SAND2022-10939C

Enabling Catalyst Adoption in SPARC

1st V. Gregory Weirs
Computational Science
Sandia National Laboratories
Albuquerque, NM USA
vgweirs@sandia.gov

2nd Elaine M. Raybourn
Applied Cognitive Science
Sandia National Laboratories
Albuquerque, NM USA
emraybo@sandia.gov

3rd Reed Milewicz
Software Engineering & Research
Sandia National Laboratories
Albuquerque, NM USA
rmilewi@sandia.gov

4th Killian Muollo
Software Engineering & Research
Sandia National Laboratories
Albuquerque, NM USA
kmuollo@sandia.gov

5th Jeffrey A. Mauldin
Scientific Apps & User Support
Sandia National Laboratories
Albuquerque, NM USA
jamauld@sandia.gov

6th Thomas J. Otahal
Scientific Apps & User Support
Sandia National Laboratories
Albuquerque, NM USA
tjotaha@sandia.gov

***Abstract*—This paper reports on Catalyst usability and initial adoption by SPARC analysts. The use case driven approach highlights the analysts' perspective. Impediments to adoption can be due deficiencies in software capabilities, but analysts identify many mundane inconveniences and barriers that prevent them from fully leveraging Catalyst. With that said, for many analyst tasks Catalyst provides enough relative advantage that they have begun applying it in their production work, and recognize the potential for it to solve problems they currently struggle with. The findings in this report include specific issues and minor bugs in Paraview python scripting, which are viewed as having straightforward solutions, and a broader adoption analysis.**

***Index Terms*—visualization, analysis, in situ, usability, adoption**

I. Introduction

Almost every widely used product starts as an idea, goes through a stage of development and prototyping to demonstrate the capability or utility, and finally, goes through another stage of development to make a usable product. The demonstration phase addresses *utility*: the capability addresses a need, and if they exist, potentially better than the alternatives. The third stage, often called "the valley of death," focuses on usability and adoption. *Usability* means that a user of the product can realize the capability provided by the product. A user might say a product is "too difficult to use" or "too complicated to understand," to complete their task to indicate barriers to usability. Even if a product provides utility and is usable, it has to fit in the environment of the user. Barriers to *adoption* include, e.g, metric wrenches when the user has nuts and bolts in imperial units, or software that does not run on the operating system of the user. Viewed broadly, the lack of utility and usability are barriers to adoption. Of course, many ideas start the process but far fewer become widely used products.

In situ analysis and visualization for engineering and science simulations on High Performance Computing (HPC) platforms are in the demonstration phase. The primary capability is that by generating analysis and visualization results as the simulation is running, one avoids write large simulation data files to disc. On state of the art HPC platforms, file I/O is relatively expensive and reducing the necessary I/O is highly desirable. A secondary and perhaps under appreciated benefit is automation. If the analysis and visualization steps are known in advance and can be completed as part of the simulation, analysts don't have to postprocess the data their simulation wrote to files. Uncertainty quantification and sensitivity analysis frequently involve large ensembles of calculations, and automated analysis and visualization of the results is crucial, even when postprocessing.

This paper reports our progress on the adoption phase for Catalyst in SPARC. Sandia Parallel Aerosciences Research Code (SPARC) is an engineering simulation code for fluid dynamics, principally the flow around vehicles re-entering the earth's atmosphere [1]. Catalyst [2] is the in situ version of the Paraview visualization application [3], [4]. The primary user interface for Paraview is a Graphical User Interface (GUI) and after reading simulation data files produced by SPARC, the user interactively applies operations to the data to produce images. The user interface for Catalyst is a Paraview python script: Paraview commands are expressed in python syntax and accessible as python modules. The general workflow, as recommended by Kitware (the developer of Paraview and Catalyst), is to generate the images desired in Paraview from an existing set of simulation data files. Then a Paraview python script can be generated from the GUI. This script can be interpreted by Catalyst to generate images when invoked by SPARC. We focus on this workflow and the Paraview python scripts. Of particular interest is understanding and editing the scripts, because analysts want to generate an initial script on a small simulation and use a modified script for a similar but much more expensive simulation. We did not examine the usability of the Paraview GUI or SPARC, except where they affect the usability of Catalyst. We are also interested in the adoption of Catalyst and Paraview for quantitative analysis, but in this paper we only examine visualization.

Paraview is a mature software application and its large, established user base demonstrates its adoption. Catalyst has essentially the same utility, so we focus on Catalyst usability

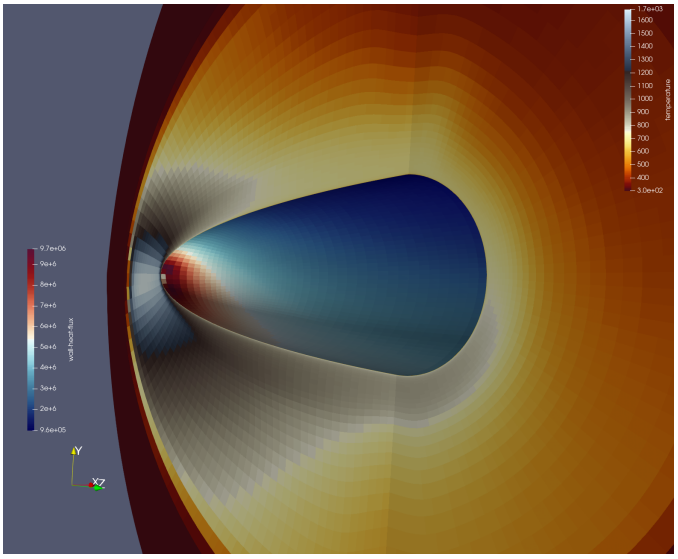

Fig. 1. Use Case 1: Corner view of flowfield temperature on symmetry and exit planes and heat transfer on the vehicle surface.

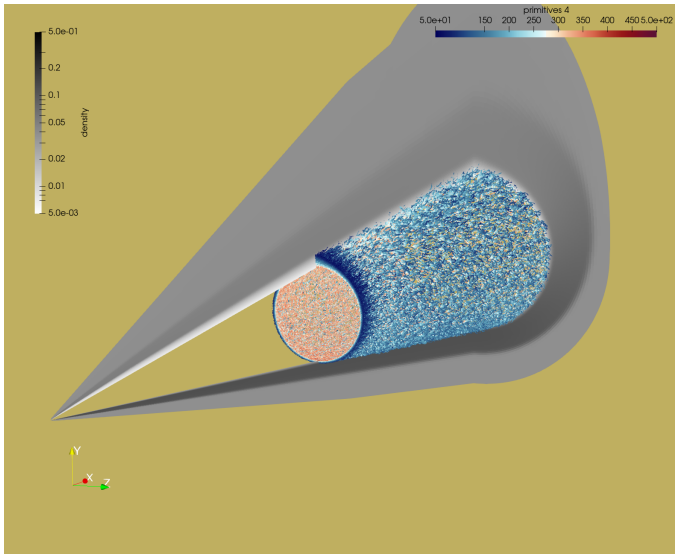

Fig. 2. Use Case 2: Isocontour of Q-criterion colored by temperature from LES, with steady state density (log scale) on symmetry and exit planes.

and adoption. We chose an analyst driven, use case approach. Use cases are typical tasks an analyst (Catalyst user) would like to accomplish.

A. Use Cases

Use cases are concrete examples of analysts' tasks. Two image generation use cases were designed to identify specific usability barriers encountered while executing an in-situ visualization workflow leveraging Paraview, Catalyst, and SPARC.

The simulation for the first use case models the flow over a blunt-nosed, axisymmetric vehicle traveling at Mach 5 at sea level, at 10^o angle of attack (AoA). SPARC iteratively solves for the steady-state, turbulent flowfield. This simulation is inexpensive, the amount of data written is small, and while images are produced throughout the simulation, only those at the end of the simulation are needed. However, it includes all the realism needed for investigating Catalyst usability and adoption. Images of the flowfield temperature are produced on the symmetry and exit planes of the simulation. A third image shows the heat flux on the vehicle surface, and the fourth image, shown in Fig. 1, includes both planes and the surface. SPARC analysts regularly produce images like these as a first look to confirm the simulation ran as expected, and then to identify any important flowfield structures.

The second use case is a large eddy simulation (LES) of the flow over a cone at Mach 8 at 6^o AoA. This simulation runs on 100 nodes of an ASC CTS-1 machine; it is a modest calculation by current LES and supercomputing standards, but is large enough that analysts limit how frequently they write the flowfield solution to disk. In Fig. 2, a contour surface of the Q-criterion identifies resolved vortical structures, and coloring the contour surface by temperature is useful for understanding heat transfer to the cone surface. The symmetry and exit planes display the gas density from a different simulation, stored in files and read by Catalyst during the LES simulation.

II. Code Integration and Support

A. Catalyst Integration with IOSS

The Input Output SubSystem (IOSS) library is part of the Sandia National Laboratories (SNL) Sandia Engineering Analysis Code Access System (SEACAS). Many SNL simulation codes, such as SPARC, Nalu, and Sierra Adagio, use IOSS to output SNL Exodus II format files and CFD General Notation System (CGNS) files in parallel. The central abstraction presented by IOSS is a database that encapsulates details about how mesh data and associated variables are written out by a simulation code. IOSS is similar to the SENSEI in-situ framework [6], in that it provides an interface between the simulation and multiple analysis targets. The Catalyst integration in IOSS implements two new IOSS database types, one for structured mesh output to Catalyst like CGNS and one for unstructured mesh output like Exodus II.

Rather than writing to a set of files in parallel, the Catalyst IOSS databases convert the data to equivalent Visualization ToolKit (VTK) data structures, and then send the mesh data to Catalyst in parallel. Catalyst runs a Python script that produces images, data extracts, and analysis products. Catalyst runs in-situ, using the same Message Passing Interface (MPI) ranks as the simulation code.

Simulation codes that use IOSS for writing simulation data to files require minimal changes to send that data to Catalyst through IOSS. Furthermore, the Catalyst integration can be tested independently of the simulation code by using test drivers that feed input mesh data from files through the test suite.

Catalyst is a large code library that has varied build requirements on different HPC platforms. For this reason, the Catalyst IOSS databases use a split implementation. The IOSS side provides an interface between the simulation and the Catalyst IOSS database types. The Catalyst side resides

in a dynamically loadable plugin library that converts the mesh data to VTK data structures, and then calls Catalyst. Simulation code build systems do not have to link against Catalyst libraries directly, which allows the Catalyst portion to be built, tested, and deployed separately.

B. SPARC Integration with Catalyst IOSS

SPARC must access the Catalyst IOSS database types. To enhance compatibility with SPARC analyst experience using file based I/O to Exodus II and CGNS files, input deck controls for Catalyst are implemented with a syntax parallel to the file based controls. New Catalyst input deck blocks use existing syntax for output frequency scheduling, variable selection, and output database type. They also contain new syntax for Catalyst specific requirements, such as Python script name, and multiple mesh inputs to the same Catalyst script.

The Catalyst IOSS plugin dynamic library must be built, tested, and deployed on all HPC platforms where SPARC is supported. Project developers employ the DevOps tools Ansible-Tower, Ansible, CDash, and CTest to meet this requirement. To give analysts early access to new features in SPARC and Catalyst, "advanced feature builds" are deployed on some HPC platforms. These advanced feature builds facilitated the use case studies described below.

In-situ technology is a new capability for analysts. A robust DevOps process lets us identify and fix issues and redeploy rapidly, which is crucial for analyst adoption.

C. Catalyst Integration Considerations

Based on our Catalyst SPARC integration experience, some important factors to consider when integrating Catalyst with a simulation code are:

- How to control Catalyst from the simulation input deck.
- How to link Catalyst to the simulation (dynamic run-time library or direct link).
- Performance impact on the simulation, for example [5].
- Transfering simulation mesh data structures to VTK mesh data structures.
- DevOps infrastructure to build, deploy, and test Catalyst with simulation.

III. Methodology

A major challenge in the design and development of visualization tools is ensuring that the product will be adopted by real-world users and be practicable for their needs. As noted by Fisher et al., a common stumbling block for the visualization and visual analytics community is that tool developers are often "not very consistent in articulating and applying methodological principles for system design and end-user evaluation" [10]. Insufficient rigor can lead to misunderstanding users' real needs or designing systems to solve the wrong problems. For that reason, a key goal of our effort has been to engage with prospective users directly and develop a systematic understanding of their needs.

A. Theoretical Framework: Five Characteristics of Innovation

In this study, we applied the Diffusion of Innovations (DOI) as a theoretical lens to investigate analysts' experiences with the Catalyst integration capability [8]. DOI theory, first pioneered by communication theorist and sociologist Everett Rogers in 1962, seeks to explain how and why innovative ideas and technologies are adopted and spread throughout a social system. In this approach, how an innovation is perceived by prospective adopters depends on different dimensions of perceived utility; as explained by Rogers, "The characteristics of innovations, as perceived by individuals, help to explain their different rate of adoption" [9]. These characteristics found on pages 15-16 include

- **Relative Advantage**: The degree to which an innovation is perceived as better than the idea it supersedes.
- **Compatibility**: The degree to which an innovation is perceived as being consistent with the existing values, past experiences, and needs of potential adopters.
- **Complexity**: The degree to which an innovation is perceived as difficult to understand and use.
- **Trialability**: The degree to which an innovation may be experimented with on a limited basis.
- **Observability**: The degree to which the results of an innovation are visible to others.

According to Rogers on page 16-17, "Innovations that are perceived by individuals as having greater relative advantage, compatibility, trialability, observability, and less complexity will be adopted more rapidly than other innovations." p.16 [9]. "Given that an innovation exists, communication must take place if the innovation is to spread" p. 17 [9]. What makes DOI theory particularly useful is that it helps in navigating the trade-offs between different dimensions of utility. For example, the Catalyst integration capability may promote relative advantage through code proximity (the ability of the visualizer to provide easy and fast access to underlying source code, see [7]), but in doing so this may also increase complexity until certain barriers to adoption are addressed.

Thus DOI theory enables researchers to explore the interplay of different innovation adoption drivers. Along these same lines, DOI theory has previously been used to study user adoption of data visualization and analytics tools in diverse contexts including healthcare [12], education [14], construction [13], and geospatial analytics [18].

B. Walkthrough and Content Analysis

Our subject matter expert (SME) developed two image generation use case workflows using Paraview, Catalyst, and SPARC. The workflow steps constituted instructions to be followed by analysts during a guided walkthrough interview in which the SME observed the analysts' actions, answered questions, provided explanations, and responded to feedback. The SME also performed a technical analysis, in which bugs and usability issues were identified, while determining the steps of the workflow. We conducted three interviews, averaging 60 - 90 minutes per interview. In each interview, the

SME generated a workflow that the users followed to perform certain tasks. During the walkthrough, users verbalized what they were thinking, asked questions, and provided feedback. We employed content analysis to analyze the video, audio, and transcripts of the use case walkthroughs. Content analysis is a methodology used to identify patterns in texts and can be quantitative, qualitative, or both [15]. Since our UUA hypothesis points to Paraview user adoption of data manipulation as a key abstraction, we used Rogers' five characteristics of innovations as categories to quantitatively code analysts' statements from the interview transcripts. Six project team members representing diverse scientific and engineering domains independently coded each of the interviews by watching the videos and matching the audio to the transcripts.

IV. Results and Discussion

A. Technical Findings

Technical findings are defined in the present paper as specific issues or recommendations identified by the SME or interviewees for Paraview, Catalyst, and the Paraview python scripting interface.

Paraview has different classes of objects. *Filters* are processing units assembled into a pipeline. The output of each filter may be displayed, as defined by its *representation*. A *view* composes the image from the active representations it contains. *Extractors*, or extracts, were added in version 5.10.0, and describe file output; data extractors send the output of a filter, and image extracts send the image in a View.

The scripts generated by the Paraview GUI are verbose. There are many lines specifying the properties of representations and views, and many of them are not necessary to include. Either the user has not changed the property values from their default values, or that property was not needed to make the image. Many of these properties are set by the user indirectly, such as through a mouse or widget interaction, and the associated property is not familiar to the analyst by the name in the script. Consequently the analyst may be uncertain about which representation and view properties are necessary, so they are hesitant to edit or delete them.

The scripts generated by the Paraview GUI embed specificity of the dataset in a way that is difficult for an analyst to identify. For example, in the color lookup table, the data value and the corresponding 3-tuple defining the color are just appended together in a python list, with no structure to distinguish the value from the tuple or one value-tuple pair from another. To rescale the color table to different minimum and maximum values, an analyst would have to recognize the structure from the values in the list, then recompute each data value and edit the list appropriately. It would be much easier to just specify the color table for a value range of, say, 0 to 1, and have Catalyst compute the corresponding values given user-specified minimum and maximum values, as the analyst would do in the Paraview GUI. There are many places in the script like this, though the color lookup table is particularly difficult to edit. As a reminder, easy modification of the scripts to apply to a different dataset is crucial, because the benefit of in situ analysis is only realized when applying it to simulations that have not yet been run.

TABLE I
Coded frequencies of DOI characteristics used in the qualitative analysis of the Flowfields and LES use cases.

Innovation Characteristics	Flowfields Use Case	LES Use Case
Relative Advantage	24	7
Compatibility	51	26
Trialability	33	9
Observability	6	2
Complexity	41	11
Total Number of Codes Applied	155	55

Another issue is that some properties recorded in the Catalyst script are not intuitive. For example, the view properties that specify the camera's location and orientation are easy to identify, but it is difficult for an analyst to know how to adjust the values to get the updated image they desire. In Paraview, these parameters are accesible but rarely displayed – instead, the analyst updates the view by using the mouse, seeing the effects but not the actual values of the camera properties.

The scripts generated by the GUI are best thought of as "journal" files that record the necessary information to regenerate the output image or data file, rather than a clear description that leads to easy understanding and adaptation to a different data set. Fortunately, the issues identified in the GUI-generated scripts have straightforward solutions, and we are working developers at Kitware to address them.

B. Usability and Adoption Findings

Usability and adoption findings were developed by performing a content analysis on the walkthroughs with analysts. Two hundred and ten unique statements made by three analysts during the interviews were identified across the image generation use cases. Each statement was then categorized into only one of the five characteristics of innovations by each coder [15]. Across the six coders, we used Cohen's Kappa [16], [17] to measure pairwise intercoder reliability (0.2238 average unadjusted). Intercoder reliability was calculated by taking the average pairwise kappa among the six coders based on whether or not they coded a statement, irrespective of *how* they coded the statement. A common discrepancy among coders was that while some coders labeled each statement at the beginning, others labeled statements at the middle or end. Therefore to compensate for this irregularity, we adjusted the kappa by grouping statements with those before and after (0.3276 adjusted). Additionally, while three coders were aligned in their coding schema (0.3641 unadjusted, 0.4181 adjusted) suggesting moderate agreement, others were not as well-aligned, resulting in lower kappa scores overall for the six coders due to the misalignment. Agreement among six coders was fair (0.21 - 0.40). Kappa scores were lower than desirable, but still within acceptable limits.

We employed Friedman's test to determine whether any of the five innovation characteristics were consistently labeled

more frequently by different coders [19]. This test yielded a p-value of 0.00027, indicating that there are innovation characteristics consistently rated higher than others across coders. Next, we used the post-hoc Friedman-Nemenyi test to measure which pairs of innovation characteristics were significantly more or less frequent relative to each other [20]. The test revealed that Relative Advantage is consistently coded more frequently than Observability (p=0.036466), and Compatibility more frequently than Observability (p=0.00100).

Table I illustrates the number of unique statements (210) that were coded in each of the characteristics of innovations categories. Note that fewer analysts' statements were coded for the chararcteristic Observability, suggesting that there were fewer instances of observability in these use cases. Additionally, in each use case, analysts' statements offered support for relative advantage with respect to data manipulation through testimonials such as, *"Yeah, so I can see myself using this immediately for some of our cavity works that we work on."* and *"Yeah, it seems quite useful right out of the box."*

While not depicted in Table I, 48 usability barriers were identifed from 41 analysts' statements. In seven of the analysts' statements, more than one coder identified different usability barriers for the same statement. Our findings indicate that some analysts' statements corroborated the technical analysis performed by the SME. For example, with respect to auxiliary objects (e.g., color lookup tables) being impractical to modify directly, analysts' statements in both the flowfield and LES image generation use cases addressed this finding. When asked by the SME, *"Do you think you could write that part yourself without the GUI?"* an analyst performing the workflow steps for the flowfield use case stated, *"We didn't change the color map for this but if we had to I don't know that I would be able to do that by hand."* While another added, *"I would say there's a difference, right, between editing a script that's there, and creating it completely from scratch."* When asked by the SME *"What else do you see in here or what do you think about the process we went through? ... Is this tractable?"* an analyst executing the steps for the LES use case stated, *"Some components of it, I think you're right where it would be more valuable to be able to dig into the python script."* It was observed that there appeared to be a lack of intuitiveness of GUI visual elements in the Catalyst script, and in the case of LES, understanding, editing, or modifying image-specific parts of the Catalyst script was difficult.

With respect to adoption, the most salient finding from applying Rogers' Innovation Characteristics to our use cases was the comparitively lower frequency of the innovation characteristic of Observability. That is to say, analysts were not as familiar with the capability afforded by the use of the Catalyst script to perform certain in situ analyses as they were with the Paraview GUI (see Table I) perhaps largely because they are not aware of others employing a Catalyst-centric approach. This is understandable, as the use case interviews may have been the first time some of the analysts had attempted these workflows.

Our research was exploratory in nature, and as such prone to certain limitations. For example, the findings of the present study are specific to the particular uses cases and the dataset was small. Intercoder agreement should also improve as we engage in further data collection and align our coding practices.

V. Conclusion

Catalyst in SPARC is entering the adoption stage for analysts in a production environment. Building on several years of effort, Catalyst is now integrated into SPARC in its parsing, simulation data transfer, and broader software engineering processes. This foundation permits developers to give more attention to usability and adoption.

We found that in practice, the Catalyst scripts generated by the Paraview GUI are journal files – they are effective for repeating an existing analysis, but they are difficult for Catalyst users to understand and modify for new analyses. For image generation tasks, the scripts are verbose and information specific to the original dataset is embedded in such a way that it is hard to find and modify. However, we believe straightforward solutions to these issues can be found and implemented, and are working with Kitware to that end. For some information in the scripts, such as the specification of camera properties and color tables, the Paraview GUI or other tools may provide the best mechanism for the analyst to specify what to include in their Catalyst script.

Through our UUA analysis, with respect to compatibility (one of Rogers' five characteristics of innovation), we concluded that analysts in our walkthroughs focused on determining how Catalyst would fit in their existing workflows. With respect to observability, we are using these preliminary results to guide our efforts to build a user community. We have deployed the workflow recipes on the SPARC Analysts' wiki to address some of the barriers identified by users such as the need for documentation. While our UUA results are exploratory, they have given us insights into the process of addressing usability barriers and technology adoption by directly engaging with the analysts for whom Catalyst was designed.

Acknowledgments

The authors greatly appreciate the engagement, questions, and feedback of the analysts that participated in our walkthroughs: Jeff Fike, Derek Dinzl, Eric Robertson, Nate Miller, Cory Stack, Stefan Domino, Ross Wagnild, and Brian Carnes.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] M. Howard, A. Bradley, S. W. Bova, J. Overfelt, R. Wagnild, D. Dinzl, M. Hoemmen and A. Klinvex. Towards Performance Portability in a Compressible CFD Code, AIAA 2017-4407. 23rd AIAA Computational Fluid Dynamics Conference. June 2017.

[2] U. Ayachit, A. Bauer, B. Geveci, P. O'Leary, K. Moreland, N. Fabian, and J. Mauldin. 2015. ParaView Catalyst: Enabling In Situ Data Analysis and Visualization. In Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV2015). Association for Computing Machinery, New York, NY, USA, 25–29. https://doi.org/10.1145/2828612.2828624

[3] J. Ahrens, B. Geveci, and C. Law, ParaView: An End-User Tool for Large Data Visualization, Visualization Handbook, Elsevier, 2005, ISBN-13: 978-0123875822

[4] U. Ayachit, The ParaView Guide: A Parallel Visualization Application, Kitware, 2015, ISBN 978-1930934306

[5] J. A. Mauldin, T. J. Otahal, A. M. Agelastos, and S. P. Domino, In-situ visualization for the large scale computing initiative milestone, In Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV '19). Association for Computing Machinery, New York, NY, USA, 1–5, 2019. https://doi.org/10.1145/3364228.3364229

[6] U. Ayachit et al., "The SENSEI Generic In Situ Interface," 2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV), pp. 40-44, 2016. doi: 10.1109/ISAV.2016.013.

[7] H. M. Keinle and H. A. Muller, "Requirements of software visualization tools: A literature survey," 2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, pp. 2-9, 2007.

[8] K. K. Kapoor, Y. K. Dwivedi, and M. D. Williams, "Rogers' innovation adoption attributes: A systematic review and synthesis of existing research," Information Systems Management, Taylor & Francis: vol. 2. pp. 74-91, 2014.

[9] E. M. Rogers, "Diffusion of innovations, 4th ed., New York: Simon and Schuster, 1995.

[10] L. A. McNamara, J. Scholtz, B. Fisher, J. G. Gomez, "What to do about those pesky users?," Albuquerque, NM: Sandia National Laboratories, 2015.

[11] R. Milewicz, and P. Rodeghero, "Position Paper: Towards Usability as a First-Class Quality of HPC Scientific Software," IEEE/ACM 14th International Workshop on Software Engineering for Science (SE4Science), pp. 41-42, 2019.

[12] D. Dolezel and A. McLeod, "Big data analytics in healthcare: Investigating the diffusion of innovation," Perspectives in health information management, vol. 16, Summer, American Health Information Management Association, 2019.

[13] P. Gholizadeh, behzad, E., and P. Goodrum, "Diffusion of building information modeling functions in the construction industry," Journal of Management in Engineering, vol. 34, no. 2, American Society of Civil Engineers, 2018.

[14] E. Habib, Y. Ma, D. Williams, H. O. Sharif, and F. Hossain, "HydroViz: design and evaluation of a Web-based tool for improving hydrology education," Hydrology and Earth System Sciences, vol. 16, no. 10, Copernicus GmbH, pp. 3767-3781, 2012.

[15] W. M. K. Trochim and J. P. Donnelly, "The research methods knowledge base, 3rd. ed., Mason, OH: Thomson, 2007.

[16] J. Cohen, A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1), pp. 37–46, 1960. https://doi.org/10.1177/001316446002000104

[17] J. R. Landis and G. G. Koch, The measurement of observer agreement for categorical data. Biometrics, 33(1), pp. 159-174, 1977. https://www.jstor.org/stable/2529310

[18] C. T. Smith, "Adoption and Implementation of Participatory GIS Technologies in Resource Management Networks: A Study of the US National Trails System," North Carolina: North Carolina State University, 2017.

[19] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," Journal of the American Statistical Association, vol. 32, no. 200, pp. 675-701. JSTOR 2279372, 1937. doi:10.1080/01621459.1937.10503522

[20] P.B. Nemenyi. "Distribution-free Multiple Comparisons," PhD thesis, Princeton, NJ: Princeton University, 1963.