Software and Systems Modeling (2023) 22:1733–1735
https://doi.org/10.1007/s10270-023-01140-2

EDITORIAL

Adopting the concept of a function as an underlying semantic paradigm for modeling languages

Benoit Combemale[1] · Jeff Gray[2] · Bernhard Rumpe[3]

Published online: 30 November 2023

There are researchers and practitioners in the areas of modeling and modeling language design who focus mainly on a syntactic point of view. For example, they may look at language design for language agglomerates like UML or SysML, as well as more customized DSLs, to consider the optimal syntactic constructs that are needed to cover all of the different kinds of phenomena occurring in a real-world problem context. Other researchers may focus on the semantic point of view to understand the meaning of a specific model as expressed in a language. A general challenge is when the same syntactic construct can be interpreted differently in various contexts of usage.

A well known but often confusing example is the class diagram. A class "Person" in such a diagram may have different meanings based on the phase of development in which it is used. If the diagram has been defined during business requirements elicitation, the class actually represents human beings. In this case, "Persons" are real objects in the world of physical things. When the same class diagram is then used for design, the very same class "Person" suddenly describes the data structure that is capable of collecting data about human beings and thus a purely virtual concept emerges. Somewhere between requirements elicitation and design, the interpretation adds a step of indirection that is not reflected in the syntax itself. Other prominent examples use different interpretations of various modeling elements, which frequently lead to confusion. The situation is even more challenging when physical systems are accompanied with digital twins, where software components intelligently control a physical representation.

✉ Bernhard Rumpe
bernhard.rumpe@sosym.org

Benoit Combemale
benoit.combemale@sosym.org

Jeff Gray
jeff.gray@sosym.org

1 University of Rennes, Rennes, France

2 University of Alabama, Tuscaloosa, AL, USA

3 RWTH Aachen University, Aachen, Germany

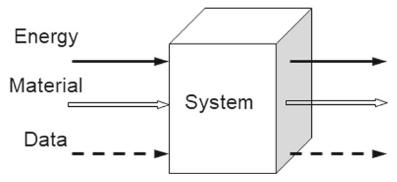
It would be helpful to have an underlying paradigm that connects all of the different interpretations and views (e.g., structure, data structure, interaction, behavior, state, agents, or activities) that are represented through various modeling techniques.

In the context of the upcoming SysML 2.0 definition, it is evident that a semantically sound and useful integration could play a major role toward addressing this challenge of differing interpretations. One possibility is to consider the often used paradigm of "function".

Both systems engineers and software engineers have created the idea of using functions as a development paradigm. In mechanical engineering and other engineering contexts, Pahl and colleagues have explored the functional paradigm as a mechanism to describe the functionality of a complex system by decomposing it into its elementary functions, which are realized by principal solutions [1]. As software engineers, we were very surprised to discover that our mechanical engineering colleagues have a fixed number of approximately 350 principal solutions that represent elementary components for all kinds of systems that describe all possible forms of atomic functions.

For conceptually or physically distributed computer systems, the same functional paradigm can be used to describe the collaboration among active computational units. Many interesting software systems are contextually and often physically distributed, which makes a functional paradigm appealing. Broy and Stølen have defined an explicit approach for such a functional paradigm [2], and others have similar approaches [3].

What we find most appealing is the degree of compatibility and similarity between the mechanical engineering and software engineering variants of the functional paradigm. They can be integrated and formalized on the mathematical concept of a function, given that there is a precise definition of "streams" of energy, material or data that enters the system and exits the system through some kind of appropriate "channel" or "interaction surface".



Lightweight versions of a functional paradigm are usually visible in various formal methods approaches, where logic-based description mechanisms are mapped into simpler semantic domains. For example, various forms of agents have their idea of functions as the underlying paradigm. Furthermore, structural decomposition in architecture languages can be explained using functional composition (i.e., connecting functions through their channels) and interaction can be described through the flow of elements on the channels. State, including data state and Statechart state, describes the internal state of the component, which becomes visible when a component may rely on the internal state when producing output. Activities and behavior represent the relationships between the input and the output of a component. The only other highly relevant and omnipresent modeling viewpoint that is not very well connected to the functional paradigm is the modeling of complex data structures (e.g., using class diagrams). Even though classes can in principle be regarded as components with input and output, in practice many classes are only used as data structures.

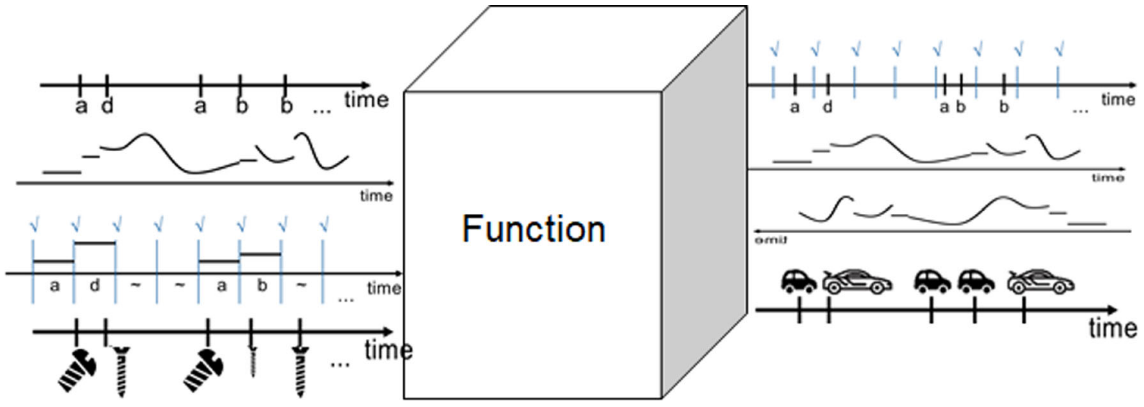
But the integration and harmonization of these viewpoints is not easy. Researchers may also find solutions for a helpful, generally usable integration of the functional paradigm and data structures. The integration of software and systems modeling paradigms based on functions seems feasible and may be the best approach to unify the relevant theories.

There is still one major challenge that we have to deal with, namely, the integration of continuously changing flows (e.g., electrical energy, analogous signals, or water) with discrete and even event-based flows such as physical objects like screws or cars or data messages. This is not so much a problem between software and mechanics, but a pure mathematical problem, where the mathematical theories exist in principle, but we have not yet sorted out what is practically the best solution path. Discretization of continuous flows is of course possible [4] but tedious to deal with when specifying systems. For many processes, computation is quick compared to the physical process. This is why Edward Lee's Ptolemy uses a super-dense time approach, where computation is so quick that little time progress occurs, but still the order of events is retained. Computation is very slow compared to the process and needs a different form of functional integration.

We can ask the mathematicians to improve the theories and let us modelers use the functional paradigm at a level of abstraction that is helpful for us to model our systems and software as efficiently as possible.

1 Content of this Issue

1. **Expert voice**
   - "A manifesto for applicable formal methods" by Mario Gleirscher, Jaco van de Pol, and Jim Woodcock
2. **BPMDS 2021 special section**
   Guest editors: Selmin Nurcan, Rainer Schmidt, and Adriano Augusto



3. **EMMSAD 2022 special section**
   Guest editors: Iris Reinhartz-Berger and Dominik Bork
4. **Theme Section on "Model and data engineering"**
   Guest editors: Christian Attiogbé, Sadok Ben Yahia, and Ladjel Bellatreche
5. **Regular paper**
   - "Correlating contexts and NFR conflicts from event logs" by Mandira Roy, Souvick Das, Novarun Deb, Agostino Cortesi, Rituparna Chaki, and Nabendu Chaki

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H., Design, E.: A Systematic Approach. Springer, Berlin (2007)
2. Broy, M., Stølen, K.: Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement. Springer Science, New York (2001)
3. Edward, A.L., Seshia, S.A.: Introduction to Embedded Systems - A Cyber-Physical Systems Approach. MIT Press, Cambridge (2017)
4. Shannon, C.: Communication in the presence of noise. Proc. Instit. Radio Eng. **37**(1), 10–21 (1949)