A SURVEY OF VISUALIZATION CAPABILITIES FOR SIMULATION ENVIRONMENTS

Bruno St-Aubin
Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, CANADA

Fernando Loor

Departamento de Computación
Universidad Nacional de San Luis
Ejército de Los Andes 950
San Luis, ARGENTINA

ABSTRACT

Simulation visualization is an effective way to understand and communicate the results of simulation studies on complex systems and processes. Among other advantages, it increases model transparency and intelligibility for all categories of users including non-experts and it can be used by modelers as a tool to debug models during development. However, simulation visualization is an uncommon subfield of research in the simulation community and, although most commercial simulation software have visualization capabilities, they are generally tightly coupled to their software. We present the results of a survey of visualization capabilities for 50 diverse simulation environments, both academic and commercial, open source and proprietary and for a variety of application domains, discussing their main advantages and gaps.

Keywords: Simulation visualization, simulation environments.

1 INTRODUCTION

Visualization is an effective way of understanding and communicating complex systems and processes. It increases intelligibility, it enhances the tractability of data, processes, or theories for all categories of users, from the layperson to the scientist expert in the field. For example, it has been argued and shown that, in the history of twentieth century physics, a higher visualizability contributed significantly to a better understanding of concepts such as atomic structures or quantum field theory (de Regt 2014; de Regt and Parker 2014). In the field of surgical medicine, quantitative research has shown that visualization improves learning and performance in the execution of surgical procedures (Malas et al. 2018). In the context of simulation, visualization is often the first step in the analysis of results and the decision-making process that follows. (Collins and Ball 2013) argue that most people, besides model developers, only see a simulation through the visualization of its results. They even argue that the visualization of the results is what matters most to those outside of the modeling and simulation (M&S) community. In other words, presentation of the results is more important than the simulation itself. Their reasoning is that, for external stakeholders, visualization and analysis is a way to verify and validate the results. It is, therefore, a way of increasing transparency in M&S. Within the M&S community, visualization can be a valuable tool to support the verification and validation of models through comparison and animation (Sargent 2008; Bell and O'Keefe 1994) and to communicate results (Collins and Ball 2013). It can be used as an additional way of checking model problems and therefore becomes another tool in the model development cycle.

Although the usefulness of visualization for simulation has been recognized in the field since its early years (Hurrion 1978), it still generally takes second stage to the advancement of theory, research on performance and domain specific model development. In (Collins, Ball, and Romberger 2015b), the authors argue that research mostly focuses on the mechanics that make up the simulation and that, since visualization does not directly affect the simulation, it is seen as a secondary consideration. They further mention that, for some, visualization can even be treated as a development annoyance that is a necessary step before presenting "proper" statistical results. They support their argument by noting that the two most used textbooks in the

field contained virtually no discussion on visualization (Banks 1998; Law 2015). To the best of our knowledge, this fact still holds true to this day for many other textbooks (Zeigler, Praehofer, and Kim 2000; Fujimoto 2000). Other books dedicate an occasional chapter to the topic (Sokolowski and Banks 2010). However, they remain few and far between.

As it currently stands, simulation experts typically rely on visualization mechanisms that are specifically built for and tailored to their application domain, or they adopt *ad hoc* visualization. For example, modelers will write data processing and visualization scripts to transform their simulation results into chart-based analytics. For simulation tools focused on formal models (mainly developed in academic contexts) the situation is worse, and visualization capabilities tend to be basic or completely absent. This is likely due to the resources involved in developing comprehensive visualization and the lack of interest from the research community regarding this topic (Collins, Ball, and Romberger 2015a). For example, *CD++* (Wainer 2002), *Cadmium* (Belloli et al. 2019), *DEVSJAVA* (Sarjoughian and Zeigler 1998) and *ADEVS* are simulators that can be used to simulate a range of systems using the Discrete-Event Systems Specification formalism (DEVS) (Zeigler, Praehofer, and Kim 2000), but that only offer basic visualization and analysis capabilities. Other simulators for formal models, such as *OpenModelica* offer a more extensive capability (2D charts, 3D visualization and additional options to display analytical charts (Höger et al. 2012; Eriksson et al. 2008)). Commercially available simulation software tends to fare much better regarding visualization and analysis. The downside to these monolithic software applications is that users become restricted to the tools provided by that software manufacturer.

Whether simulation software is open-sourced or proprietary, meant for use-case specific or generic simulation scenarios, the visualization capabilities of simulation software are usually tightly coupled to the simulator. To the best of our knowledge, no simulation software uses any kind of standard or specification to format their outputs. They usually either do not provide an actual output file, employ software specific format, or allow the user to define their own output format. Visualization decoupled from the simulator is generally not a consideration in the development of simulation software. Therefore, simulation visualization engines are impossible to reuse across simulators.

In this paper, we present a survey of visualization capabilities in simulation software to establish an overview of the current state of visualization in the field of simulation. We do this to highlight the role and importance of visualization and simulation and to evaluate the impact that reusable visualization tools could have on the field, and to provide an opportunity for discussion for potential visualization standards, which can be achieved by decoupling the visualization from the simulator.

2 VISUALIZATION AND ANALYSIS IN SIMULATION

Within the field of simulation, visualization has generated a limited amount of research. Most often, research on visualization techniques occurs in the context of simulation studies in application domains. In (Kuljis, Paul, and Chen 2001) the CLINSim discrete-event simulator and associated visualization platform was used to model queues in hospitals. The visualization platform uses specific icons to show different actors, colors to show their states, vector graphics for rooms, etc. Visualization was important to improve wait times since it allowed non-experts to understand the impact of their decisions. There are many other domain-specific examples of simulation visualization. CAPSIS is an open-source software designed to model and simulate forest growth modeling (Dufour-Kowalski et al. 2011) using a variety of analytical charts, 2D and 3D spatial representations. In microbiology, (Zoellner et al. 2018) built a framework to follow microbial contamination of produce across supply chains. The framework relies on differential equations specific to the domain and the tool does not represent supply chains visually. Instead, it provides a series of analytical features to users such as line charts, bar charts, heatmaps, etc. These examples are representative of visualization platforms strongly coupled to a simulator and meant to cater the needs of a specific domain.

Tightly coupled, application specific visualizations provide a non-negligible advantage over generic visualization. The representation of results is tailored to the domain of application and therefore, assumptions can be made about the data to simplify the development of the visualization platform. In (Seide, Jensen, and Kieser 2021) the authors use radar charts to visualize highly dimensional results of drug trials in medical treatment simulations using a use case specific *R* script to process and plot their data. Some

assumptions made here are that the data will always be quantitative, continuous, and multi-dimensional. In (Busaman et al. 2015), custom-built visualization based on finite volume method and adaptive tree grids was used to visualize rainwater overland flow. Their result is a tightly coupled combination of simulation models, algorithms, result processing, and visualization for a specific use case. It can safely be assumed that the data is spatial with a continuous coverage. Likewise, (Schyndel et al. 2016) focuses on a pedestrian crowd model for indoor environments. Citing concerns about communicating results from the modelers to the domain experts, they propose the development of a web-based 3D visualization platform to highlight their results. The assumption here is that the model will represent pedestrian movement in a building: the visualization relies on a complex 3D architectural model that cannot be replaced easily. As can be seen from these examples, constraining the application domain of the simulation makes it easier to build rich visualizations, the trade-off being that they cannot be used for other application domains. Such examples are abundant in literature on the topic.

Research on the theoretical and philosophical aspects of simulation visualization is more uncommon. (Macal 2001) wrote, in an introduction to a special issue on visualization, that "visualization offers one of the most promising means to convey information from a simulation model to decision-makers in a meaningful way". He noted that, up to that time, research focused mainly on five broad categories: animation from simulation results, visual interactive simulation, visual interfaces for modeling, combination of visualization and simulation for decision-making and virtual simulated environments. The author further remarked that a clear gap existed in this subfield of simulation since the only special issue on the topic, at the time, dated back to 1987 (SIMULATION 1987). In the past few years however, research on the philosophical and theoretical considerations of the role of visualization and analysis in simulation has become more common. Collins et al. have been particularly fruitful regarding this topic (Collins, Ball, and Romberger 2015a; Knowles Ball and Collins 2012; Collins, Knowles Ball, and Romberger 2015b). In (Collins, Ball, and Romberger 2014), they highlight the importance and role of visualization for simulation by writing that "ultimately, the acceptance of complexity models within mainstream science and society will depend on the results that are produced visually". They also remark that visualization is a communication mechanism for non-expert users, that it can shine light on salient patterns of a system, particularly when they are non-linear, that it can easily identify non-converging simulations through analytical charts, etc. (Vernon-Bido, Collins, and Sokolowski 2015) discusses advantages of visualization: a deeper insight into the model, a support for debugging them, improved model quality, support for model validation, improved decision-making, etc.

Finally, there is a body of research that warns about the pitfalls of visualization for simulation. (Collins, Knowles Ball, and Romberger 2015b) discusses the seeming opposition between the fact that visualization is a secondary concern for modelers while at the same time often being the only part of a simulation that decision-makers use. Therefore, visualization can disproportionately influence any decision that results from it. It can also lead to several documented pitfalls, one of the better explored topics related to visualization. They also identify four ways in which visualization can potentially mislead the interpretation of simulation results. Those are: inclusion of extraneous elements, inclusion of baseless endogenous elements (visual fluff), inaccurate interpretation (due to a disconnect between representation and model), and accessibility (such as color-blindness or other impairments). In (Banks and Chwif 2011), the authors review numerous aspects of M&S and offers warnings to take into consideration. They discuss data collection, model building, verification and validation, analysis, etc. They also suggest that a visualization should provide a general view of the model and be organized according to well-defined criteria. Similarly to (Roman 2005), they also warn readers to not get confused by fancy graphics that may be misleading or dishonest. They note that visualization can support validation, increase the acceptance of a model by decision-makers and to increase sales.

3 QUALIFYING VISUALIZATION CAPABILITIES

Authors have discussed other aspects of simulation visualization. In (Wenzel and Jessen 2001), an approach to map discrete-event system models onto 3D animations is proposed. They establish a semantic body of rules to describe graphic usage for simulation representation in the context of facility lifecycles. They also note improved communication and collaboration, as well as reduced project times as benefits of a proper visualization platform. (Muller and Schumann 2003) present a review of visualization methods for time

dependent data, such as simulation results. They focus on characterizing analytical approaches to presenting simulation results, and on a taxonomy for the time axis as well as visualization techniques including charts, dynamic representations for animated reconstruction of the simulation and, event-based visualization.

(Vernon-Bido, Collins, and Sokolowski 2015) presents the following four types of visualization:

- Concept and diagram visualization rely on conceptual models, flowcharts, or other diagrams to reconstruct a simulation.
- Quantitative visualization relies on analytical charts and graphs to summarize or otherwise present a global analysis of the simulation.
- Seek and find visualization allows users to manipulate data while the visualization occurs. This usually takes the form of a graphic user interface where users can affect the simulation.
- Pattern and flow visualization focus on the interaction of data elements, usually through animated graphics.

It can be argued that the seek-and-find visualization type is more of a simulator issue rather than a visualization issue. Indeed, conducting this type of visualization requires an interactive simulator that allows a model to be changed at runtime, while simulation is executing. Therefore, once a model is rebuilt and simulated anew, results could be output following the same specification and therefore, changes would be reflected in any visualization, provided the platform is made to support such a change.

(Van Tendeloo and Vangheluwe 2017) evaluates existing DEVS simulation tools, and they review 7 academic tools and a single proprietary, commercial tool. They still provide 5 clear stages of visualization for discrete event-based simulators:

1. Identification of models and when they are triggered.
2. Visualization of a model's state at any given time.
3. Visualization of messages exchanged between models.
4. Identification of a model's internal and external transitions.
5. Visualization of the sequence of exchanged messages.

For each simulation environment we survey in the next section, we will indicate whether they support concept and diagram visualization (diagram) or quantitate visualization (charts). For DEVS simulation environments, we also indicate which of the 5 stages of visualization are met.

4 VISUALIZATION CAPABILITIES FOR SIMULATION ENVIRONMENTS

We selected 50 such tools to illustrate the heterogeneous state of simulation visualization and the widespread challenges that exist within that field. The list of tools is not exhaustive since there is an ever-increasing amount of simulation tools currently available on the market. We analyze the different visualization capabilities available for each of them. Complete tables of the tools reviewed are made available as appendices to this paper. The following characteristics were evaluated for each tool:

- **Open or proprietary**: one of our objectives is to determine the factors that drive the inclusion of visualization capabilities to a simulation tool. As mentioned in (Banks and Chwif 2011) and (Roman 2005), visualization generally leads to increased sales so it would stand to reason that proprietary software would offer better visualization to their users. There is also a correlation between the academic and commercial nature of these tools. Open-source simulation tools tend to be issued from academic contexts while proprietary tools come from a commercial context.
- **Logging process**: simulation tools tend to offer three levels of support for logging capability. If they do not offer any logging capability, the column indicates "none". If they offer an automated logging capability that uses an internally defined format, the column will indicate "auto". If they allow users to define their own process for logging results, the column will indicate "user-defined". This characteristic is an indicator of the potential for interoperability. Since users cannot be expected to follow a specification on their own, the only case where a specification for simulation logs could be reliably implemented is when logging is automated.
- **Simulation formalism**: we surveyed this characteristic to determine whether it was less likely for a given methodology to support visualization, potentially indicating that developing visualization

capability is even more difficult for this method. Unfortunately, in some cases, it was not possible to accurately determine the employed method.

- **Visualization method**: we indicate which of 5 visualization methods are supported. We add an additional category for tools that do not fit clearly into one of the categories. This is meant to indicate how common each one is. It could also be viewed to indicate how prohibitive is the increased effort required to support certain visualization methods (i.e., 3D vs. 2D visualization).
  - **2D**: tools that offer 2D animation; for instance a GUI to build models with animations. Some tools allow to define the appearance of entities, and to include contextual information such as background images to help the user to connect the simulation with the original system.
  - **3D**: tools that offer 3D visualization. Animations can include moving entities if the field of view can be freely controlled by the users, pan the scene and control its roll, pitch, and yaw.
  - **Graph-based**: Tools that offer visualizing a simulation on a graph type of chart. This is more typical with graph-based formalisms such as Petri Nets or State Machines.
  - **Grid-based**: Tools that offer visualizing a simulation on a graph. Tools with this capability allow visualization on a 2D or 3D lattice with basic models replicated as cells.
  - **Map-based**: Tools that offer visualizing a simulation on a geographic map. Tools with this capability typically provide a way to view models on a map and animate their movement or the evolution of their state across the simulation.
  - **Other**: Tools that offer visualization capabilities that do not fit into one of the 5 categories. For example, agent-based tools that can animate entities (workers; transport carts in a factory floor) like AnyLogic or Arena.
- **Analytics**: Tools that offer the capability of visualizing analytical charts derived from the simulation results. Typical examples of these are pie charts, bar charts or line charts. These are not grouped with the other visual methods because they do not involve reconstructing the simulation on a step-by-step basis. They are usually simpler to implement.
- **Domain**: Tools that provides enough flexibility to simulate multiple application domains will indicate a "Multi" in this column or "Single" otherwise. General purpose simulation tools are for example ADEVS (Nutaro 2010) or OpenModelica (Fritzson et al. 2020), they usually require modelers to code models themselves but can simulate any real-world system. Specific purpose simulation tools are geared towards a narrow application domain and are often parametric or rely on predetermined libraries of models. For example, energy in buildings with EnergyPlus (Crawley et al. 2001) or forest growth with CAPSIS (Dufour-Kowalski et al. 2011).
- **Language**: This column indicates the programming language in which the simulation tools were developed. This is provided mostly for context and did not factor in the analysis.

To consider the analysis of the survey that will follow, we first present a series of charts that provide insight on the sample (Figure 1). The first chart shows that the survey contains a higher proportion of open source software. This is because commercial software tends to be opaquer regarding some of their characteristics such as development language, logging capabilities or simulation formalism. Because of this, it was often impossible to determine their capability clearly enough to include them in the survey. The second chart shows that the sample likely overrepresents the DES and DEVS formalisms. This is due to our research interests being oriented specifically towards the DEVS formalism and we are interested in how to improve visualization of DEVS models. The third chart shows that more than half of the surveyed software provide a flexible logging process of some sort. This is double-edged. On one hand, it means that the software does not provide rigid, well-defined output formats for their results which makes interoperable visualization and analysis of simulation results difficult if not impossible. However, it also means that they provide the required flexibility to implement standards or specifications for their outputs. The fourth and final chart provides an overview of the visualization capabilities of the entire sample. As can be seen, 2D, 3D and analytics are the most used visualization methods while map and grid-based representation, are much less popular. This is likely because these two methods are meant for models that represent a spatial real-world system only. Another interesting take away is that around 25% percent of simulators have no visualization capability at all. These simulators stand to benefit the most from interoperable visualization and analysis.

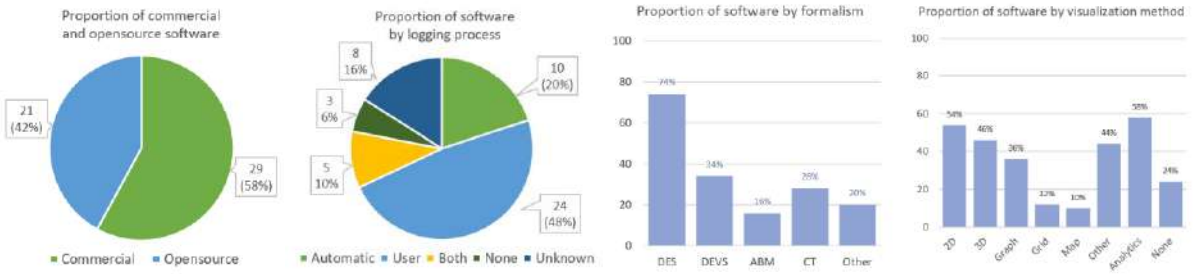

Figure 1: Characterization of the survey sample.

The importance of visualization becomes apparent when the visualization capabilities are put in relation with the open source or commercial nature of simulation software (Figure 2). Commercial simulation software overwhelmingly tends to provide some sort of visualization capability. Since this software is generally meant for end users that are not experts in simulation theory, an intuitive way to present simulation results is required. Furthermore, as noted by many other authors, visualization capability increases sales of the software, an undeniably important factor (Banks and Chwif 2011). Therefore, commercial producers can afford to spend time and resources on visualization capabilities when it results in additional sales. For open source simulation software, the opposite is true, particularly when considering that many of this software originates from academic efforts. An interesting point to note is that grid-based visualization does not seem to be a worthwhile visualization method for commercial software. This is possibly because this method is suited to a specific category of simulation problem and is perhaps still too abstract for non-expert users. Map-based visualization is much more common in commercial software than open source software although it is uncommon overall. Again, this is likely due to it being suited to a specific category of problems, namely the visualization of spatial models.

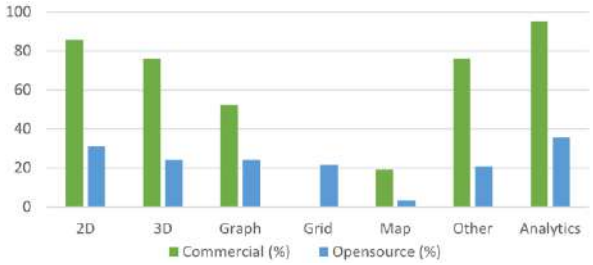

Figure 2: Proportion of commercial and open source software by visualization method.

The survey indicates that overall, simulation software meant for specific domains tend to focus on 2D or 3D visualization more than their counterpart (Figure 3). We surmise that this is because these visualization methods require more effort to develop than other approaches since elements specific to the application domain must be visually represented. For example, a 3D crowd simulator such as in (Schyndel et al. 2016), requires models specifically built to represent certain simulation elements, notably the building and at least one variation of a pedestrian. Running the simulation in any another setting requires a 3D model of that setting. This multiplies the development effort when trying to generate visualizations. Building 3D representations such as these is a costly endeavor. We can also see from the chart that graph-based and grid-based visualization are mostly employed in multi-domain simulation software, possibly because they are simple visualization mechanisms that can easily be adapted to a variety of simulation domains.

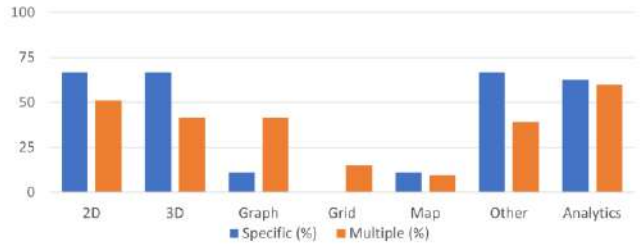

Figure 3: Proportion of software by application domain and visualization method.

The next chart shows visualization capabilities according to simulation formalisms. DEVS simulation software tend to provide limited visualization capabilities, generally limited to analytics, graph based or grid based representations (Figure 4). We believe this is due to two factors. One, these representations are easier to develop than 2D or 3D visualization and are favored by an R&D community that attributes less importance to visualization. Two, grid and graph-based visualization lend themselves well to the reproduction of discrete event traces: there is no need to interpolate an animation between two events as would be the case with a 3D visualization (for example a pedestrian walking). Simulation software based on agents or continuous time provide the best overall capabilities, particularly for 2D and 3D visualization.

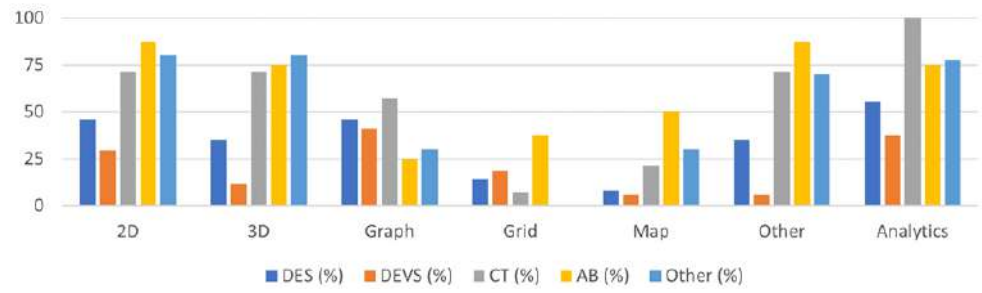

Figure 4: Proportion of software by formalism and visualization method.

As discussed earlier, (Van Tendeloo and Vangheluwe 2017) provides levels of visualization for DEVS simulation software. Grouping the 16 DEVS based simulation software in the survey according to their levels of visualization, it becomes clear that there is a gap in visualization capability (Figure 5).

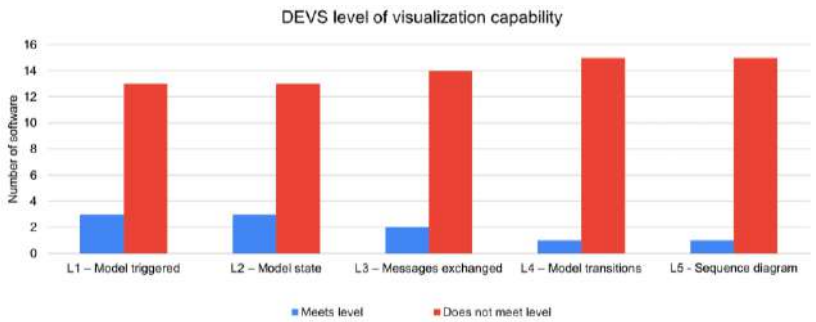

Figure 5: Proportion of software by formalism and visualization method.

Barely a quarter of the software provides visualization that allows users to see triggered models and their state across the simulation. Only one software meets all levels of visualization: *MS4ME,* a proprietary, DEVS only software.

5 CONCLUSION

We discussed the role and the state of simulation visualization and presented a survey of simulation tools available in both academic and commercial contexts. The survey shows that there are significant gaps when it comes visualizing simulation results. For all the advantages that visualization offers, increased collaboration, support in debugging models and faster model development iterations, it is generally put aside by academic simulation tools. Commercial tools tend to fare much better, but they offer no interoperability for the visualization of results. A user cannot import results of a one tool into another for visualization or simply, to compare results. This is an obstacle to transparency in simulation, one of the better-known pitfalls of simulation. We believe this could be addressed by decoupling visualization from simulation models, simulators, and even, simulation formalisms when possible.

This could be achieved by designing a specification for simulation results. Indeed, providing better tools for model developers, simulation software developers and consumers of simulation products is a major step towards democratization of the field. A once developed, always ready to use visualization platform could have radical impacts on the simulation lifecycle. Modelers would avoid spending efforts and resources on developing their own *ad hoc* simulation visualizations. They could use this common platform to analyze,

debug, demonstrate, and share them with others. This would speed up the simulation development cycle and allow modelers to focus on modeling rather than the development of tools. This is particularly relevant in an academic context where resources are often limited. Any tool that supports the specification could be easily used by modeler. An increasingly rich ecosystem of robust visualization tools would emerge gradually and be at the disposal of the community.

REFERENCES

Banks, J. 1998. *Handbook of Simulation*. Hoboken, New Jersey, John Wiley & Sons, Inc.

Banks, J., and L. Chwif. 2011. "Warnings about Simulation". *Journal of Simulation* 5 (4), pp. 279–91.

Bell, P.C., and R.M. O'Keefe. 1994. "Visual Interactive Simulation: A Methodological Perspective". *Annals of Operations Research* 53 (1), pp. 321–42.

Belloli, L., D. Vicino, C. Ruiz-Martin, and G.A. Wainer. 2019. "Building Devs Models with the Cadmium Tool". In *Proceedings of the 2019 Winter Simulation Conference*, pp. 45–59.

Busaman, A., K. Mekchay, S. Siripant, and S. Chuai-Aree. 2015. "Dynamically Adaptive Tree Grid Modeling for Simulation and Visualization of Rainwater Overland Flow". *International Journal for Numerical Methods in Fluids* 79 (11), pp. 559–79.

Collins, A.J., and D. Knowles Ball. 2013. "Philosophical and Theoretic Underpinnings of Simulation Visualization Rhetoric and Their Practical Implications". In *Ontology, Epistemology, and Teleology of Modeling and Simulation*, pp. 173–91.

Collins, A.J., D. Knowles Ball, and J. Romberger. 2014. "Exploring the 'Whys' of Simulation Visualization". In *MODSIM World 2014*, pp. 1–8.

Collins, A.J., D. Knowles Ball, and J. Romberger. 2015. "A Discussion on Simulations' Visualization Usage". In *Proceedings of the 2015 Winter Simulation Conference*, pp. 2827–35.

Collins, A.J., D. Knowles Ball, and J. Romberger. 2015. "Simulation Visualization Issues for Users and Customers". *Simulation Series*, 47 (2), pp. 17–24.

Crawley, D.B., L.K. Lawrie, F.C. Winkelmann, W.F. Buhl, Y.J. Huang, C.O. Pedersen, R.K. Strand, R.J. Liesen, D.E. Fisher, M.J. Witte, and J. Glazer, 2001. "EnergyPlus: Creating a New-Generation Building Energy Simulation Program". *Energy and Buildings*, 33 (4), pp. 319–31.

Dufour-Kowalski, S., B. Courbaud, P. Dreyfus, C. Meredieu, and F. Coligny. 2011. "CAPSIS: An Open Software Framework and Community for Forest Growth Modeling". In *Annals of Forest Science* 69 (2), pp. 221–33.

Eriksson, H., H. Magnusson, P. Fritzson, and A. Pop. 2008. "3D Animation and Programmable 2D Graphics for Visualization of Simulations in OpenModelica". *Science*, pp. 184–95.

Fritzson, P., A. Pop, K. Abdelhak, A. Ashgar, B. Bachmann, W. Braun, D. Bouskela, R. Braun, L. Buffoni, F. Casella, R. Castro, R. Franke, D. Fritzson, M. Gebremedhin, A. Heuermann, B. Lie, A. Mengist, L. Mikelsons, K. Moudgalya, L. Ochel, A. Palanisamy, V. Ruge, W. Schamai, M. Sjölund, B. Thiele, J. Tinnerholm, and P. Östlund, 2020. "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". *Modeling, Identification and Control* 41 (4), pp. 241–95.

Fujimoto, R.M. 2000. *Parallel and Distributed Simulation Systems*. John Wiley and Sons Inc.

Höger, C., A. Mehlhase, C. Nytsch-Geusen, K. Isakovic, and R. Kubiak. 2012. "Modelica3D - Platform Independent Simulation Visualization". In *Proceedings of the 9th International MODELICA Conference,* Munich*,* Germany, pp. 485–94.

Hurrion, R.D. 1978. "An Investigation of Visual Interactive Simulation Methods Using the Job-Shop Scheduling Problem". *The Journal of the Operational Research Society* 29 (11), pp. 1085-1093.

Knowles Ball, D., and A.J. Collins. 2012. "Simulation Visualization Rhetoric and Its Practical Implications". In *48th Annual Meeting of Southeastern Chapter of INFORMS*, pp. 584–91.

Kuljis, J., R. Paul, and C. Chen. 2001. "Visualization and Simulation: Two Sides of the Same Coin?". *Simulation* 77 (3–4), pp. 141–52.

Law, A.M. 2015. *Simulation Modeling and Analysis*. 5th ed. Tucson, Arizona, McGraw-Hill Education.

Macal, C.M. 2001. “Simulation and Visualization”. *SIMULATION* 77 (3–4), pp. 90–92.

M., Tarek, T. Al-Atassi, T. Brandys, V. Naik, H. Lapierre, and B.K. Lam. 2018. “Impact of Visualization on Simulation Training for Vascular Anastomosis”. *Journal of Thoracic and Cardiovascular Surgery* 155 (4), pp. 1686-1693.

Muller, W., and H. Schumann. 2003. “Visualization Methods for Time-Dependent Data - an Overview”. *Proceedings of the 2003 Intl. Conference on Machine Learning and Cybernetics*, pp. 737–45.

Nutaro, J.J. 2010. *Building Software for Simulation: Theory and Algorithms, with Applications in C++*. Wiley Publishing.

Regt, H.W. 2014. “Visualization as a Tool for Understanding”. *Perspectives on Science*,22(3), 377–96.

Regt, H.W., and W.S. Parker. 2014. “Introduction : Simulation, Visualization, and Scientific Understanding”. *Perspectives on Science* 22 (3), pp. 311–17.

Roman, P.A. 2005. “Garbage in, Hollywood Out”. *Proceedings SimtecT*, pp. 2–6.

Sargent, R.G. 2008. “Verification and Validation of Simulation Models”. In *2008 Winter Simulation Conference*, pp. 157–69.

Sarjoughian, H.S., and B.P. Zeigler. 1998. “DEVSJAVA : Basis for a DEVS-Based Collaborative M & S Environment”. In *SCS International Conference on Web-Based Modeling and Simulation*, 7 p.

Schyndel, M.V., O. Hesham, G.A. Wainer, and B. Malleck. 2016. “Crowd Modeling in the Sun Life Building”. *Proceedings of SimAUD*. 8 p.

Seide, S.E., K. Jensen, and M. Kieser. 2021. “Utilizing Radar Graphs in the Visualization of Simulation and Estimation Results in Network Meta-Analysis”. *Research Synthesis Methods*, 12(1), 96–105.

SIMULATION. 1987. “Special Animation and Discrete Event Simulations Issue”. 49 (3).

Sokolowski, J.A., and C.M. Banks. 2010. *Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains*. Hoboken, New Jersey, John Wiley & Sons, Inc.

Tendeloo, Y.V., and H.L. Vangheluwe. 2017. “An Evaluation of DEVS Simulation Tools”. *Simulation* 93 (2), pp. 103–21.

Vernon-Bido, D., A.J. Collins, and J.A. Sokolowski. 2015. “Effective Visualization in Modeling and Simulation”. In *48th Annual Simulation Symposium*, pp. 33–40.

Wainer, G.A. 2002. “CD++: A Toolkit to Develop DEVS Models”. *Software - Practice and Experience* 32 (13), pp. 1261–1306.

Wenzel, S., and U. Jessen. 2001. “The Integration of 3-D Visualization into the Simulation-Based Planning Process of Logistics Systems”. *Simulation*, 77 (3–4), pp. 114–27.

Zeigler, B.P., H. Praehofer, and T.G. Kim. 2000. *Theory of Modeling and Simulation*. Academic Press.

Zoellner, C., M.A. Al-Mamun, Y. Grohn, P. Jackson, R. Worobo. 2018. “Postharvest Supply Chain with Microbial Travelers: A Farm-to-Retail Microbial Simulation and Visualization Framework”. *Applied and Environmental Microbiology*, 84 (17), pp. 1–13.

AUTHOR BIOGRAPHIES

BRUNO ST-AUBIN received his Ph.D. from Carleton University. His research interests include geospatial simulation using DEVS and simulation environments. His email address is staubin.bruno@gmail.com.

FERNANDO LOOR is a Ph.D. student at the Departamento de Computación at Universidad Nacional de San Luis. His email address is fernandoloor1@gmail.com.

GABRIEL WAINER, Ph.D., is a Full Professor at the Department of Systems and Computer Engineering at Carleton University. He is a Fellow of SCS. His email address is gwainer@sce.carleton.ca.

A SURVEY OF VISUALIZATION CAPABILITIES OF SIMULATION SOFTWARE

Software	Open source or proprietary	Logging process	Simulation formalism	Visualization method						Analytics	Domain	Language
				2D	3D	Graph	Grid	Map	Other			
ADEVS	Open source	User	DES, DEVS	N	N	N	N	N	N	N	Multi	C++, Java
AnyLogic	Proprietary	User	DES, DEVS, ABM, CT, Others	Y	Y	Y	N	Y	Y	Y	Multi	Java
Arena	Proprietary	Auto, User	DES, CT	Y	Y	Y	N	N	Y	Y	Multi	
Autodesk Inventor	Proprietary		Other	Y	Y	N	N	N	Y		Single	
Cadmium	Open source	Auto	DES, DEVS	N	N	N	N	N	N	N	Multi	C++
CAPSIS	Open source	User	Other	Y	Y	N	N	N	Y	Y	Single	Java
CD++	Open source	Auto	DES, DEVS	N	N	Y	N	N	N	N	Multi	C++
CPN Tools	Open source	User	DES	N	N	Y	N	N	N	N	Multi	
DEVS / C++	Open source	User	DES, DEVS	N	N	N	N	N	N	N	Multi	C++
DEVS / HLA	Open source	User	DES, DEVS	N	N	N	N	N	N	N	Multi	
DEVSimPy	Open source	Auto	DES, DEVS	N	N	N	N	N	N	Y	Multi	Python
DEVSJAVA	Open source		DES, DEVS	N	N	N		N	N		Multi	Java
DEVS-Suite	Open source	Auto, User	DES, DEVS	Y	N	Y	Y	N	N	Y	Multi	Java
Dymola	Proprietary	None	CT	Y	Y	N	N	N	Y	Y	Multi	Modelica
EnergyPlus	Open source	Auto	Other	N	N	N	N	N	N	N	Single	C++
ExtendSim	Proprietary	Auto, User	DES, ABM, CT, Others	Y	Y	Y	N	Y	Y	Y	Multi	
FlexSim	Proprietary		ABM	Y	Y	N	N	N	Y	Y	Multi	
GALATEA	Open source	Auto	DES, DEVS	N	N	Y	N	N	N	N	Multi	Java
GoldSim	Proprietary	User	DES, CT	N	N	Y	N	N	N	Y	Multi	
JaamSim	Open source	User	DES	Y	Y	Y	N	N	N	Y	Multi	Java
JDEVS	Open source	Auto, User	DES, DEVS	Y	Y	Y	Y	N	N	N	Multi	Java
js-simulator	Open source	User	DES, ABM	Y	N	N	Y	N	Y	N	Multi	JavaScript
LibCppSim	Open source	User	DES	N	N	N	N	N	N	N	Multi	C++
MASON	Open source	User	DES, ABM	Y	Y	N	Y	N	Y	Y	Multi	Java

Software	Open source or proprietary	Logging process	Simulation formalism	Visualization method						Analytics	Domain	Language
				2D	3D	Graph	Grid	Map	Other			
MedModel	Proprietary	User	DES	Y	N	N	N	N	Y	Y	Single	
Micro Saint Sharp	Proprietary	User	DES	Y	Y	Y	N	N	Y	Y	Multi	
MS4ME	Proprietary	User	DES, DEVS	Y	N	Y	N	N	N	Y	Multi	Java
NetLogo	Open source	User	ABM	Y	N	N	Y	Y	N	Y	Multi	Scala, JAVA
OpenFlows	Proprietary	Auto	Other	Y	Y	N	N	Y	N	Y	Single	
OpenModelica	Open source	None	CT	Y	Y	N	N	N	Y	Y	Multi	Modelica
PowerDEVS	Open source	User	DES, DEVS, CT	N	N	Y	N	N	N	Y	Multi	C++
Ptolemy II	Open source	User	DES, CT	N	Y	N	N	N	Y	Y	Multi	Java
Python DEVS	Open source	Auto	DES, DEVS	N	N	N	N	N	N	N	Multi	Python
Radiance	Open source	Auto	ABM	N	Y	N	N	N	Y	N	Single	
SIM.JS	Open source	None	DES	N	N	N	N	N	N	N	Multi	Javascript
Simcad Pro	Proprietary		DES, CT	Y	Y	Y	N	N	Y	Y	Multi	
SimEvents for Simulink	Proprietary	User	DES, ABM, CT	Y	Y	N	N	Y	Y	Y	Multi	Java, Matlab scripting language
SIMULIA	Proprietary		Others	Y	Y	N	N	N	Y	Y	Multi	
SimJulia	Open source	User	DES	N	N	N	N	N	N	N	Multi	Julia
SimPy	Open source	User	DES	N	N	N	N	N	N	N	Multi	C++
SimScape	Proprietary		Other	Y	Y	N	N	N	Y	Y	Single	
SIMUL8	Proprietary	User	DES, CT	Y	Y	Y	N	N	Y	Y	Multi	
SmallDEVS	Open source	Auto, User	DES, DEVS	N	N	N	N	N	N	N	Multi	
Tecnomatix Plant Simulation	Proprietary		DES	Y	Y	Y	N	N	Y	Y	Multi	
TRNSYS	Proprietary	Auto	Other	Y	Y	Y	N	N	N	Y	Single	
VisualSim	Proprietary	User	DES, CT	N	N	N	N	N	N	Y	Multi	
VLE	Open source	User	DES, DEVS, CT	Y	N	N	Y	N	N	Y	Multi	C++
VUMIE	Proprietary	Auto	Other	N	N	N	N	N	Y	N	Single	
WITNESS	Proprietary		DES, CT	Y	Y	Y	N	N	Y	Y	Multi	
X-S-Y	Open source	User	DES, DEVS	N	N	N	N	N	N	N	Multi	C#

B SURVEY OF THE LEVEL OF VISUALIZATION FOR DEVS SIMULATION

Software	L1 – Model triggered	L2 – Model state	L3 – Messages exchanged	L4 – Model transitions	L5 - Sequence diagram
ADEVS	N	N	N	N	N
Cadmium	N	N	N	N	N
CD++	N	N	N	N	N
DEVS / C++	N	N	N	N	N
DEVS / HLA	N	N	N	N	N
DEVSimPy	N	N	N	N	N
DEVSJAVA	N	N	N	N	N
DEVS-Suite	Y	Y	Y	N	N
GALATEA	N	N	N	N	N
JDEVS	Y	Y	N	N	N
MS4ME	Y	Y	Y	Y	Y
PowerDEVS	N	N	N	N	N
Python DEVS	N	N	N	N	N
SmallDEVS	N	N	N	N	N
VLE	N	N	N	N	N
X-S-Y	N	N	N	N	N