modeling of system dynamics, whereas the proposed approach does not capture these dynamics well-enough because it simply states the input/output expected modification of system variables (and their invariant properties), defined from an end-user point of view, without any other behavioral details on its realization. Matching these views is not an easy task, because the causality relationship underlying the dynamics of the system may be orthogonal to the causality or material flows balance deduced from the physics of a process, as proposed by the authors. Therefore, the resulting system decomposition may not be suitable for encapsulating the required dynamic behavior. References investigating the interconnection of physical models with hybrid-automata-based dynamic models, such as [1] should be exploited to this end.

Finally, it would have been helpful for the reader if the authors had stated the type of generic constructs required for some other application domains, and not only the process domain treated in the paper.

**Reference**


---


V. Valero and M.E. Cambronero

Instituto de Investigación en Informática de Albacete, Universidad de Castilla-La Mancha, Campus Univ. s/n, Albacete, Spain

This paper presents a technique for the description of Industrial automation systems, which is based on the B language [1], with the purpose of validating the final system with regards to the system requirements. It is based on an “a priori” approach of verification, by progressively refining an abstract specification with simple properties to be checked into more and more tangible and precise models. Then, this paper considers a rather simple example (a hydraulic system consisting of a tank and a valve) to illustrate this methodology.

The use of the B language is justified by stating that it is an efficient formalism to support model-driven specifications and proof-oriented specifications. This language has some capabilities that allow the verification of properties (invariants), and the instantiation of generic constructs, thus allowing the reuse of already proved constructs. The authors mention that there are some alternative approaches, like Petri nets of Finite State Automata, but they discard these techniques because they are mainly focused on the system dynamics, and hardly cover the description of some other properties, although it is not clear what kind of properties they are referring to. In fact, these techniques are broadly used for the design of this kind of system, and they are supported by tools that allow the designer to simulate the system and check some properties of interest on it.

An important class of properties that in many Industrial automation systems must be ensured are related to time, in the sense of fulfilling some kind of temporal restrictions in the execution of some actions, or fixing some bounds for the response time of some events in the system. We have not found a discussion about these timed properties in the paper, which could be a topic for
further research. Actually, most classical models of description have been extended to deal with time aspects. For instance, timed automata [2] are a very well-known formalism that has been used for the description of control systems, and it is supported by some tools, like UPPAAL [3] or KRONOS [4]. We can also find many timed extensions of the classical process algebras: timed CCS [5], temporal CCS [6], timed CSP [7], TPAL _p_ [8] and timed ACP [9]. In the case of Petri nets there are essentially two trends when extending them with time: either time is associated with transitions (transition durations), or time is associated with places or tokens (representing delays for the corresponding tokens to be available). There is also a third group of models that includes some kind of time information in the arcs, but in this case time must also be associated with places, tokens or transitions. A survey of the different approaches to introduce time in Petri nets can be found in Ref. [10].

Thus, the limits that Petri nets, Finite State Automata or process algebras have for describing some properties of interest can be overcome by adequately extending these models, for instance, by including time information, as we have seen.

However, the weak point in this paper is that there is no general description of how this methodology can be applied for a more general application. The example that illustrates the methodology is very simple, so it is not very useful in order to understand how it must be applied in general. In fact, owing to the mathematical orientation of the B language, it can be an extremely demanding task to use it to describe a rather complex system. To deal with this problem, the authors propose combining the descriptions written in terms of B machines, with equivalent descriptions written using more accepted formalisms, like UML. The authors mention that some effort has been made to translate UML descriptions into B machines [11, 12]. However, owing to the semiformal character of UML, these translations are not at all automatic, so the difficulties in the use of the B language cannot be skimmed over so easily.

Another aspect that is not covered in the paper is the cost of the refinements, i.e. at each step of refinement a check must be made to ensure that the invariants are preserved. This check is accomplished using the Atelier B tool, but the process is far from being automatic, so it can really be very expensive for a complex application.

Finally, the “a priori” verification approach is based on some properties that the designer must identify in the early stages of the system design, but, in many cases, once the system has been designed, some new properties appear that should also be checked. According to the proposed methodology, these new properties should be added into the B machines, and then, a new global check should be made. In terms of the classical “a posteriori” approaches, the cost of these proofs is known, as we only need to execute the model-checker with a new goal predicate, and we can check any property, without being limited to the proof of invariants.

References