Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

USING A SOFTWARE DESIGN PATTERN FOR REDESIGN ROUTED DEVS FORMALISM

Mateo Toniolo
Maria Julia Blas
Silvio Gonnet

Universidad Tecnológica Nacional – Facultad Regional Santa Fe
Lavaisse 610
Santa Fe, 3000, ARGENTINA

ABSTRACT

Routed DEVS (RDEVS) models improve traditional discrete-event models by enhancing the development of routing processes over predefined behaviors. In this paper, we demonstrate how a Software Engineering design pattern, specifically the Decorator pattern, was applied to the RDEVS formalism design to include event tracking into the models without altering their expected behavior. As a result, we provide a solution that allows getting structured data from RDEVS models at execution time.

1 INTRODUCTION

The Routed DEVS (RDEVS) formalism (Blas et al. 2017) simplifies the process of defining routing processes over the Discrete Events System Specifications (DEVS) (Zeigler et al. 2018) models by introducing a new modeling level known as routing behavior. Acting as a "layer" above DEVS, RDEVS provides routing functionality without requiring direct interaction with DEVS functions. The core definition of RDEVS is based on conceptual modeling allowing the use DEVS simulators as execution engines. However, when RDEVS models are run with DEVS simulators, the resulting data remains DEVS-based, limiting the analysis of event flows between models influenced by routing policies. To address this limitation, a conceptual modeling-based solution using event trackers is proposed to gather structured data from RDEVS simulations. The solution, centered on the Decorator pattern, adds responsibilities to the existing conceptualization of RDEVS simulation models without altering their behavior. Our proposed solution involves implementing a Java package attached to the RDEVS Library to collect structured event flow data dynamically during the simulation, recorded in JavaScript Object Notation (JSON).

2 DESIGN AND IMPLEMENTATION OF RDEVS TRACKERS

Unlike regular DEVS events, RDEVS models utilize events embedded with routing data. This data is vital for understanding various aspects of RDEVS dynamics, such as the number of accepted/rejected events in a routing node, the types of events accepted/rejected by specific nodes under certain state conditions, and the frequency of produced output events accepted/rejected by destination models.

In software engineering, the Decorator design pattern defines a flexible approach to enclosing a component in another object that adds a "border" with the intent of attaching an additional responsibility dynamically. In the context, we have altered the conceptual design of the RDEVS formalism by including tracking responsibilities as “decorators” of the set of elements used in discrete-event models. The UML class diagram of such additional feature can be seen in the link www.ingar.santafe-conicet.gov.ar/rdevs.

In the implementation phase, a comprehensive approach was adopted to address each component of the formalism, including models, data types, couplings, and events following our conceptualization. For seamless data capture and real-time monitoring during the simulation process, dedicated Java classes were developed for each component. These classes were carefully designed to not only capture the initial data required for simulation but also to dynamically collect and record any relevant data that emerges throughout the simulation. By tailoring specific Java classes for each formalism component, the tool ensures efficient and accurate tracking of information, facilitating a detailed analysis of the system's behavior under various scenarios. This implementation strategy guarantees the generation of valuable insights and provides a solid foundation for understanding the intricate dynamics of the RDEVS models during execution.

At the beginning of each simulation, the trackers attached to the models are automatically created through the initialization process. During this step, the data related to the model structure (i.e., static data) is collected in each tracker. On the other hand, the data related to the simulation execution (i.e., dynamic data) is obtained during the simulation process (once the models are initialized). Once the simulation concludes, the tracker model contains all the dynamic data related to event exchange. To efficiently store this data, a JSON file is generated. JSON, a lightweight data-interchange format written in JavaScript object notation, proves to be an ideal choice. By incorporating the "@Expose" tag into a predefined navigation among the Java classes, we ensure that the minimum set of data required to reconstruct the model is stored.

The adoption of JSON formatting offers several advantages. Firstly, it enables seamless integration with other software tools for in-depth analysis of RDEVS models. With JSON data, specific visualizations can be designed to enhance comprehension of the intricate routing processes within RDEVS simulations. The visualization module is a key part of our research, using the well-organized JSON data to provide valuable insights into the complex routing dynamics in RDEVS models. Visualization techniques help researchers understand the intricate network interactions in a straightforward way. It makes interpreting simulation results much easier, giving us a deeper understanding of how the system behaves and what's going on behind the scenes. Currently, we employ Sankey diagrams (supported by the Google Charts library) to represent the flow of data between nodes. When applied to RDEVS, these diagrams provide a visual representation of the event flow between ports and models within the formalism. Additionally, we are currently working on developing a parser that takes the JSON data with the RDEVS structure and generates an HTML or .JS file to create the Sankey diagram automatically. This parser will streamline the process of visualizing the routing dynamics, allowing us to efficiently interpret and communicate the simulation results in a more accessible and interactive manner. The integration of the parser with the visualization module will further enhance the utility of the tool, making it a comprehensive solution for studying and presenting the event flow between ports and models within the RDEVS formalism. This tool significantly contributes to RDEVS research, helping us to understand the model's routing behavior.

Defining an output structure in the JSON format was essential to effectively present information from all simulation runs, ensuring accessibility and clarity for further analysis and interpretation. You can find an example of data captured in a simulation execution at www.ingar.santafe-conicet.gov.ar/json-rdevs, which faithfully represents an RDEVS model simulation, including all relevant data structures and event exchanges following our conceptualization.

3 CONCLUSIONS

Our alternative solution provides an accurate separation of concerns that allows maintaining the advantages of using DEVS simulators for executing RDEVS models while collecting structured data regarding how routing policies are allowing/blocking the domain processing. Besides allowing to collect RDEVS-based data with the tracking, we expect to improve the extensibility, maintainability, and readability of RDEVS models. The DEVS-based data gathered by the DEVS simulator can be followed with the RDEVS-based structured data obtained from our trackers to allow a complete analysis of the simulation models.

REFERENCES

Blas, M., S. Gonnet, and H. Leone. 2017. "Routing Structure over Discrete Event System Specification: A DEVS Adaptation to Develop Smart Routing in Simulation Models". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page. 774-785. Piscataway, New Jersey, Institute of Electrical and Electronics Engineers, Inc.

Zeigler, B. P., A. Muzy, and E. Kofman. 2018. *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. 3rd ed. London, Academic Press.