Effects of Functional and Declarative Modeling Frameworks on System Simulation

John Morris Gregory Mocko John Wagner
Dept. of Mechanical Engineering
Clemson University
Clemson, SC, USA
Submitted 25 July 2025

Summary of article submitted to Modeling, Estimation and Control Conference (MECC 2025) in Pittsburgh, PA, USA

Modelers, trying to represent the behavior of a complex system, are limited by the expressiveness of the available frameworks [1]. The ultimate goal of modeling a system is to understand its interactions, generally for the purpose of simulating its state. Due to the prevalence of computer simulation, digital systems and counting methods have received the bulk of consideration over the past decades, focusing discussion of different modeling paradigms on the design of software systems. The purpose of this paper is to review how four of these structural categories effect the simulation of physics-based systems, namely imperative, declarative, functional, and object-oriented paradigms.

Imperative models describe only the steps of conducting a single simulation and are comprised of a single process. Declarative models, on the other hand, encode the behavior of a system into a framework that allows many possible simulations to be executed. The manner of doing so is directly related to representing system behaviors as functions constraining the values of state variables [2]. Most declarative frameworks–such as bond graphs, stock and flow diagrams, or Bayesian networks–express system behavior as a set of functions encoded in the model framework. This is explicitly the method of representation for functional paradigms, which lead to high degree of simulatability–though often at the cost of being more difficult to develop. Object-oriented models are easier to develop because they allow the partitioning of a system to a set of independent states. The ability of object-oriented models to be simulated is shown to be directly related to how functional the framework is: frameworks that employ non-functional methods, data hiding (through encapsulation), and imperative inter-system coupling often fail when employed to model the heterogeneous systems typical of physics-based modeling [3].

Simulation is defined in terms of deriving the state of a system in relation to a set of known states [4]. Consequently, simulation is the process of encoding a set of functions transforming known inputs to an unknown output, in such a way that respects the behavior of the modeled system [5]. Every paradigm can be evaluated based on its ability to represent and rearrange behavioral functions into an eventual simulation. Highly simulatable frameworks maximize the amount of function composition possible for a system, such that all system behaviors can be discovered by interrogating the model.

The effects on simulation for each of these paradigms is demonstrated through a case study of a double pendulum. The pendulum is scoped to a set of six variables, with its behavior given by six functions relating the variables. This model is expressed in a strictly imperative framework (using the MATLAB programming language), a functional-imperative framework (block diagrams), an imperative-functional and imperative-non-functional object-oriented framework (using the C++ programming language), a declarative-object-oriented framework (Modelica), and a declarative-functional framework (constraint hypergraphs [6]). Each of these frameworks are compared by their simulatability, measured by the extent of structural reuse afforded by the framework as well as the number of simulations that can be autonomously derived from the model.

The performance of each framework is explained in terms of its ability to rearrange functions. Observations include that simulatability is directly tied to the declarativity of a framework, and that frameworks must enforce global states and functional methods to provide declarative processing.

[1] M. J. Beeson, ''Towards a computation system based on set theory,'' *Theoretical Computer Science*, vol. 60, no. 3, pp. 297–340, Dec. 1988, ISSN: 03043975. DOI: 10.1016/0304-3975(88)90115-6. Accessed: Mar. 7, 2025.

[2] J. C. Willems, ''The Behavioral Approach to Open and Interconnected Systems,'' *IEEE Control Systems Magazine*, vol. 27, no. 6, pp. 46–99, Dec. 2007, ISSN: 1941-000X. DOI: 10.1109/MCS.2007.906923. Accessed: Sep. 17, 2024.

[3] P. Wegner, ''Concepts and paradigms of object-oriented programming,'' *SIGPLAN OOPS Mess.*, vol. 1, no. 1, pp. 7–87, Aug. 1990, ISSN: 1055-6400. DOI: 10.1145/382192.383004. Accessed: Feb. 19, 2025.

[4] B. Fong, ''The Algebra of Open and Interconnected Systems,'' Ph.D. dissertation, arXiv, Oxford University, Sep. 2016. DOI: 10.48550/arXiv.1609.05382. arXiv: 1609.05382. Accessed: Sep. 3, 2024.

[5] B. P. Zeigler, A. Muzy, and E. Kofman, *Theory of Modeling and Simulation*, Third. Elsevier, 1976, ISBN: 978-0-12-813370-5. DOI: 10.1016/B978-0-12-813370-5.00002-X. Accessed: Feb. 26, 2025.

[6] J. Morris, G. Mocko, and J. Wagner, ''Unified System Modeling and Simulation via Constraint Hypergraphs,'' *Journal of Computing and Information Science in Engineering*, vol. 25, no. 6, p. 061 005, Apr. 2025. DOI: 10.1115/1.4068375.