An Ontological Approach for the Validation of Simulation Models of Manufacturing Systems

Sergio Benavent-Nácher[1,*,†], Stefano Borgo[2,†], Pedro Rosado Castellano[1,†] and Francesco Compagno[3,†]

[1]Universitat Jaume I, Castellón de la Plana, Av. Vicent Sos Baynat, s/n, 12006 Castelló de la Plana, Spain
[2]Laboratory of Applied Ontology ISTC-CNR, Via alla Cascata, 56/C, 38123 Povo TN, Italia
[3]University of Trento, Via Calepina, 14, 38122 Trento TN, Italia

Abstract

Model verification and validation are important tasks in the field of system modeling, and are essential in the development of software simulation systems. This type of software systems must be well-constructed to be compiled and executed, and must also adequately represent the actual systems to which it refers. As a software system, languages for the implementation of simulation systems usually have very general syntactic verification mechanisms, but they do not support the extensions of these constraints to include aspects related to the domain of the actual simulated system. To overcome these limitations, integrating ontological modeling and related reasoning capabilities into the simulation system development can be a significant improvement. The main goal is to develop ontological modules, here based on the foundational ontology DOLCE, that address the domain of the actual system and its representation in a simulation model. Although the paper concentrates on simulation models for the manufacturing systems, the approach is general and can be applied in other domains. The paper also discusses practical examples of (types of) inconsistencies that this approach detects.

Keywords

Manufacturing System, Simulation Model, Verification, Validation, Consistency, Ontology, DOLCE

1. Introduction

The behavior analysis of a system usually requires making a set of assumptions about how it works, and modeling this behavior to be analyzed as mathematical or logical relationships. For simple systems, it may be possible to define an analytic solution using mathematical methods (such as algebra, calculus, or probability theory) to obtain exact information on questions of interest. However, most real-world systems are too complex to be computed analytically, and the most common alternative solution is the use of simulation models, where a model is numerically solved and result data are gathered in order to forecast the characteristics of the model [1] . So, a simulation system is a computational model that mimics the behavior of a (real or conceptual) complex system (such as manufacturing systems), supporting the analysis, experimentation and prediction of its performance in different scenarios without having to interact with the real system. The main goal of a simulation system is to model a system using a set of parameters, variables and mathematical relating equations, encoding its behavior to compute its performance according to certain stimuli or scenarios. Obviously, this computational representation of the system is conditioned by the type of analysis, which establishes the type of key characteristics and behaviors that must be considered, usually ignoring other details that do not affect the proposed analysis.

For context, it is important to clarify that this paper focuses on the study of discrete and multistage

Proceedings of the Joint Ontology Workshops (JOWO) - Episode XI: The Sicilian Summer under the Etna, co-located with the 15th International Conference on Formal Ontology in Information Systems (FOIS 2025), September 8-9, 2025, Catania, Italy
*Corresponding author.
†These authors contributed equally.
benavens@uji.es (S. Benavent-Nácher); stefano.borgo@cnr.it (S. Borgo); rosado@uji.es (P. R. Castellano); francesco.compagno@unitn.it (F. Compagno)
0000-0002-4091-542X (S. Benavent-Nácher); 0000-0001-6001-2765 (S. Borgo); 0000-0001-9822-9484 (P. R. Castellano); 0000-0003-1002-608X (F. Compagno)

manufacturing systems, specifically on the use of simulations during its design to predict and analyze the material flow and the productivity, as has been explored in previous works [2, 3, 4]. During these previous experiences, the difficulty of verifying and validating the simulation model has been repeatedly confirmed. Although in some domains verification and validation (V&V) are often confused or considered synonymous, in this work they are clearly differentiated. Verification refers to checking the completeness and correctness of the model based on syntactic principles or constraints that, in the case of a simulation system model, allow its compilation. On the other hand, validation refers to ensuring that the model adequately represents reality, the reference system, through a semantic nature check. The specification of simulation languages generally includes constraints that support the identification of certain errors in the definition of the models, but they are purely syntactic and address very general issues. The lack of mechanisms to include additional constraints tuned to the domain under study limits the V&V of simulation models.

Faced with some of these limitations of traditional simulation languages and methodologies, a promising solution currently under investigation is the integration of the ontological modeling and the use of the reasoning capabilities of this domain into the design and development of simulation systems. As a first step in this direction, this paper presents some key conceptual basis for an ontological description of simulation systems, specifically those oriented to the analysis of manufacturing systems. The set of conceptual classes and their main relationships and characteristics are textually described and also depicted by diagrams. This first conceptual approach, which in the future will lead to the development of a complete and axiomatized ontology, adopts the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [5] as a foundational ontology, in order to assuring a greater consistency of the proposal according to well-defined and reliable DOLCE classes and relationships.

The paper is structured as follows. Section 2 briefly summarizes some relevant previous works and proposals. Section 3 presents the proposal, conceptually addressing various aspects of both simulation systems and simulated manufacturing systems. Furthermore, in order to offer an initial approach to its practical application, section 4 includes a brief description of the methodology in which the proposed classes will be used, and some examples of errors that this V&V resource is intended to detect. Finally, section 5 presents the main conclusions of the work.

2. Previous works

The development and modeling of manufacturing systems have been widely studied from a conceptual point of view focusing on different dimensions of this type of system (construction, operation, control, etc.) and giving rise to multiple works and standards, such as [6, 7, 8]. Most of these rules establish definitions using natural language, but lack the formality necessary to be interpreted by computers. Over the last two decades, different ontologies have been also developed to formally define the main concepts and relationships that characterize manufacturing systems [9, 10, 11]. Beyond general ontologies on the manufacturing domain, other works focus on more specific aspects, such as the definition of resources [12], the characterization of their capabilities [13], or the geometric definition based on features [14, 15, 16]. Many works address the combined use of ontologies and simulation from a generic perspective. Some of these works, such as [17], compare ontologies and simulations, also studying their combined or complementary use. Some methodologies use ontologies during initial steps of simulation system design to formulate a conceptual model, as presented in [18], while other proposals define the ontological model from a defined simulation system to compare models, query the model or reusing some information, as presented in [19]. However, no works have been found applying this ontological approach to the V&V of simulation models focused on the analysis of manufacturing systems.

Alternatively, some recent works address validation of manufacturing simulation systems based on the integration of SysML [20] (language widely used in systems modelling) in simulation design and modeling, as presented in [21, 22]. Most of these proposals are based on the definition of SysML profiles as a domain-specific modelling language (DSML). For example, methodology presented in [4] includes the design and validation of simulation systems using different SysML profiles [23, 24] to later be

automatically translated into a simulation language (Modelica), where simulations can be executed. It is important to note that the definition of these profiles includes rules implemented with Object Constraint Language (OCL) [25] that can be checked in the model to which the profile is applied. However, the modeling with SysML does not fully meet the specific needs of a simulation system. This limitation has prompted the exploration of other alternatives like the integration of ontological modeling and reasoning, but the abstraction and conceptualization effort in this type of SysML-based proposal has been considered and partially reused during the development of the ontology-based proposal.

3. An ontological approach to manufacturing systems simulation

This section presents a brief description of a set of key classes and relationships defined to support the V&V of systems simulations, specifically those oriented to the analysis of manufacturing systems. In the current state of development, these classes are focused on the representation of the simulation system, supporting a model verification enriched with additional checks according to the manufacturing domain semantics. This set of classes and relationships could also be extended to represent a particular manufacturing system to be analyzed (reference system), increasing the validation capabilities, but this work is still in progress.

Throughout this section, all classes are named using italics to facilitate their identification. Moreover, in addition to the textual descriptions, some diagrams are depicted using UML notation in order to graphically represent the main classes and relationships here considered or proposed. In these diagrams, DOLCE classes are depicted with grey rounded rectangles, while the new classes are represented with white regular rectangles. Some diagrams exceptionally include class instantiation examples depicted as ovals.

3.1. Systems of interest identification and modeling

As mentioned previously, the focus of this work is the V&V of models that represent simulation systems defined to analyze manufacturing systems. In this work, a model is understood as the formal representation of an entity of interest, that is, a set of information that captures a selected part of the structure and behavior of such an entity (a DOLCE *SocialObject*). To be shared, a *Model* must be encoded in a *RepresentingThing* [26] (a DOLCE *PhysicalObject*), for example, a written report (on paper or any other material support) or digital files, among others.

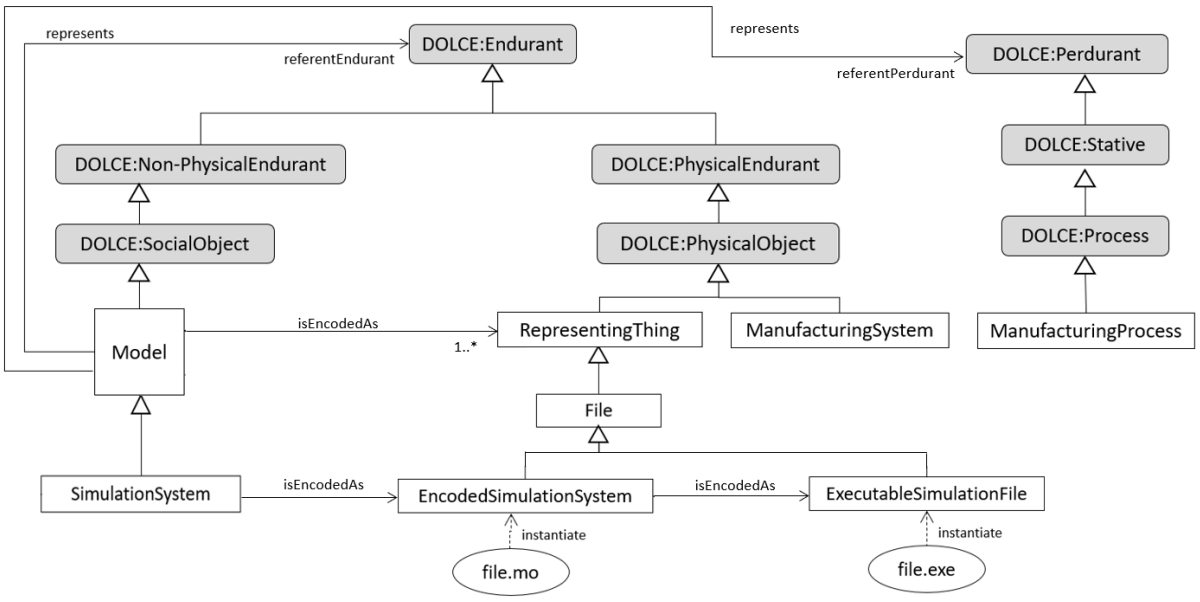

Figure 1: Model representing different endurants and perdurants of interest.

With respect to the type of reality to be modeled, this work is focused on software *SimulationSystems* understood as an algorithm, that is, a type of model (a subclass of *Model*, thus of *SocialObjet*), that can be compiled and executed to analyze a *ManufacturingProcess* (a DOLCE *Perdurant*) that is executed by a *ManufacturingSystem* (a DOLCE *PhysicalObject*). Given this broad focus, *Model* should include information about both an *Endurant* (with the role *referentEndurant*) and a *Perdurant* (with the role *referentPerdurant*), as depicted in Figure 1.

Focusing on the characteristics of the *SimulationSystems*, like any type of *Model*, they must be encoded in *RepresentingThing*, in this case a type of digital *File* represented by the class *EncodedSimulationSystem*. For example, in the Modelica language, the simulation system model is a computable file with name-extension "file.mo". This type of model is usually user-defined, but it can't be run directly. An intermediate compilation is necessary to obtain the *ExecutableSimulationFile*, that is, the executable file (e.g. "file.exe"), whose execution returns the process simulation.

3.2. Simulation system architecture

Since the objective of the proposal is the V&V of simulation systems, the paper is focused on characterizing the *SimulationSystem* at the concept level, and not its explicit implementation. Adopting a modular architecture, common in different modeling and simulation paradigms (object-centric orientation, functional or event-based paradigms, etc.), the *SimulationSystem* (complete algorithm that can be executed) can be built from previously and separately defined models (*PartialSimulationModels*, as another specialization of *Model*) that can not be solved autonomously (only as part of a *SimulationSystem*). Moreover, this work adopts some conceptual bases of a very widespread and multi-domain simulation language: Modelica. *ModelicaModel* represents any *Model* that adopts the principles of Modelica, and two specializations are proposed: *ModelicaPartialModel* (specialization of *PartialSimulationModels*) and *ModelicaSimulationSystem* (specialization of *SimulationSystem*, so it represents a *ModelicaModel* that can be compiled and executed). Moreover, as shown in Figure 2, a *ModelicaSimulationSystem* is generally composed by simpler *ModelicaPartialModels*, and a *ModelicaPartialModels* can also be composed by other *ModelicaPartialModels* (recursive composition relation in Figure 2).

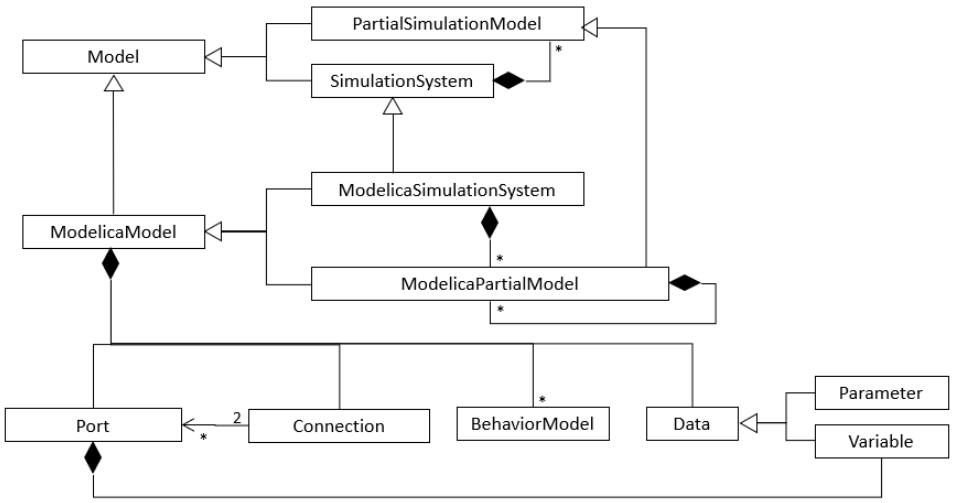

Figure 2: General structure of a Modelica simulation model.

Detailing the characteristics of a *ModelicaModel*, this type of model can contain the following elements: a) *Data*, both *Parameters* (with constant values known before the execution) and *Variables* (data whose value is computed and updated over the simulated time during the execution); b) *Connections* between its components (parts), connecting *Ports* (a set of *Variables* and the direction of the data flow) owned by them; and c) *BehavioralModels* to represent the behavior of the entity, encoded using different strategies, as detailed in next subsection.

3.3. Simulation of a manufacturing system

The design of a simulation system is closely conditioned by the type of system to be represented and the type of analysis to be performed. Important issues such as the structure, the type of parameters and variables considered, the data flows defined and the type of emulated behaviors depend largely on these conditions. From this point, the proposal is focused on the specific case of simulation systems defined under Modelica principles to analyze the flow of materials throughout a discrete and multistage manufacturing system that modifies some physical characteristics of the processed product. Furthermore, although there are different alternative approaches to designing the simulation system (e.g., focused on processes or activities), the approach adopted in this work aims to replicate the resource structure of the manufacturing system, designing a simulation system structure parallel to the physical structure of the analyzed manufacturing system. This consideration conditions the rest of the proposal, which should be adapted if alternative approaches are explored. This particular case is represented by the *ManufacturingSimulationSystem* class (specialization of *ModelicaSimulationSystem*). Considering this type of simulation, a *ManufacturingSimulationSystem* is composed of *ModelicaPartialModels* representing some environmental element of considered scenario (*Environment_sim*), or manufacturing resources that processes products (*ProcessingResource_sim* class as a type of *ManufResource_sim*), as depicted in Figure 3.

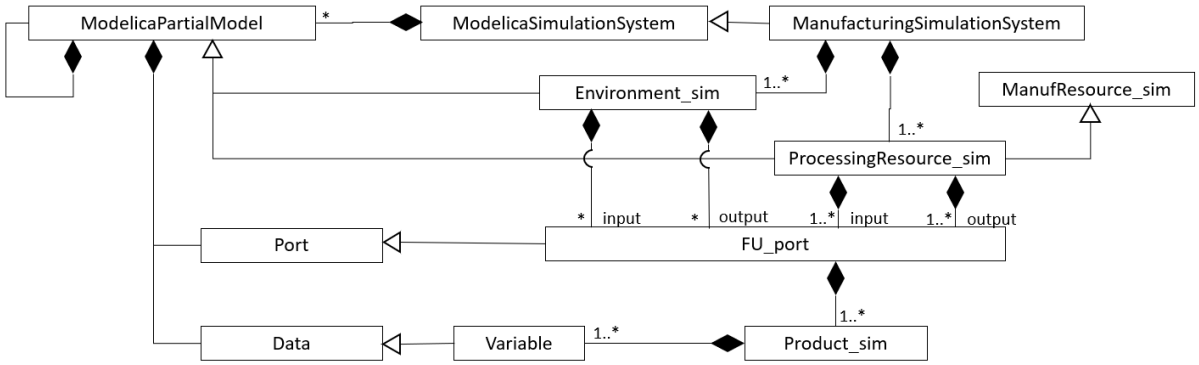

Figure 3: Manufacturing elements represented in the simulation system.

In the context of manufacturing systems, there are multiple classifications that identify different types of resources based on, for example, their primary function in manufacturing or how they interact with the product and affect its characteristics. Aligned with some of these classifications, various specializations of the *ManufResource_sim* class are proposed in Figure 4: *ControlResource_sim* represents resources oriented to the monitoring and decision making (without participating in the material flow simulation); *ProcessingResource_sim* represents any resource that is traversed by the simulated flow of materials; *TransformativeResource_sim* represents resources that change some variables of the product; and *LogisticalResource_sim* participates in the simulated flow of materials without changing the product characteristics, representing warehouses or transport simulations, for example.

Focused on the analysis of the materials flow, any *ProcessingResource_sim* must have ports (*FU_ports*) through which to exchange at least data relating to the products units flowing according to the process plan (Figure 3). These product data are grouped into the *Product_sim* class, including variables that represent key characteristics (from the analysis point of view) of the particular product/s that are flowing through a certain part of the simulated manufacturing system at a particular instant of simulated time. These variables can represent, for example, the identifier of the particular product and its components, the type of product and components, etc. *Product_sim* constitutes the main flow unit considered during the design of a *ManufacturingSimulationSystem*.

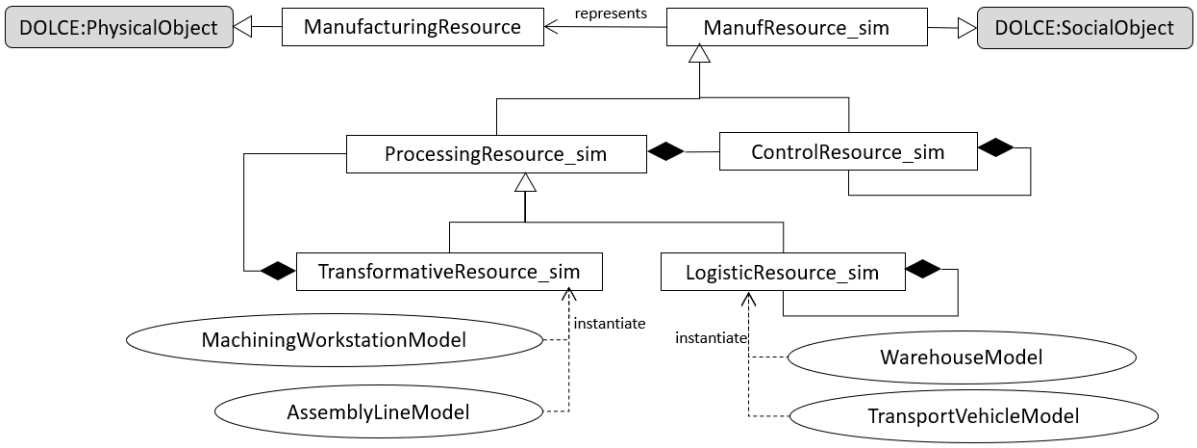

Figure 4: Types of simulated manufacturing resources.

3.4. Behavior of simulated manufacturing resources

Different types of simulation models can be identified based on the type of behavior represented. Figure 5 presents a proposed classification that, while presenting generic classes, is exemplified with examples from the manufacturing systems domain. *NonBehavioralModel* contains simulation elements that are composed of data only, and they lack behavior descriptions. Data can be either parameters (invariant over the simulated time) or variables whose value is computed or modified by other elements of the simulation system. For example, considering the adopted approach (replicating the resource structure of the manufacturing system) for the analysis of the material flow, the owned behavior of resources like tools or fixtures are not necessarily considered, so they can be modeled only as a set of data (e.g. identification, availability, geometric characteristics, etc.) without behavior. On the other hand, a *BehavioralModel* includes the representation of some type of behavior, whether it is explicitly defined in the model (*ModelWithExplicitBehavior*) or emerges from the behavior of its components and the interactions between them (*ModelWithEmergentBehavior*). A typical example of *ModelWithExplicitBehavior* is the representation of a workstation, defining how it works over time and detailing its interactions with other resources or products. The behavior of a manufacturing line or a complex manufacturing system is not defined explicitly, but it emerges from the interaction of the different modeled workstations in the simulation system. Other aspects of the resources, such as the participation of human operators, are not considered in this work.

Focusing on the explicit definition of behaviors, different types of *BehaviorModels* can be considered. As depicted in Figure 5, three levels of complexity are distinguished in the definition of simulated behaviors: simple, conditioned and complex behaviors. *SimpleBehaviors* can be expressed with mathematical equations to obtain an analytical solution for some variables from some parameters. In *ConditionedBehaviors*, different alternatives (set of simple behaviors) are defined depending on some state variables, so in a particular instant during executions one of the alternative behaviors will govern the system, while others are ignored. *ConditionedBehaviors* can be represented through algorithms (*ConditionalClause*) that are typically governed by if-clauses, although other logical clauses could be also defined (e.g. when-clauses). However, *ConditionedBehaviors* can be also represented with *StateMachines*, considering a set of possible states and the events that trigger state changes in the system, as described in more detail in the next subsection. Finally, complex behaviors are related with previously described emergent behaviors, because they are too complex to obtain an analytical solution or a set of equations or algorithms to be directly represented.

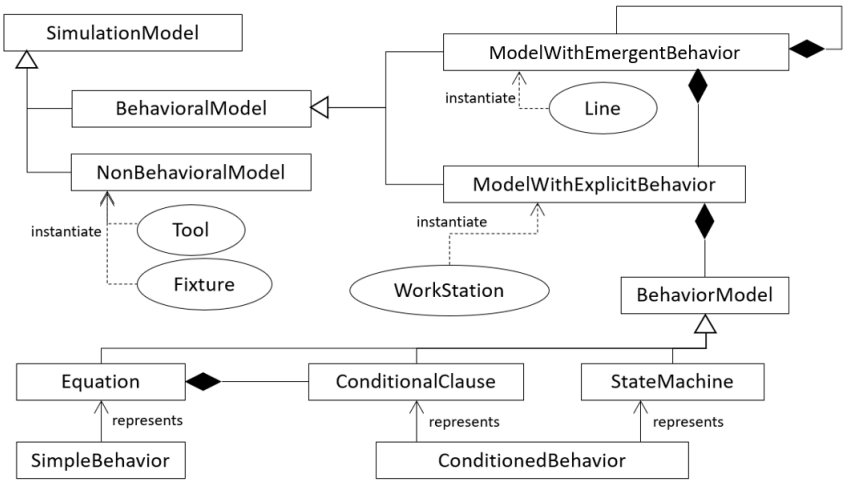

Figure 5: Types of simulation models according to the represented behavior.

3.5. Manufacturing process modeling

A key part of modelling a manufacturing system is the characterization of the commonly named manufacturing process, generally understood as the sequence of activities or operations executed to obtain a certain product using the necessary resources. However, DOLCE proposes various specializations of the *Perdurant* class that would allow alternative representations of the manufacturing process adopting different points of view or approaches. This work proposes to use *StateMachines* to model a simple run of the system (a DOLCE *Accomplishment*), but including closed loops so, during the simulation execution, the described *Accomplishment* can be repeatedly executed, obtaining the simulation of a *ManufacturingProcess* (a DOLCE *Process*) and considering several units of the same product type.

As depicted in Figure 6, a *StateMachine* is a type of *Model* focused on representing an *Accomplishment*. A *StateMachine* is composed of at least two *StateRepresentation* and at least one *Transition*. A *StateRepresentation* is a model part (also *SocialObjects*) that represents a *State* (e.g. being executing a task or just idle, waiting for a new order or task request, etc.). Each *StateRepresentation* has an associated behavior, defined through a set of equations. A *Transition* is a model part that relates two different *StateRepresentations* with a specified direction to represent a possible change of the active state between the current and the subsequent one. Each *Transition* must include a *TriggerRepresentation* (represents a trigger, that is an event or an *Achievement* that activates the *Transition*) and, sometimes, a *Guard* (a logical expression defined to encode certain conditions that must be met to trigger the *Transition*).

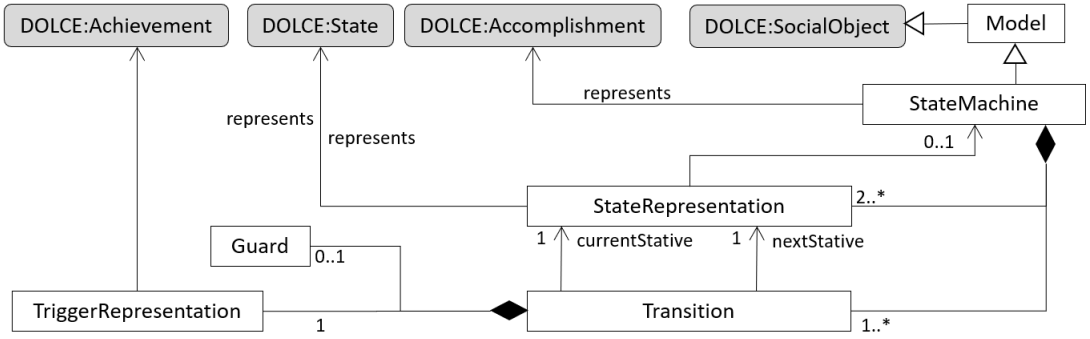

Figure 6: State machine models for manufacturing process representation.

The main advantage offered by this type of model is that it allows for the generic representation of different temporal parts (with duration) associated with states, which change based on a set of events (specific instants). The integration of this type of model in executable models (such as simulations)

allows for the determination of the active state at each instant of the simulated time and, in addition, allows for the assignment of a different type of behavior to each of the states considered.

4. Examples of improvement on the state of the art

The proposal presented in this paper is part of a research line whose goal is to be able to transform a simulation model into an ontological model (instantiated) on which to run reasoners that take into consideration the axioms defined in the ontology. Although the detailed development of this content (complete ontology, model transformation, methodology) is outside the scope of this paper, the developed methodology is briefly described below as an application framework of the proposal. On the one hand, a Python code has been implemented to automate the transformation of a Modelica model into an ontological model, including the definition of the necessary classes and instances. On the other hand, a set of OWL axioms and SHACL constraints supporting the basic characteristics of the domain-specific concepts have been defined, so reasoners can be run on the transformed models to check the constraints defined in ontological language. One of the main reasons for using SHACL in addition to OWL-based validation is to be able to define cardinality-related constraints that are not compatible with the open-world assumption of other ontological languages. Some of the models and algorithms developed can be consulted in [27].

Based on this partially developed methodology and the consideration of the set of classes presented in this work, various experiments have been developed in order to test the proposal and identify its advantages. This section summarizes some practical examples to identify some current limitations on the traditional simulation system validation that are overcome or improved with the integration of ontological modeling (using some of the previously presented classes) and reasoning, checking if the defined axioms are met.

A first example is the inter-model validation. Simulation models are typically hand-implemented in parallel with specification models, without automated transformations or formal dependencies between them. This strategy increases the risk of introducing discrepancies between the designed manufacturing system (and supported processes) and its simulation model. Ontological-based validation enables the comparison of both models (once they have been transformed into ontological models), checking if individuals of the design model has an equivalent individual in the simulation domain. To proceed with this check, dependences between different general classes (from both design and simulation domains) must be defined (and inherited in the different considered specializations), supporting the alignment between both representations of the same system.

A second use of this proposal is the validation of the process plan simulation. Taking the specification of the product process plan as a reference, simulation model must contain: a) enough elements (simulated manufacturing resources) planned operations to manufacture a product; b) direct connections between these elements to simulate the planned sequence of operations. The verification inherent to simulation languages is limited to verifying whether each connection is possible between the related ports, but the validation of the alignment with the process plan is not possible.

Finally, a third case of use is presented to validate behavior simulation of manufacturing resources. As mentioned above, simulations typically combine different levels of aggregation in systems modeling, explicitly defining behavior in the most atomic components (*ModelWithExplicitBehavior*) and establishing the necessary relationships and interactions from which the behavior of more complex systems (*ModelWithoutExplicitBehavior*) emerges. Adopting this differentiation, the definition of libraries with reusable models, specially focused on the behavioral elements, is proposed. The use of predefined elements allows to assure the well-definition of their behaviors and limits the risk of introducing errors during user system construction, whose behavior emerges from the library elements and the user-defined connections. Moreover, limiting the behavior definition only on the atomic level also prevents the overlaps or contradictions in the overall behavior of the system.

5. Conclusions

Building models for the analysis of complex systems requires mechanisms to assure their consistency and well-construction. The validation mechanisms supported by typical simulation languages are limited, while the integration of ontological modeling and associated reasoning mechanisms offer significant advantages. Aligned with this approach, this work presents an ontological approach to simulation modeling for the analysis of manufacturing systems.

On the one hand, the set of concepts related to the simulation system, its structure, and the definition of simulated behaviors allow for the verification of the simulation system and its well-construction. On the other hand, the integration of concepts from the manufacturing domain facilitates the alignment between the simulation model and the actual manufacturing system, also supporting validation mechanisms to check if the simulation model adequately represents the analyzed system.

From these foundations, a complete ontology must be formally defined, including the axioms that allow consistency checking using reasoners. The main goal of the global proposal is to convert a simulation system model (and other types of models) into an ontological model with classes and individuals, on which reasoners can be run to verify whether the axioms defined in the ontology are met. At the time of publishing this work, some initial experiments have been already developed obtaining very promising results, although these study cases are still under development.

Moreover, to overcome limitations of the open-world assumption inherent to some ontological languages, the complementary implementation of SHACL constraints is an interesting solution that must be explored in order to cover a more complete and varied range of possible errors and inconsistencies in simulation models.

Finally, other lines of work aim to extend the proposal in two main directions: a) to include the classes and axioms necessary to check the detailed definition of behaviors; b) to include concepts related to different types of analysis, with particular interest on those simulation systems that incorporate the analysis of geometric quality.

Acknowledgments

This work has received support from Universitat Jaume I (Spain), through grants for international stays at other research centers. Grant number: E-2024-13.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. M. Law, W. D. Kelton, W. D. Kelton, Simulation modeling and analysis, volume 3, Mcgraw-hill New York, 2007.

[2] S. Benavent Nacher, P. Rosado Castellano, F. Romero Subiron, J. V. Abellán-Nebot, Multidomain simulation model for analysis of geometric variation and productivity in multi-stage assembly systems, Applied Sciences 10 (2020) 6606.

[3] S. Benavent Nacher, P. Rosado Castellano, F. Romero Subiron, J. V. Abellán-Nebot, Control strategies comparison for a multi-stage assembly system using simulation, in: IOP Conference Series: Materials Science and Engineering, volume 1193, IOP Publishing, 2021, p. 012089.

[4] S. Benavent Nácher, Modelado y simulación híbrida de sistemas de fabricación multi-etapa orientado a la evaluación de la calidad geométrica y la productividad, Ph.D. thesis, Universitat Jaume I, Castelló de la Plana, Spain, 2024.

[5] D. Porello, L. Vieu, W. Terkaj, S. Borgo, F. Compagno, E. M. Sanfilippo, Dolce in owl: The core theory, in: Proceedings of the 8th Workshop on Foundational Ontology (FOUST VIII) - Joint

Ontology Workshops (JOWO) - Episode X: The Tukker Zomer of Ontology, and satellite events, CEUR Workshop Proceedings, Twente, Netherlands, 2024.
[6] I. 22400-1:2014, Automation systems and integration — key performance indicators (kpis) for manufacturing operations management (2014).
[7] I. 16739-1:2024, Industry foundation classes for data sharing in the construction and facility management industries management (2024).
[8] I. 62264-1:2013, Enterprise-control system integration (2013).
[9] S. Borgo, P. Leitão, Foundations for a Core Ontology of Manufacturing, Springer US, Boston, MA, 2007, pp. 751–775.
[10] Z. Usman, R. I. Young, N. Chungoora, C. Palmer, K. Case, J. A. Harding, Towards a formal manufacturing reference ontology, International Journal of Production Research 51 (2013) 6553–6572.
[11] V. Zaletelj, E. Hozdić, P. Butala, et al., A foundational ontology for the modelling of manufacturing systems, Advanced Engineering Informatics 38 (2018) 129–141.
[12] E. M. Sanfilippo, W. Terkaj, S. Borgo, Ontological modeling of manufacturing resources, Applied ontology 16 (2021) 87–109.
[13] L. Solano, F. Romero, P. Rosado, An ontology for integrated machining and inspection process planning focusing on resource capabilities, International Journal of Computer Integrated Manufacturing 29 (2016) 1–15.
[14] E. M. Sanfilippo, S. Borgo, Feature-based modelling and information systems for engineering, in: Congress of the Italian Association for Artificial Intelligence, Springer, 2015, pp. 151–163.
[15] F. Romero Subirón, P. Rosado Castellano, G. M. Bruscas Bellido, S. Benavent Nácher, Feature-based framework for inspection process planning, Materials 11 (2018) 1504.
[16] N. Anjum, J. A. Harding, R. I. Young, K. Case, Manufacturability verification through feature-based ontological product models, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 226 (2012) 1086–1098.
[17] C. Turnitsa, J. J. Padilla, A. Tolk, Ontology for modeling and simulation, in: Proceedings of the 2010 Winter Simulation Conference, IEEE, 2010, pp. 643–651.
[18] P. Benjamin, M. Patki, R. Mayer, Using ontologies for simulation modeling, in: Proceedings of the 2006 winter simulation conference, IEEE, 2006, pp. 1151–1159.
[19] K. Grolinger, M. A. Capretz, J. R. Marti, K. D. Srivastava, Ontology–based representation of simulation models (2012).
[20] System Modeling Language, Technical Report 1.6, Object Management Group, 2019.
[21] C. Nigischer, S. Bougain, R. Riegler, H. P. Stanek, M. Grafinger, Multi-domain simulation utilizing sysml: state of the art and future perspectives, Procedia CIRP 100 (2021) 319–324.
[22] L. McGinnis, E. Huang, K. S. Kwon, V. Ustun, Ontologies and simulation: a practical approach, Journal of Simulation 5 (2011) 190–201.
[23] S. Benavent-Nácher, P. Rosado Castellano, F. Romero Subirón, J. V. Abellán-Nebot, Sysml4ta: A sysml profile for consistent tolerance analysis in a manufacturing system case application, Applied Sciences 13 (2023) 3794.
[24] S. Benavent-Nácher, P. Rosado Castellano, F. Romero Subirón, Sysml4gdpsim: A sysml profile for modeling geometric deviation propagation in multistage manufacturing systems simulation, Applied Sciences 14 (2024) 1830.
[25] Object Constraint Language, Technical Report 2.4, Object Management Group, 2014.
[26] R. Mizoguchi, S. Borgo, An ontology of representation, Applied Ontology (2025) 15705838251337943.
[27] F. Compagno, S. Benavent Nácher, Github repository: modelica_constraints. url: https://github.com/kataph/modelica_constraints (2025).