Software and Systems Modeling (2025) 24:1315–1316
https://doi.org/10.1007/s10270-025-01312-2

EDITORIAL

Standards in software development and modeling

Marsha Chechik[1] · Benoit Combemale[2] · Jeff Gray[3] · Bernhard Rumpe[4]

Published online: 2 August 2025

There are important standardization bodies that actually create very good and widely used standards in engineering and development. This is prominent in other engineering domains, but less common in computer science. We may speculate about the reasons, but it may be that computer science is relatively young, and therefore, techniques and methods evolve frequently, and standards may hinder this form of innovation. A second reason may be that in computer science, large companies are developing the de facto standards that are not necessarily becoming formal standards.

But in computer science, standards also ensure compatibility, interoperability, reliability, security, reusability, and potentially many other good properties across services, applications, systems, and technologies. And we all know some key categories and examples of relevant standards, such as programming language standards (e.g., ISO/IEC 9899—for C, Java Community Process (JCP) specs—for Java, ECMA-262/ISO/IEC 16262—for JavaScript, HTTP/HTTPS (RFC 9110) protocol—for web communication, and RFC 8259—for JSON).

The most relevant standards for Software & Systems Engineering are UML (first by the OMG and later by ISO/IEC 19505), IEEE 830 / ISO/IEC/IEEE 29148—for Software Requirements Specification, and the newly emerging standards around the digital twin technologies stack that are in discussion by the Digital Twin Consortium (DTC) and the Industrial Digital Twin Association (IDTA). As a side note, we would like to mention that the UML standardization was an important reason for starting this journal with its first issue in 2001.

However, while it is possible to standardize technologies and methods that can be used within and for software development, the converse is also true—it is also possible to use software development techniques to define standards in other domains. This allows us to state the following question:

> *"Where and how can software models be adopted to formalize standards used in the various existing domains?"*

When looking at today's standards in engineering and other domains (e.g., public healthcare administration, environmental sciences), we observe that many of these standards mention data structures for data exchange or storage, communication and interaction protocols or workflows, and other software-related elements. This is not surprising, because standards often deal with information, and information systems are inherently software. It is definitely a missed opportunity that these standards either (1) do not precisely formalize data structures, interactions, workflows and other software concepts, these things, or; (2) are define their own way of describing these aspects, within their own local standard definition language given in some appendix of the standards document.

There are numerous examples in newer standards that follow the approach of defining their own meta-language in the appendix and using models of this specific meta-language to define the actual standard. Among these, there are negative examples where the meta-language was only used within a specific standard, and a more general language (with associated tools) could have been used instead. But, there are also positive examples, such as the Specification and Description Language (SDL defined by ITU-T Recommendation Z.100 series), which was originally only meant to describe the interactions between telecommunication gadgets, but later also inspired other standards. Telecoms are also responsible for the Abstract Syntax Notation One (ASN.1), a formal notation for describing data transmitted by telecommunications

✉ Bernhard Rumpe
bernhard.rumpe@sosym.org

Marsha Chechik
marsha.chechik@sosym.org

Benoit Combemale
benoit.combemale@sosym.org

Jeff Gray
jeff.gray@sosym.org

[1] University of Toronto, Toronto, ON, Canada

[2] University of Rennes, Rennes, France

[3] University of Alabama, Tuscaloosa, AL, USA

[4] RWTH Aachen University, Aachen, Germany

protocols, that existed long before JSON and has recently been lifted to a JSON encoding.

Usually, these meta-standards have serious deficits on their own, as they are often incomplete, do not consider corner cases, lack precisely defined semantics (if at all), and lack tool support, which is either nonexistent or proprietary. Another issue that also occurs is the lack of common data structures used between standards, which limits the potential for exchangeability.

This raises a relevant question: "Are we, as computer scientists and designers of modeling and DSL languages, actually unable to create appropriate modeling techniques and languages for such standardization bodies, or have we simply done a poor job of communicating the existing languages that are available?" We feel that it is the latter, and we should begin by investigating how to help standardization bodies support more precise and formalized definitions of data structures, interaction patterns, workflows, and the like, such that the standard itself becomes formally accessible, consistency checks are possible, and automatic tooling for generation of interaction code and data storage are provided.

This would give the other engineering domains much greater opportunity and allow (1) a much earlier availability of constructive tools designed around the standards; (2) compatibility and interoperability between the standards; (3) advanced and helpful tools that can be specialized to map the standard definition directly into innovative tools (even by smaller companies); (4) communication and data exchange standardization in an ever growing internet, where many technical gadgets have not even been designed and produced yet; and (5) quick production of digital twins for any kind of physical gadget.

Furthermore, the modeling community needs to provide easy-to-use languages for the aforementioned purposes, but these languages must also be standardized. For this purpose, we need a precisely defined meta-standard that can be used to describe how to define standards. Standardization bodies can help to lead such an effort. It is also interesting to understand why the UML is not widely adopted as a standardization technique. Do we need a refined and potentially restricted form of the UML, that we might call StandardML, for such purposes?

As always, we are very interested in seeing foundational papers on these topics, such as a detailed analysis of an existing standard to help answer the question about how much of a standard can be precisely standardized using a language like the UML, or answering the converse question—what is missing in the UML (or SysML) to actually define relevant parts of the standard.

Funding Open Access funding enabled and organized by Projekt DEAL.