impact.

Within our example of distributed tracing, prior research on trace comparison [43, 44] has influenced later features in open-source tools [46]; however, long-standing problems such as accessible visualisation for non-experts [47] have received little attention and remain a major pain point for practitioners [30, 47].

The essence of the problem is missing exposition. Systems research often omits details of visualisation, even when these components exist. By doing so, it forgoes an opportunity to disseminate new problems and domain insights to a wider audience of visualisation 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 visualisations on the span swimlanes first presented by Dapper [48].

In this paper we show that systems research has historically under-presented the user-facing visualisations. The details that readers require are often simple, such as prototype screenshots; yet they make the visualisation 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 visualisations [34]. In following the checklist, systems researchers methodically and concisely produce a description of the user-facing visualisations, the problem definition for these components, challenges already encountered, and insights into possible future visualisation approaches. We hope that by shining the spotlight on visualisation we can increase the impact of systems research by extending its

reach to a broader audience of practitioners, visualisation researchers, and systems researchers.

2 MOTIVATION

2.1 VL-Tools: Visualisation-in-the-Loop Tools

VL-Tools are software tools that aid system developers and operators with some high-level task, and present output that those human users must interpret and reason about visually. In general, VL-Tools commonly analyse a system's code, configuration, deployment, or runtime execution; and their goal is often to provide users with actionable insight into the system's behavior and properties. VL-Tools produce output data that is large, structured, and multi-dimensional, and users benefit from having data visualisations rather than consuming the data in its raw from. Users may also need to correlate the output data with other sources (e.g. a source code base), further increasing the task complexity.

Concretely, we use the term VL-Tools to refer to tools that:

- produce complex, potentially multi-faceted data;
- produce data that is well-suited to visualisation;
- require humans to visually interpret the output data and make decisions based on it;
- cannot be automated fully, or automation cannot be fully trusted;
- and benefit when human users perform their task faster and more effectively.

Our interest in VL-Tools stems from a simple premise: if a user has to interpret complex data output by a tool and make decisions based on this interpretation, then *how* users consume the output – through UIs and visualisation – is of significant importance. Consider the following example areas and tools:

Distributed tracing tools such as Dapper [48]. 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 visualisation (cf. Figure 1), to gain detailed insight into a request's execution.

Network Profiling Tools such as tpprof[53] facilitate users in designing, understanding and optimising networks. These tools process and rank complex usage data in order to present traffic as composite graphs and heatmaps. This output is leveraged by operators in the implementation and maintenance of networks.

System Modeling Tools generate models of system behavior from profiling data (e.g. resource consumption models [52]), often incorporating static analysis of source code. Human users can inspect these models to aid system understanding and problem diagnosis (e.g. diagnosing scheduling problems).

Network Configuration Tools such as SelfStarter[23] collate data from multiple nodes and sources of varying formats(such as access control lists) and allow system operators to detect outliers. SelfStarter presents this information using a template structure which highlights anomalous discrepancies and allows the user to mitigate misconfigurations caused by a range of different factors.

Other topics where VL-Tools are prevalent include debugging, modelling, and performance analysis, at the application, operating system, and network level. An example of a tool that does not meet the criteria for a VL-Tool would be a command-line tool that produces basic or trivial output and doesn't require humans to perform a complex task. Likewise, many systems research topics may have usability concerns (e.g. research on high-level programming APIs) but do not meet the criteria for VL-Tools. Finally, while there exist interaction modalities outside of visualisation, these are not the focus of this work.

2.2 Opportunities

Users are central to motivating VL-Tools, but thereafter research often focuses on the technical challenges of 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 a VL-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 VL-Tool back to its human users and exposes how the complex output of the data is presented. Namely, doing so increases the potential impact, by extending the reach of research to practitioners, human computer interaction (HCI) and visualisation researchers, and ultimately back to systems researchers.

Practitioners. Practitioners are often the intended audience and motivation for VL-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 visualisation or user interface that human users are expected to interact with.

Consider again distributed tracing. Open-source tools Jaeger [22] and Zipkin [55] both leverage designs from Dapper [48] (Figure 1), a tool developed by industrial researchers, where there is clear impetus for practical tools and the authors explicitly present their visualisation approach for distributed tracing. By contrast, X-Trace [15], an academic distributed tracing tool, only introduced its visualisation in later experience reports [14, 41].

Research that 'closes the loop' of a VL-Tool 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 paper does not explicitly discuss the visualisation 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 visualisation reduces this uncertainty, even if some challenges are unsolved. Screenshots or descriptions of prototypes serve both as evidence and as starting points for practitioners, even though these may only be byproducts of the research.

For example, Figure 1 illustrates how the the standard swimlane diagram used by today's distributed tracing tools evolved from ideas presented by Magpie [6] and Dapper [48].

Eliminating uncertainty increases the appeal to practitioners who may want to implement ideas. This is even the case for unsolved challenges.

Visualisation Research. VL-Tools are human-in-the-loop solutions to systems problems which provide visualisation to their user and are therefore a natural fit for visualisation research [34]. Visualisation research can address common challenges shared by different kinds of data and VL-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 visualisation literature [45].

In general, VL-Tools wrangle multi-dimensional, multi-faceted, and structured data – a difficult research challenge that receives attention across a wide range of application domains [25, 29, 49]. Of particular note are medical science and bioinformatics, where a focus on user interaction and visualisation has spurred development and adoption of common visualisation approaches [36, 38, 39], and so much research interest that dedicated visualisation conferences now exist in this field [19]. Similarly, high-performance computing has received attention due to the importance of performance optimisation and relatively homogeneous systems [21].

In contrast to these other application domains, VL-Tools receive relatively little direct attention in the visualisation literature. In the past 5 years of IEEE VIS out of roughly 950 papers we found only 1 VL-Tool paper (CloudDet [51], centered on cloud performance anomalies) and a further 7 papers with broad similarities (4 software engineering [12, 20, 26, 33] and 3 high-performance computing [28, 35, 50]). Generally, we have observed that when VL-Tools are presented in visualisation literature, it is often by the same group responsible for developing the original VL-Tool. For example, Spectroscope [43, 44], Pajé [10, 37] and ShiViz [7, 17] each have tandem contributions in systems and visualisation.

We believe this observation demonstrates a missed opportunity for systems research. Interesting visualisation challenges exist in VL-Tools, but by default, *only* systems researchers have visibility of those challenges. In the aforementioned examples, researchers were able to pursue visualisation research because they were already familiar with the VL-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 [34]. Thus, when systems researchers choose *not* to pursue visualisation research questions, there is little chance that other researchers will pursue them either. Overall, this situation leads to point solutions for specific VL-Tools, but little technique-driven research that addresses commonalities across many tools.

To benefit from the attention of visualisation research, VL-Tools must offer visibility of their challenges. For systems researchers this entails communicating details about the human-facing components, and specifically how they present their complex output to users. Counter-intuitively, there is substantial value when systems research acknowledges visualisation 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 HCI and visualisation researchers to approach problems without requiring a priori application domain knowledge.

Systems Community. The needs of human-users can also drive systems research into VL-Tools. In §2.1 we provided several examples of VL-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, VL-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. 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.

3 SURVEY OF VL-TOOLS

In this section we conduct a survey of 1,274 research papers published between 2017-2021 at six premier systems conferences – ASPLOS, NSDI, OSDI, SIGCOMM, SoCC, SOSP. Combined these venues represent systems research broadly and cover a diverse range of topics.

The survey design is inspired by the *"what, why, how?"* principle of visualisation research [34]. 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 visualisation 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 visualisation, and cross-validated by three systems researchers. In order to mitigate the risks associated with a single person surveying over 1,000 papers we performed an extensive cross-validation of the results by selecting a random subset of 50 papers and providing them to three other researchers along with a guide on how to execute the survey. From this cross validation, 100% of the results matched with the lead author's. In order to allow for full scrutiny and further examination of the results we make the full result set available[1]. Table 1 summarises the results.

Q1: Does the paper present a VL-Tool?
Yes/No/Unclear

From the survey of 1,274 papers, we found that 7.7% (98) met the criteria for a VL-Tool: they describe tools with complex output that must be actioned by human users. A further 12 papers explicitly stated that their output took a non visual form and these are excluded from the further survey questions and revisited in §4.2. From our 98 VL-Tool papers, 43 made explicit reference to utilising visualisation, the remaining 55 discussed presenting complex output to the users, potentially in a visual form, but did not explicitly state its form. These 55 papers are included as they are highly likely to utilise visualisation. We touch on this lack of clarity in §3.1.

The remaining 1,164 papers were split into 1,039 (81.5%) that did not present a VL-Tool at all and so were not relevant to our survey. The non-relevant papers include tools that integrated into some external system and effectively outsource all user interaction, e.g. reporting metrics to Prometheus [5]. The remaining 125 (9.8%) of papers could not be categorised. The main reason for this would be that the work motivated a problem for human users, but thereafter did not provide any elaboration of the user's role. For the rest of the survey we do not consider the 125 'unclear' papers.

Ultimately, we completed the full survey on 98 VL-Tool papers that either explicitly utilised visualisation, or provided complex output to a user in a potentially visual form.

Q2: Does the paper show a screenshot or mock-up?
Yes/No/Partial

We ask this question preemptively because many papers use example screenshots as a vehicle to explain visualisation. Of the 98 VL-Tool papers, 28.6% (28) include a screenshot or mock-up.

59.2% (58) do not include a screenshot or mock-up. We did not consider it sufficient to indicate the existence of a visualisation (e.g. in an overview diagram) without providing further detail. 12.2% (12) do not include a screenshot or mock-up, but provide some depiction of the visualisation beyond just its existence. For example, Seer [16] depicts stylised Gantt charts in its system overview diagram (Figure 2), thereby indicating a data abstraction used by its visualisation and is marked as "partial". Papers that provide an overview diagram with no indication of the form of output, such as TrackIO [13] are marked as "no".

Q3: Does the paper explain the visualisation? ("*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

[1] https://gitlab.mpi-sws.org/cld/sysviz

	Q1 (n=1,274)			Q2 (n=98)			Q3 (n=98)		Q4 (n=98)		Q5 (n=98)	
	Yes	Unclear	No	Yes	Partial	No	Yes	No	Yes	No	Yes	No
%	7.7	9.8	81.5	28.6	12.2	59.2	37.8	62.2	17.4	82.6	12.2	87.8
Raw Count	98	125	1,039	28	12	58	37	61	17	81	12	86

Table 1: Full Survey Results

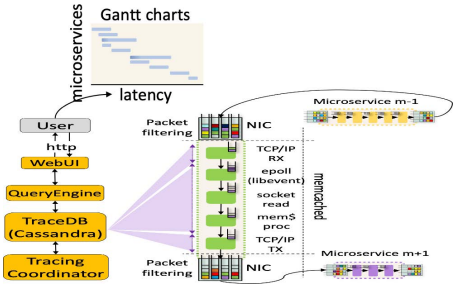


Figure 2: Seer [16] depicts stylised Gantt charts in its system overview diagram, indicating a data abstraction used by its visualisation

caption; or annotations directly in the screenshot. Most papers only provide one-line statements or figure captions. Several, such as tpprof and SelfStarter [53][23], provide substantial detail. Of the 98 VL-Tool papers, 37.8% (37) explain the user-facing output and 62.2% (61) do not provide any explanation.

Q4: Does the paper motivate the visualisation? (*"Why"*)

Yes/No

This question ties the visualisation 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 [32] 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 98 VL-Tool papers, 17.4% (17) motivate the visualisation and the remaining 82.6% (81) 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 visualisation could be implemented, by proposing and/or implementing visual idioms. For example, tpprof [53] 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 98 VL-Tool papers, 12.2% (12) provide an explanation and 87.8% (86) provide no explanation.

3.1 Takeaways

Most VL-Tool research omits visualisation details. Despite us identifying 110 papers (98 VL-Tool papers and the 12 non visualisation tool papers) that provided complex output to users, only 50% of these explicitly state the form of their output. In both work that does explicitly describe using visualisation, and work that implies it, screenshots, descriptions, and designs are absent for the majority beyond an initial problem motivation. Q1 identified a conservative lower bound of 7.7% of papers presenting VL-Tools; a further 9.6% of papers not included in subsequent survey questions *potentially also* include human-facing visualisation components, but this wasn't clear from the paper alone.

Most VL-Tool descriptions focus on "what" and not "why". Most research papers focus on making a VL-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).

Work that provides implementation details provides even more exposition. When VL-Tool research satisfies the later survey questions and provides motivation description of how the visualisation was implemented, they often satisfy the earlier questions too and provide significant exposition of their output. Of the 12 papers that satisfied question 5, 9 provide a screenshot, 11 explain their visualisation and 6 motivate their visual representation – significantly higher than the global average. We invite readers to carry out further analysis of our results by downloading the full dataset.[2]

[2]https://gitlab.mpi-sws.org/cld/sysviz

Industry papers are more user-centric. Of the 98 VL-Tool papers, 52 were published by industrial research groups. When these papers explicitly discuss having used visualisation (22 papers), they perform better than their non-industry counterparts (21 papers): 50% provide screenshots compared to 33%; 60% explain the visualisation (52% for non industry); 36% motivate their visualisation (19% non industry) and 18% describe their implementation (14% non industry). We expect that industrial researchers more directly draw from practical evidence and needs.

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

VL-Tools are on the rise. The number of VL-Tool papers published across these conferences has steadily increased since 2017: 15 in 2017; 18 in 2018; 23 in 2019; 29 in 2020 and 25 in 2021. 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 optimisation, and configuration research often ties runtime measurements back to source code and use paths and walks through source code as data abstractions [31, 54]. Change over time is commonplace due to code changes, workload changes, and self-adaptation; differencing and comparison are common task abstractions. Many VL-Tools today remain myopic, yet live in a broader ecosystem; combining tools or relating information between tools remains an open challenge [3, 24]. We also found that bug-localisation work [40] whilst repeatedly identifying the importance of an end-users interaction rarely addressed how this could be achieved. **These are all compelling future directions for VL-Tool visualisation research.**

Summary A common theme to all three opportunities is *exposition*. Practitioners, visualisation researchers, and systems researchers benefit when research closes the loop between the VL-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.

4 CLOSING THE LOOP

This paper's goal is to nudge researchers towards more exposition of the human-facing data outputs and visualisation components of their VL-Tools. This is not a panacea to the problem, but it is a good starting point. 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 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 visualisation was implemented. If so, screenshots contextualise 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 stylised mock-up.

3. Describe the data outputs and visualisation. Not all readers are domain experts, yet assuming knowledge was a common pitfall in our survey. Describe the data types the tool visualises 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 visualisation. Avoid only an abstract specification: if a tool is designed to be flexible (e.g. *"Dapper supports a map of key-value annotations"* [48]) 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 visualisation from the user's perspective. Explain why the tool's visual representation was the 'right' representation for the user. This justification can be anecdotal, intuition, or empirical evidence. In many cases the representation may already be justified by prior work, though this should be revisited for any changes or additions. If possible, decouple the justification from technical limitations of the tool.

5. Explain how you built it. If a visualisation was implemented, describe any ideas or intuitions that went into it, such as visual motifs that users found compelling, or established visualisation design patterns that could be leveraged. Conversely, if the visualisation was apparently trivial to implement, say so.

4.1 Above and Beyond

Our checklist is pragmatic and establishes what we believe is an easy baseline for VL-Tool research to achieve. We have not argued for visualisation components to judiciously adhere to visualisation design principles, nor that they be scientifically evaluated if visualisation is not the scientific focus of the paper. Exposition alone has substantial value.

Nonetheless, there are further steps that go above and beyond our proposed checklist. To achieve rigor in the presentation of visualisation components, the well-established *"what, why, how"* principle [34] from the visualisation research literature describes how to explicitly state what data the visualisation shows, why it shows that data, and how the data is presented in a visual form provides sufficient exposition for other interested researchers. A step further is to scientifically evaluate aspects of the visualisation, which can follow established principles for visualisation and UI evaluation [2, 27]. In this paper we have not proposed a general set of quantitative metrics for VL-Tools, but we hope that these emerge in the future as more VL-Tools incorporate scientific evaluation of visualisation components.

Several exemplar works demonstrate effective presentation of visualisation components. SelfStarter [23] is a VL-Tool to detect misconfigured networks; the paper provides clear explanations and examples of output, motivates the need for visualisation, and explains the visual techniques employed – without detracting from the paper's technical contributions. SecureCode [9] is a VL-Tool to detect risky infrastructure scripts; the paper presents a mostly textual visualisation, yet places the human user front and center, and gives clear exposition of the problem's impact on intended users. The gold standard of visualisation presentation is naturally found in the visualisation research community; exemplar works such as CloudDet [51] are featured in the *applications* track of conferences like IEEE VIS.

4.2 Beyond Visualisation

In this paper we have focused on VL-Tools, defined as tools that produce complex outputs and have human users who benefit from data visualisations. However, a broad range of systems research work involves human users in other contexts. A notable example are high-level programming APIs that aim to simplify programming for users, such as MapReduce [11], Spark [4], and TensorFlow [1]. Additionally, there exist commercial products that utilise non-visual modalities for interaction, such as smart speakers with no display which rely solely on audio interaction and output. Human-in-the-Loop tool research may, in the future, also expand and begin to utilise non-visual modalities too. Moreover, there already exists systems research, such as Net2Text [8], which produces purely textual complex output and relies on human interaction. We believe that the checklist and general concepts discussed in this paper can extend to other systems works that incorporate elements of HCI. Concretely, this work can form the basis for understanding how systems research should present the "interface" to human users – the key concepts, interactions and expectations bridging users to tools.

5 CONCLUSION

In this paper we described the important role of visualisation in systems research through *Visualisation-in-the-Loop Tools* (VL-Tools). We advocated the benefits of effective documentation and presentation of visualisation components, both for the systems community and to a wider audience (§2). We surveyed 1,274 research papers (§3) published at systems conferences over the last 5 years and found that a significant proportion describe VL-Tools that rely on humans to interpret complex data outputs, with implicit or explicit visualisation components. We found that less than 40% of these works included any form of example output, example visualisation, or explanation thereof. Moreover, we found that less than 20% of papers surveyed motivated the design or described the implementation of visualisation components.

To address this, we have advocated that an ideal VL-Tool publication would provide example screenshots of visualisations, explain the data outputs and presentation, motivate the visual representation, and explain how the visualisation was implemented. We provide a checklist built on our survey to aid future researchers in presenting their user-facing tools to facilitate the most benefit and exposition of their work. Several options exist to enable this exposition without impinging on the limited page real-estate; for example, inclusion in appendices; expanded technical report versions of the paper; and even dual-publications in systems and visualisation venues.

Exposition is a low barrier to entry for systems researchers. It is only a step towards addressing the problems raised in this paper, yet it is a valuable step: a *lack* of exposition only serves to exacerbate those problems. By including visualisation in the presentation of VL-Tools, papers elevate the human user and highlight their importance to the work. We believe that this first step can serve as a stepping stone to increase the reach of systems research and ultimately help establish meaningful collaborations between systems and visualisation researchers and communities.

REFERENCES

[1] Martín Abadi. 2016. TensorFlow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*. 1–1.

[2] Wolfgang Aigner, Stephan Hoffmann, and Alexander Rind. 2013. Evalbench: A software library for visualization evaluation. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 41–50.

[3] Dan Ardelean, Amer Diwan, and Chandra Erdman. 2018. Performance analysis of cloud applications. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 405–417.

[4] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. 2015. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1383–1394.

[5] Prometheus Authors. 2015. Prometheus - Monitoring System and Time-Series Database. Retrieved January 2021 from https://prometheus.io/.

[6] Paul Barham, Austin Donnelly, Rebecca Isaacs, and Richard Mortier. 2004. Using Magpie for request extraction and workload modelling.. In *OSDI*, Vol. 4. 18–18.

[7] Ivan Beschastnikh, Perry Liu, Albert Xing, Patty Wang, Yuriy Brun, and Michael D Ernst. 2020. Visualizing distributed system executions. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 29, 2 (2020), 1–38.

[8] Rüdiger Birkner, Dana Drachsler-Cohen, Laurent Vanbever, and Martin Vechev. 2018. Net2Text: query-guided summarization of network forwarding behaviors. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 609–623.

[9] Ting Dai, Alexei Karve, Grzegorz Koper, and Sai Zeng. 2020. Automatically detecting risky scripts in infrastructure code. In *Proceedings of the 11th ACM Symposium on Cloud Computing*. 358–371.

[10] J Chassin De Kergommeaux, Benhur Stein, and Pierre-Eric Bernard. 2000. Pajé, an interactive visualization tool for tuning multi-threaded parallel applications. *Parallel Comput.* 26, 10 (2000), 1253–1274.

[11] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (2008), 107–113.

[12] Sabin Devkota, Pascal Aschwanden, Adam Kunen, Matthew Legendre, and Katherine E Isaacs. 2020. CcNav: Understanding Compiler Optimizations in Binary Code. *IEEE transactions on visualization and computer graphics* (2020).

[13] Ashutosh Dhekne, Ayon Chakraborty, Karthikeyan Sundaresan, and Sampath Rangarajan. 2019. TrackIO: tracking first responders inside-out. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*. 751–764.

[14] Rodrigo Fonseca, Michael J Freedman, and George Porter. 2010. Experiences with Tracing Causality in Networked Services. *INM/WREN* 10, 10 (2010).

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

[16] Yu Gan, Yanqi Zhang, Kelvin Hu, Dailun Cheng, Yuan He, Meghna Pancholi, and Christina Delimitrou. 2019. 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*. 19–33.

[17] Stewart Grant, Hendrik Cech, and Ivan Beschastnikh. 2018. Inferring and asserting distributed system invariants. In *Proceedings of the 40th International Conference on Software Engineering*. 1149–1159.

[18] Brendan Gregg. 2017. Flame Graphs. Retrieved January 2021 from http://www.brendangregg.com/flamegraphs.html.

[19] BioViz Interest Group. 2022. BioVis Conference. Retrieved September 2022 from http://biovis.net/.

[20] Katherine E Isaacs and Todd Gamblin. 2018. Preserving command line workflow for a package management system using ASCII DAG visualization. *IEEE transactions on visualization and computer graphics* 25, 9 (2018), 2804–2820.

[21] Katherine E Isaacs, Alfredo Giménez, Ilir Jusufi, Todd Gamblin, Abhinav Bhatele, Martin Schulz, Bernd Hamann, and Peer-Timo Bremer. 2014. State of the Art of Performance Visualization.. In *EuroVis (STARs)*.

[22] Jaeger 2017. Jaeger: Open Source, End-to-End Distributed Tracing. Retrieved January 2021 from https://www.jaegertracing.io/.

[23] Siva Kesava Reddy Kakarla, Alan Tang, Ryan Beckett, Karthick Jayaraman, Todd Millstein, Yuval Tamir, and George Varghese. 2020. Finding Network Misconfigurations by Automatic Template Inference. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*. 999–1013.

[24] Jonathan Kaldor, Jonathan Mace, Michał Bejda, Edison Gao, Wiktor Kuropatwa, Joe O'Neill, Kian Win Ong, Bill Schaller, Pingjia Shan, Brendan Viscomi, et al. 2017. Canopy: An end-to-end performance tracing and analysis system. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 34–50.

[25] Johannes Kehrer and Helwig Hauser. 2012. Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey. *IEEE Transactions on Visualization and Computer Graphics* 19, 3 (2012), 495–513.

[26] Youngtaek Kim, Jaeyoung Kim, Hyeon Jeon, Young-Ho Kim, Hyunjoo Song, Bohyoung Kim, and Jinwook Seo. 2020. Githru: Visual Analytics for Understanding Software Development History Through Git Metadata Analysis. *arXiv preprint arXiv:2009.03115* (2020).

[27] Heidi Lam, Enrico Bertini, Petra Isenberg, Catherine Plaisant, and Sheelagh Carpendale. 2011. Empirical studies in information visualization: Seven scenarios. *IEEE transactions on visualization and computer graphics* 18, 9 (2011), 1520–1536.

[28] Zhimin Li, Harshitha Menon, Dan Maljovec, Yarden Livnat, Shusen Liu, Kathryn Mohror, Peer-Timo Bremer, and Valerio Pascucci. 2020. SpotSDC: Revealing the Silent Data Corruption Propagation in High-performance Computing Systems. *IEEE Transactions on Visualization and Computer Graphics* (2020).

[29] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. 2016. Visualizing High-Dimensional Data: Advances in the Past Decade. *IEEE Transactions on Visualization and Computer Graphics* 23, 3 (2016), 1249–1268.

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

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

[32] Sonu Mehta, Ranjita Bhagwan, Rahul Kumar, Chetan Bansal, Chandra Maddila, B Ashok, Sumit Asthana, Christian Bird, and Aditya Kumar. 2020. Rex: Preventing bugs and misconfiguration in large services using correlated change analysis. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*. 435–448.

[33] Haris Mumtaz, Shahid Latif, Fabian Beck, and Daniel Weiskopf. 2019. Exploranative code quality documents. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 1129–1139.

[34] Tamara Munzner. 2014. *Visualization analysis and design*. CRC press.

[35] Huu Tan Pham Nguyen, Abhinav Bhatele, Nikhil Jain, Suraj Kesavan, Harsh Bhatia, Todd Gamblin, Kwan-Liu Ma, and Peer-Timo Bremer. 2019. Visualizing hierarchical performance profiles of parallel codes using callflow. *IEEE transactions on visualization and computer graphics* (2019).

[36] Seán I O'Donoghue, Benedetta Frida Baldi, Susan J Clark, Aaron E Darling, James M Hogan, Sandeep Kaur, Lena Maier-Hein, Davis J McCarthy, William J Moore, Esther Stenau, et al. 2018. Visualization of biomedical data. *Annual Review of Biomedical Data Science* 1 (2018), 275–304.

[37] F-G Ottogalli, Cyril Labbé, Vincent Olive, Benhur de Oliveira Stein, J Chassin de Kergommeaux, and J-M Vincent. 2001. Visualization of Distributed Applications for Performance Debugging. In *International conference on computational science*. Springer, 831–840.

[38] Georgios A Pavlopoulos, Dimitris Malliarakis, Nikolas Papanikolaou, Theodosis Theodosiou, Anton J Enright, and Ioannis Iliopoulos. 2015. Visualizing genome and systems biology: technologies, tools, implementation techniques and trends, past, present and future. *Gigascience* 4, 1 (2015), s13742–015.

[39] Bernhard Preim and Charl P Botha. 2013. *Visual computing for medicine: theory, algorithms, and applications*. Newnes.

[40] Fabian Ruffy, Tao Wang, and Anirudh Sivaraman. 2020. Gauntlet: Finding Bugs in Compilers for Programmable Packet Processing. In *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*. 683–699.

[41] Raja R Sambasivan, Rodrigo Fonseca, Ilari Shafer, and Gregory R Ganger. 2014. 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).

[42] Raja R Sambasivan, Ilari Shafer, Jonathan Mace, Benjamin H Sigelman, Rodrigo Fonseca, and Gregory R Ganger. 2016. Principled workflow-centric tracing of distributed systems. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*. 401–414.

[43] Raja R Sambasivan, Ilari Shafer, Michelle L Mazurek, and Gregory R Ganger. 2013. Visualizing request-flow comparison to aid performance diagnosis in distributed systems. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2466–2475.

[44] Raja R Sambasivan, Alice X Zheng, Michael De Rosa, Elie Krevat, Spencer Whitman, Michael Stroucken, William Wang, Lianghong Xu, and Gregory R Ganger. 2011. Diagnosing Performance Changes by Comparing Request Flows.. In *NSDI*, Vol. 5. 1–1.

[45] Alper Sarikaya, Michael Correll, Lyn Bartram, Melanie Tory, and Danyel Fisher. 2018. What do we talk about when we talk about dashboards? *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 682–692.

[46] Yuri Shkuro. 2019. 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/.

[47] Cindy Shridharan. 2019. Distributed Tracing – we've been doing it wrong. Retrieved February 2021 from https://copyconstruct.medium.com/distributed-tracing-weve-been-doing-it-wrong-39fc92a857df.

[48] Benjamin H Sigelman, Luiz Andre Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan, and Chandan Shanbhag. 2010. Dapper, a large-scale distributed systems tracing infrastructure. (2010).

[49] Junpeng Wang, Subhashis Hazarika, Cheng Li, and Han-Wei Shen. 2018. Visualization and Visual Analysis of Ensemble Data: A Survey. *IEEE Transactions on Visualization and Computer Graphics* 25, 9 (2018), 2853–2872.

[50] Katy Williams, Alex Bigelow, and Kate Isaacs. 2019. 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 (2019), 1118–1128.

[51] Ke Xu, Yun Wang, Leni Yang, Yifang Wang, Bo Qiao, Si Qin, Yong Xu, Haidong Zhang, and Huamin Qu. 2019. Clouddet: Interactive visual analysis of anomalous performances in cloud computing systems. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 1107–1117.

[52] Suli Yang, Jing Liu, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. 2018. Principled schedulability analysis for distributed storage systems using thread architecture models. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*. 161–176.

[53] Nofel Yaseen, John Sonchack, and Vincent Liu. 2020. tpprof: A Network Traffic Pattern Profiler. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*. 1015–1030.

[54] Yongle Zhang, Serguei Makarov, Xiang Ren, David Lion, and Ding Yuan. 2017. Pensieve: Non-intrusive failure reproduction for distributed systems using the event chaining approach. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 19–33.

[55] Zipkin 2016. Zipkin: A Distributed Tracing System. Retrieved January 2021 from http://zipkin.io/.