Observable Semantics for Characterising Consistency Between Heterogeneous Models

Henriette Färber[1](✉), Romain Pascual[1,2], Terru Stübinger[1], and Mattias Ulbrich[1]

[1] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{henriette.faerber, stuebinger, ulbrich}@kit.edu
[2] MICS, CentraleSupélec, Université Paris-Saclay, Saclay, France
romain.pascual@centralesupelec.fr

Abstract. The design of complex cyber-physical systems increasingly relies on heterogeneous, multi-domain models, each capturing different system aspects. As these models evolve independently, inconsistencies may arise. Such inconsistencies remain difficult to detect due to gaps between the modelling domains. We propose to bridge this gap by introducing a formal notion of *observables*, inspired by physics, as measurable system properties, such that every model can constrain the possible values of an observable. We define a semantic framework where observables induce consistency relations and show that any such relation can be expressed using suitable observable semantics. To capture realistic engineering scenarios, we extend the framework with *meta-model spanning* and *compound* observable semantics, enabling the modular construction of complex constraints. We also provide encodings of these observable semantics back into the original framework, preserving the original results and showing that extensions remain expressively equivalent to initial ones. Finally, we discuss some practical implications of our framework, namely how observables can support cross-domain communication, separation of concerns, and extensibility, allowing consistency requirements to evolve as part of the modelling process.

Keywords: Model-driven development · Model consistency · Observable Semantics · Formal foundations · Cyber-physical systems

1 Introduction

The design of modern cyber-physical systems (CPS), such as smart grids or autonomous vehicles, requires integrating diverse domain knowledge from mechatronics, system engineering, and computer science, each with distinct methodologies and modelling principles [18]. Their increasing complexity demands scalable methods for managing this heterogeneity throughout the development process.

Model-driven development (MDD) [2, 13] divides system descriptions into purpose-oriented abstractions called models [25, p. 131–133]. This modularisation enables engineers from different domains to work independently. However, models may overlap in their descriptions, leading to contradictions that hinder system realisation. *Consistency management* [19, 24] addresses the challenge of preserving consistency between interdependent models as they evolve.

This challenge is amplified when overlapping models are defined using heterogeneous formalisms. For example, consider two models of an electric vehicle (EV): one defines the geometry of the wheelbase, while the other is a simulation of the steering behaviour. Both depend on the front-to-rear axle distance, yet this information may be encoded differently or remain implicit. Besides, domain-specific modelling languages (DSMLs) use distinct syntactic and semantic conventions, and encode implicit assumptions. As a result, establishing consistency at the syntactic level becomes impractical, requiring complex coupling mechanisms or a shared formal representation, both of which demand considerable effort and cross-domain expertise.

We propose a shift in perspective and define consistency semantically via the *system properties* that models constrain rather than syntactic correspondences. Such constraints restrict possible realisations of the system, narrowing the set of admissible implementations. We embody this idea by introducing *observables*, inspired by physics, where observables are measurable quantities such as energy or temperature. The semantic abstraction of observables facilitates reasoning about model consistency across differing formalisms and creates a shared interdisciplinary cognitive space for domain experts [15]. Formally, let P be the (hypothetical) set of all possible system realisations. An observable is a function $o : P \to V_o$, where V_o is the value space. Each model m restricts admissible realisations and, by extension, the set of admissible values $o(m) \subseteq V_o$. In the EV example, both the geometry and simulation models constrain the axle distance. Any observable o naturally induces a consistency criterion: a set of models is consistent if there exists some common $v \in V_o$ in their observable values.

Rather than a universal semantic abstraction to cover all aspects of a CPS, which is both unrealistic and impractical, we propose maintaining a *collection of observables*, each focusing on a distinct property. The advantages are threefold: (1) *Separation of concerns*: Observables can be defined and reasoned about independently by domain experts. (2) *Traceability*: Inconsistencies can be linked to the observable(s) they violate. (3) *Extensibility*: New observables can be added when previously unknown dependencies are found during the design process.

Contributions We present observables as a novel, flexible formal framework for consistency management in MDD. Observables induce abstract semantics over models, supporting engineers in both informal and formal consistency assessments, and may serve as guidelines for tooling. Building on the formal framework of Pascual et al. [20] describing consistency in MDD, we demonstrate how observables unify semantic aspects across heterogeneous models. The main contributions of this paper are:

(1) A formal definition of observable semantics as abstract semantics in the sense of [20] (Sect. 4).
(2) A completeness result showing that any consistency relation between models can be related to an observable (Prop. 2).

(3) Two extensions for constructing observable semantics over multiple models and over other observables (Sect. 5).
(4) A reduction from the extended constructions back to the base framework, ensuring the already proven properties (Sect. 5.5).
(5) A qualitative validation discussing the potential of observables in engineering workflows (Sect. 6).

2 Related Work

Consistency management for heterogeneous models has been extensively studied in MDD. Fradet et al. [10] proposed a formal framework for UML-like diagrams with explicit constraints and decision procedures to ensure structural consistency. Knapp and Mossakowski [17] surveyed consistency in UML/OCL, identifying structural constraints, transformation rules, and distributed semantic models as dominant strategies. Stünkel et al. [26] developed a unifying graph-based framework for consistency management in evolving multi-model systems, leveraging inter-model correspondences and constraint propagation.

Several works investigate semantic approaches to inter-model consistency. Chen et al. [6, 7] introduce *semantic anchoring*, a mapping from the abstract syntax of DSMLs to the abstract syntax of semantic units, e.g., abstract state machines (ASMs) [12] or even another DSML [5]. While semantic units are compositional [8], they encode executable behavioural semantics, which limits their general applicability for broader consistency analysis. Other approaches use ontologies as the shared semantic space [9, 21], translating and merging models into a single ontology. In these cases, the expressiveness for defining consistency is inherently constrained by the underlying ontology language. For example, Perin and Wouters [21] extend the OWL language to capture behavioural aspects, while Dibowski and Massa [9] tailor their approach to smart building design. Finally, some approaches exploit mathematical constructions as a shared semantic foundation. For instance, Grönniger et al. [11] uses the *system model* from Rumpe [22], a characterisation of object-oriented systems as mathematical theories, to provide the semantics of associated DSMLs, while Calegari et al. [4] use the theory of institutions to provide abstract model semantics. However, their focus is on translation and equivalence of models, rather than inter-model consistency as we pursue here.

A shared limitation in these approaches is that they rely on meta-level artefacts – such as syntactic correspondences, traceability links, or propagation rules – to define consistency. Such artefacts are often complex to define and maintain, especially as models evolve independently. In contrast, our approach investigates the shared system properties – *observables* – that models semantically constrain, inspired by observational semantics and indistinguishability under observation in algebra and coalgebra [14, 23]. Our notion of observables connects to view-based modelling, where dependencies and overlap across model views are analysed semantically [17]. The significant difference is that observables allow disregarding the internal structure of models (and meta-models), focusing on the constrained aspects of the systems.

3 Background & Context

Before formally introducing observables in Sect. 4, we briefly summarise the formal framework on which our definitions are based. Our work builds on the VITRUVIUS approach [16], a model-driven representation technique for heterogeneous systems, and the formalisation of model consistency by Pascual et al. [20]. We refer the reader to the original sources for a more comprehensive discussion.

Notation 1 *Given a set I and an I-indexed family $(X_i)_{i\in I}$ of sets, we write $\prod_{i\in I} X_i$ or simply $\prod_I X_i$ for the Cartesian product of the sets $(X_i)_{i\in I}$, which consists of tuples $(x_i)_{i\in I}$ where for every $i \in I$, $x_i \in X_i$. When the context is clear, we write x_I, or simply x, for a tuple $(x_i)_{i\in I}$ and access its i-th coordinate as x_i. Similarly, we write $\bigcap_I X_i$ for the intersection of all the sets $(X_i)_{i\in I}$.*

3.1 The V-SUM Approach

The VITRUVIUS approach by Klare et al. [16] relies on the concept of *virtual single underlying models* (V-SUMs). Such models externally behave like a single model but internally rely on a collection $(m_i)_{i\in I}$ of models. Each model belongs to its own meta-model M_i, such that we obtain a notion of V-SUM meta-models (V-SUMM), the meta-models for the V-SUMs. However, to ensure that a V-SUM behaves like a single model from the outside, the inside models must be kept consistent. As such, a V-SUMM also contains a description of a V-SUM's consistency. As our work does not depend on how meta-models specify models, we can follow the framework of Pascual et al. [20] and consider a meta-model as a set of admissible models with a V-SUMM then consisting of a collection of meta-models and a consistency relation.

Definition 1 (V-SUM meta-model – [20, Def. 2]). *A* V-SUM meta-model *is a pair $\mathcal{M} = (\prod_I M_i, CR)$ where each M_i is a meta-model (which we will treat as the set of models conforming to the meta-model) and $CR \subseteq \prod_I M_i$ is a consistency relation.*

3.2 Running Example

Consider a simplified description of an electric vehicle (EV) as a motivating example. An EV comprises several components – such as the vehicle body, engine, and battery – mounted on the EV's chassis. This chassis contains a suspension system, which connects the wheels to the chassis and is responsible for, e.g., absorbing shocks while driving. As such, the suspension system imposes an upper bound on the overall admissible EV weight $w_{\max} \in \mathbb{R}_+$. The EV is described by four models.

- A *motion model* $m_{\text{mot}} \in M_{\text{mot}}$ that captures the electrical components responsible for propulsion, including the battery and engine.
- A *CAD model* $m_{\text{geo}} \in M_{\text{geo}}$ that describes the vehicle body's geometry: its shape and structure.

- A *materials model* $m_{\text{mat}} \in M_{\text{mat}}$ that defines the mechanical properties of the materials used in the vehicle body, such as their density.
- A *chassis model* $m_{\text{ch}} \in M_{\text{ch}}$ that encodes suspension characteristics and specifies a maximum admissible weight $w_{\max} \in \mathbb{R}_+$.

The total weight must not exceed $w_{\max}$ to ensure a valid EV configuration. It consists of the weight of the motion subsystem (from m_{mot}) and the body (computed as volume from m_{geo} multiplied by density from m_{mat}). For simplicity, we treat the body as a single aggregated component.

3.3 Heterogeneous Model Consistency

We consider models $m_i \in M_i$, indexed by a finite set I, where each M_i is a meta-model. We write $\prod M = \prod_I M_i$ for the product of these meta-models. From a consistency relation $CR \subseteq \prod M$, we define *equiconsistency*, grouping models that behave identically with respect to consistency:

Definition 2 (Equiconsistency – [20]). *Equiconsistencies are the relations* $\sim_i^{CR} \subseteq M_i \times M_i$ *given by* $m_a \sim_i^{CR} m_b \Leftrightarrow CR^{\nabla i}(m_a) = CR^{\nabla i}(m_b)$*, where*

$$CR^{\nabla i}(\nu) := \{(m_j)_{j \neq i} \in \prod_{j \neq i} M_j \mid CR(m_1, \ldots, m_{i-1}, \nu, m_{i+1}, \ldots, m_n)\} \enspace .$$

To reason about consistency semantically, Pascual et al. [20] introduced *abstract* semantics, which map models to a semantic domain.

Definition 3 (Abstract Semantics – [20, Def. 7]). *An* abstract semantics *is a mapping* $[\![\cdot]\!]\colon M \to S$ *from a meta-model* M *to a* semantic space S*.*

Quotienting by the kernel of such a semantics ($m \equiv m'$ iff $[\![m]\!] = [\![m']\!]$) yields a canonical semantic space $M/\!\equiv$ isomorphic to the image of $[\![\cdot]\!]$. Since equivalence relations form a complete lattice, the space of abstract semantics also forms a lattice $\mathcal{L}_{\text{sem}}^{M}$, allowing comparison and refinement. Abstract semantics also enable the definition of semantic consistency relations which can be lifted back into the modelling space:

Definition 4 (Semantic Consistency Relation – [20]). *Given abstract semantics* $[\![\cdot]\!]_i\colon M_i \to S_i$*, a* semantic consistency relation *is a relation* $SCR \subseteq \prod_I S_i$*, and* $m_i \in M_i$ *are (semantically) consistent if and only if* $SCR(([\![m_i]\!]_i)_{i \in I})$*.*

Definition 5 (Semantics-Induced Consistency – [20]). *A semantic consistency relation* $SCR \subseteq \prod_I S_i$ *yields a* semantics-induced *consistency relation* CR_{SCR} *such that* $CR_{SCR}(m_I) :\Longleftrightarrow SCR(([\![m_i]\!]_i)_{i \in I})$*.*

Thus, a semantic consistency relation should agree with the given CR, and abstract semantics are deemed *compatible* with CR if it induces CR from a semantic consistency relation. Among all such semantics, the *natural semantics* form the least informative compatible abstraction, capturing exactly the information required to decide consistency.

Definition 6 (Compatibility – [20, Def. 8]). *Given a V-SUM meta-model $\mathcal{M} = (\prod M, CR)$, a family of abstract semantics $[\![\cdot]\!]_i \colon M_i \to S_i$ is* compatible *with CR if and only if there exists $SCR \subseteq \prod_I S_i$ such that $CR_{SCR} = CR$.*

Definition 7 (Natural Semantics – [20, Def. 9]). *The semantics $[\![\cdot]\!]_i^{\text{nat}} \colon M_i \to M_i/{\sim_i^{CR}}$ are called the* natural semantics *for CR.*

Pascual et al. [20] proved that families of semantics $([\![\cdot]\!]_i)_{i \in I}$ where each $[\![\cdot]\!]_i$ is an element of the quotient sublattice $\mathcal{L}_{\text{sem}}^{M_i}/[\![\cdot]\!]_i^{\text{nat}}$ are compatible with CR. The result can actually be strengthened.

Theorem 1 (Lattice of Compatible Semantics – compare [20, Prop. 2]). *The semantics of $(\mathcal{L}_{\text{sem}}^{M_i}/[\![\cdot]\!]_i^{\text{nat}})_{i \in I}$ are the compatible semantics.*

Proof. From [20, Prop. 2], it remains to show that any family of semantics compatible with CR is composed of elements of the quotient sublattices $\mathcal{L}_{\text{sem}}^{M_i}/[\![\cdot]\!]_i^{\text{nat}}$. It suffices to prove that for every $i \in I$, $\equiv_i \subseteq \sim_i^{CR}$ where $\equiv_i$ is the kernel of $[\![\cdot]\!]_i$.

Let $m_a \equiv_i m_b$. Then, $[\![m_a]\!]_i = [\![m_b]\!]_i$. By compatibility $CR = CR_{SCR}$ for some SCR. Thus, for $m \in \prod M$, $CR(m) \iff CR_{SCR}(m) \iff SCR([\![m]\!])$. Then $CR^{\nabla i}(m_a)$ is the set

$$\left\{(m_j)_{j \neq i} \in \prod_{j \neq i} M_j \mid SCR([\![m_1]\!]_1, \ldots, [\![m_{i-1}]\!]_{i-1}, [\![m_a]\!]_i, [\![m_{i+1}]\!]_{i+1}, \ldots [\![m_n]\!]_n)\right\} .$$

Now, since $[\![m_a]\!]_i = [\![m_b]\!]_i$, $CR^{\nabla i}(m_a) = CR^{\nabla i}(m_b)$ and then $m_a \sim_i^{CR} m_b$.

4 Simple Observable Semantics

While it might be insightful to view observables as mappings $o \colon P \to V_o$, we will not further consider the set P of hypothetical systems and assume a set O of predefined observables about the system and, for each observable $o \in O$, a domain of possible values V_o.

Definition 8 (Observable semantics). *An observable $o \in O$ yields an I-indexed family of observable semantics given by functions $o_i \colon M_i \to \mathscr{P}(V_o)$.*

Intuitively, the observable semantics of a model m w.r.t. $o \in O$ is the set of values o may take for any system adhering to m. Since models typically permit multiple implementations, this semantics is set-valued.

Example 1. Consider the EV from Sect. 3.2. The total vehicle weight can be encoded with an observable w^{total} with value domain $V_{w^{\text{total}}} = \mathbb{R}_+$. Let w^{total} be an observable for total vehicle weight, with value domain $V_{w^{\text{total}}} = \mathbb{R}_+$. The motion model m_{mot} defines a lower bound $l \in \mathbb{R}_+$ on the weight, hence $w^{\text{total}}_{\text{mot}}(m_{\text{mot}}) = [l, \infty)$. As the model imposes no upper bound, all higher weights are allowed. Note that all observable semantics share the same codomain $\mathscr{P}(V_o)$, ensuring comparability across components.

4.1 Observational Consistency

Observables can serve as a proxy for a notion of "joint realisability" that engineers usually refer to when saying that the *system can be built* [3]. As discussed in [20], model consistency expresses the idea that a collection of models jointly describes at least one plausible system. We capture this idea using observable-induced consistency.

Definition 9 (Observable-induced consistency). *Any observable* $o \in O$ *induces a consistency relation* $CR_o \coloneqq \{m_I \in \prod M \mid \bigcap_I o_i(m_i) \neq \emptyset\}$.

Thus, each observable $o \in O$ yields an *observational V-SUM meta-model* $\mathcal{M} = (\prod M, CR_o)$.

Example 2. Let us consider the models $m_{\text{mot}}, m_{\text{geo}}, m_{\text{mat}}, m_{\text{ch}}$ from Sect. 3.2 and the observable w^{total} from Ex. 1. Let $l \in \mathbb{R}_+$ be the lower bound for total vehicle weight imposed by m_{mot}. We can further assume that m_{ch} specifies an upper bound $w_{\max} \in \mathbb{R}_+$, while m_{geo} and $m_{\text{materials}}$ place, on their own, no constraint. Then, the intersection of observable values is $[l, w_{\max}]$, and the models are consistent if and only if $l \leq w_{\max}$.

Proposition 1. *For any* $o \in O$*, the semantics* $(o_i)_{i \in I}$ *are compatible with* CR_o.

Proof. Consider the semantic consistency relation $SCR_o \coloneqq \{(s_i)_{i \in I} \in \prod_I \mathscr{P}(V_o) \mid \bigcap_I s_i \neq \emptyset\}$. Then, for any $m_I \in \prod M$,

$$m_I \in CR_o \iff \bigcap_I o_i(m_i) \neq \emptyset \iff (o_i(m_i))_{i \in I} \in SCR_o \ .$$

More abstractly, this result holds because observable semantics are specific instances of abstract semantics, and SCR_o is the associated semantical consistency relation in the sense of [20], enforcing non-empty intersection.

4.2 Consistency-Induced Observables

Having made the connection from observables to consistency, the natural question is to explore the reverse connection: Can consistency be encoded via an observable? Since observable semantics are a special kind of abstract semantics, the question boils down to seeing whether their construction suffices to encode consistency.

Notation 2 *For a model* $m_i \in M_i$*, we write* $p_{i,\{m_i\}}$ *for* $\{m_i\} \times \prod_{j \neq i} M_j$.

Definition 10 (Consistency-induced observable). *Let* $\mathcal{M} = (\prod M, CR)$ *be a V-SUM meta-model. The family of abstract semantics*

$$\Big(o[CR]_i \colon M_i \to \mathscr{P}(\prod M) \ ; \ m_i \mapsto CR \cap p_{i,\{m_i\}}\Big)_{i \in I}$$

defines the consistency-induced observable $o[CR]$.

Proposition 2. *For any V-SUM meta-model* $(\prod M, CR)$, $CR_{o[CR]} = CR$.

Proof. Let $m_I \in \prod M$ be a tuple of models.

$$m_I \in CR_{o[CR]} \iff CR \cap \bigcap_I p_{i,\{m_i\}} \neq \emptyset \iff m_I \in CR \ .$$

A careful reader will have noticed that for any $m_i \in M_i$, $CR^{\nabla i}(m_i)$ is isomorphic to $o[CR]_i(m_i)$. The two functions induce the same equivalence kernel: $m_a \sim_i^{CR} m_b \iff o[CR]_i(m_a) = o[CR]_i(m_b) \iff CR^{\nabla i}(m_a) = CR^{\nabla i}(m_b)$. Prop. 2 shows that any arbitrary consistency relation CR can be encoded by a single observable. However, this consistency-induced observable $o[CR]$ is generally not appropriate to be used in practice (nor is $CR^{\nabla i}$) as the codomain $\mathscr{P}(\prod M)$ reflects full system descriptions and lacks intuitive, domain-relevant interpretation. Instead, it would be preferable to construct CR using observables, each capturing one relevant aspect of the system, motivating methods to combine observables, explored in the next section.

5 Beyond Simple Observables Semantics

Observables enable reasoning about consistency in terms of concrete, i.e., *observable*, values. However, although logically expressive, the framework developed so far – which we refer to as *simple observable semantics* – lacks the means to capture some additional constraints naturally. Consider, for instance, the requirement in Sect. 3.2 in which the chassis model m_{ch} imposes a threshold $w_{\max}$ on the total weight of the EV. Ideally, we want to add the EV's total weight as an observable. However, we currently cannot describe that the total weight value should be the sum of the subsystems' weight values (available from the models m_{geo}, m_{mat} and m_{mot}).

This section extends the notion of observable semantics to support such composite requirements. We proceed in two steps. First, we introduce *meta-model spanning observable semantics*, which allows observable values to depend on information from models defined over distinct meta-models. Second, we define an algebraic construction on observables to create *compound observables*. We also provide an encoding of the associated observable semantics back into the original framework, ensuring that all previous results remain valid.

5.1 Limitations of Tuple-Based Encoding

Example 3. A natural starting point towards describing the total vehicle weight is to define separate observables for the weight contributions of each subsystem, namely the weight of the electrical components, the body volume, and the material density. We denote these by o^{mot}, o^{geo}, and o^{mat}, respectively. Formally, each of these observables has semantics defined by:

$$o_j^s(m_j) = \begin{cases} \mathbb{R}_+ & \text{if } j \neq s \\ \{v_s\} & \text{if } j = s \end{cases} \qquad (1)$$

where $s \in \{\text{geo}, \text{mat}, \text{mot}\}$ and v_s denotes the concrete values extracted from model m_s. Each observable thus yields a singleton set $\{v_s\}$ for its corresponding model but remains unconstrained by the other models.

While this encoding allows reasoning about individual values, simple observables are inherently local and inadequate for aggregated constraints that span multiple models. There is no way to state that the *total weight* must remain below a threshold, as it remains unconstrained in every single model.

Example 4. One might attempt to encode the desired constraint with a single observable o reflecting relevant intermediate quantities (e.g., volume, density, and component weights) and check a constraint on their combination:

$$\begin{aligned}
o_{\text{geo}}(m_{\text{geo}}) &= \{(v_{\text{geo}}, b, c, d) \mid b, c, d \in \mathbb{R}_+\} \\
o_{\text{mat}}(m_{\text{mat}}) &= \{(a, v_{\text{mat}}, c, d) \mid a, c, d \in \mathbb{R}_+\} \\
o_{\text{mot}}(m_{\text{mot}}) &= \{(a, b, v_{\text{mot}}, d) \mid a, b, d \in \mathbb{R}_+\} \\
o_{\text{ch}}(m_{\text{ch}}) &= \{(a, b, c, w_{\max}) \mid a, b, c \in \mathbb{R}_+, a \cdot b + c \leq w_{\max}\}
\end{aligned}$$

The family of abstract semantics of o aggregates values from the four models (note that o_{geo} is the observable semantics o gives to m_{geo}, whereas o^{geo} in Ex. 3 is a wholly separate observable): the geometry model m_{geo} provides the body volume value v_{geo}, the materials model m_{mat} provides the body density value v_{mat}, the motion model m_{mot} provides the weight value of motion-related components v_{mot}, and the chassis model m_{ch} provides the maximum admissible weight value $w_{\max}$. The constraint ensuring that the total weight does not exceed $w_{\max}$ is encoded in the value set of m_{ch}. Note that the tuple-valued observable semantics are obtained for the first three models by lifting the respective observable semantics from Ex. 3 to encompass all components. Formally, $o_s(m_s) = \prod_I o^i_s(m_s)$.

While this approach can represent the desired constraint and may support formal developments (cf. Sect. 5.5), it presents conceptual limitations from a modelling perspective. First, the observable values – tuples of real numbers – diverge from the intuitive notion of observables as directly accessible or measurable quantities: these tuples are not derivable from the system and are not likely to reflect artefacts that engineers would explicitly annotate or reason about. Second, the constraint (i.e., that the combined weight may not exceed a threshold) is embedded within the observable. This embedding blurs the line between modelling and consistency checking. Instead of providing an observable quantity, the observable encodes a decision procedure answering whether a configuration is acceptable. To address these issues, we introduce the notion of *meta-model spanning (MMS) observable semantics*, which constrains the value set from multiple models. We introduce *compound observables* as a constructive way to build such semantics, e.g., building a single observable w_{total} from the three intuitive observables o^{geo}, o^{mat} and o^{mot}.

5.2 Meta-model Spanning Observable Semantics

Since observables are defined via a set of domain values from which we derive the observable semantics, we can generalise observable semantics to collections of meta-models. Given an observable $o \in O$, we define a family of semantics

$$\Big(o_J \colon \prod_{j \in J} M_j \to \mathscr{P}(V_o)\Big)_{J \subseteq I} \quad . \tag{2}$$

Without further conditions, the value of o for M_i might conflict with its value for $\prod_{j \in J} M_j$ where $i \in J$. To prevent such a conflict, we require the following *coherence condition*: for all $J \subseteq I$ and all $K \subseteq J$, the values computed from o_J must be included in those obtained by projecting and evaluating via o_K. Formally, let $\pi_{J,K}$ be the projection $\prod_J M_j \to \prod_K M_k$, then

$$o_J \subseteq \bigcap_{K \subseteq J} o_K \circ \pi_{J,K} \quad . \tag{3}$$

Definition 11 (MMS observable semantics). *An observable $o \in O$ yields a family of meta-model spanning (MMS) observable semantics as in eq. (2) if the family satisfies the coherence condition of eq. (3).*

The coherence condition on MMS observable semantics mimics Def. 9, ensuring that combining models can only restrict observable values. Additional values may only arise when disentangling meta-models.

Observation 1 *Any observable $o \in O$ with a family of observable semantics $(o_i \colon M_i \to \mathscr{P}(V_o))_{i \in I}$, yields a family of MMS observable semantics defined as*

$$o_J \colon \prod_{j \in J} M_j \to \mathscr{P}(V_o) \ ; \ m_J \mapsto \bigcap_{j \in J} o_j(m_j) \quad .$$

Thus we can also treat a family of simple observable semantics for o as if it were a MMS observable, and write o_J for it. Then, the observable-induced consistency from Def. 9 can be refined and defined both for observables semantics and MSS observables semantics as

$$CR_o = \{m_I \in \prod M \mid o_I(m_I) \neq \emptyset\} \quad .$$

Also note that for $i \in I$, $o_{\{i\}} = o_i$, while the notion is ill-defined on the empty set, such that we always consider non-empty subsets of I.

Example 5. We aim to construct MMS observable semantics for w^{total} to describe total vehicle weight. For $s \in \{\text{geo}, \text{mat}, \text{mot}\}$, $w^{\text{total}}_s(m_{\text{s}}) = [v_{\text{s}}, \infty)$. The observable semantics for the collection of these three meta-models is

$$w^{\text{total}}_{\{\text{geo},\text{mat},\text{mot}\}}(m_{\text{geo}}, m_{\text{mat}}, m_{\text{mot}}) = \{v_{\text{mat}} \cdot v_{\text{geo}} + v_{\text{mot}}\} \quad .$$

In conjunction with $w^{\text{total}}_{\text{ch}}(m_{\text{ch}}) = [0, w_{\max}]$, we can deduce that for a tuple of models $m = (m_{\text{geo}}, m_{\text{mat}}, m_{\text{mot}}, m_{\text{ch}})$:

$$w^{\text{total}}_{\{\text{geo},\text{mat},\text{mot},\text{ch}\}}(m) = \begin{cases} \{v_{\text{mat}} \cdot v_{\text{geo}} + v_{\text{bat}}\} & \text{if } v_{\text{mat}} \cdot v_{\text{geo}} + v_{\text{bat}} \leq w_{\max} \\ \emptyset & \text{otherwise} \end{cases}$$

Hence, $CR_{w^{\text{total}}}$ contains the tuples of models such that the total weight is at most $w_{\max}$, which accurately represents the requirement on vehicle weight.

Ex. 5 illustrates how defining observable semantics at various levels of the products of meta-models allows precise encoding of cross-model constraints, such as a system-wide weight threshold. It also shows that we need a solution to construct said observable semantics.

5.3 Compound Observables

We often need observables that are not atomic but derived from others through computations to describe and compare system properties. For this, we introduce *compound observables*, built by composing existing observables with functions from a given algebra. Compound observables provide a structured way to express computations and constraints that involve multiple aspects of the system, such as computing total weight from individual component weights.

Let $\Sigma = (S, F)$ be a multi-sorted signature, where S is a set of sorts and F is a set of function symbols. Each function symbol $\sigma \in F$ has a profile $(s_1, \dots, s_k, s) \in S^+$, allowing to write $\sigma : \prod_{1 \leq i \leq k} s_i \to s$ and say σ has arity k. We fix a Σ-algebra $\mathcal{A}$ that provides a concrete interpretation for these symbols. It consists of a carrier set $[\![s]\!]_{\mathcal{A}}$ for each sort $s \in S$ and an operation $[\![\sigma]\!]_{\mathcal{A}} \colon \prod_{1 \leq i \leq k} [\![s_i]\!]_{\mathcal{A}} \to [\![s]\!]_{\mathcal{A}}$ for each $\sigma : \prod_{1 \leq i \leq k} s_i \to s$ in F.

As an algebra allows interpreting the symbols, we can construct observable semantics via structured application of the operations from the algebra: we apply the operations of $\mathcal{A}$ pointwise on the models. For this, we first have to assume that $\mathcal{A}$ is *suitable* for O, that is, for each observable $o \in O$, there is a sort $s \in S$ such that $V_o = [\![s]\!]_{\mathcal{A}}$. Given observables $(o^k)_{k \in K}$, the pointwise application of a function $\sigma : \prod_K s_k \to s$ is defined as follows:

Definition 12 (Pointwise Lifted Interpretation). *Let $\sigma : \prod_K s_k \to s$ be a function symbol in F and $(o^k)_{k \in K}$ be observables with value sets $([\![s_k]\!]_{\mathcal{A}})_{k \in K}$. Then for each $i \in I$, the pointwise lifted interpretation $\sigma^\bullet[o^K]_i$ of σ on $(o^k)_{k \in K}$ for M_i is defined as*

$$\sigma^\bullet[o^K]_i(m_i) \coloneqq \left\{ [\![\sigma]\!]_{\mathcal{A}}(v_K) \mid v_K \in \prod_K o_i^k(m_i) \right\} .$$

Proof (Well-definedness). Assuming that each observable o^k admits an I-indexed family of observable semantics, then $o_i^k(m_i) \subseteq [\![s_k]\!]_{\mathcal{A}}$, meaning that $[\![\sigma]\!]_{\mathcal{A}}(v_K)$ is a well-defined computation.

Example 6. Let σ be interpreted as a multiplication function in the algebra. Suppose that o^{geo} and o^{mat} provide the vehicle body's volume and material mass values. Then $\sigma^\bullet(o^{\text{geo}}, o^{\text{mat}})$ represents the weight of the vehicle body.

The pointwise interpretation computes one expression per observable tuple. Compound observables generalise this idea by allowing *multiple expressions* to be evaluated from different subsets of observables, such that all evaluations yield consistent results. We can now compute a derived quantity and ensure it satisfies a condition expressed elsewhere.

Definition 13 (Compound Observable). *Let $(o^k)_{k \in K}$ be observables with value sets $(\llbracket s_k \rrbracket_{\mathcal{A}})_{k \in K}$. A finite set $\mathcal{C}$ of index sets $c \subseteq K$ together with a function symbol $\sigma_c : \prod_{\ell \in c} s_\ell \to s$ in F for each $c \in \mathcal{C}$ defines the compound observable $\sigma^\bullet_{\mathcal{C}}[o^K]$ with observable semantics $\sigma^\bullet_{\mathcal{C}}[o^K]_J \colon \prod_{j \in J} M_j \to \mathscr{P}(\llbracket s \rrbracket_{\mathcal{A}})$ given by*

$$\sigma^\bullet_{\mathcal{C}}[o^K]_J(m_J) \coloneqq \bigcap_{c \in \mathcal{C}} \sigma^\bullet_c[o^c]_J(m_J) \ . \tag{4}$$

As simple observables are used in this construction, we use the derived MMS observable semantics from Def. 13. The intersection over $\mathcal{C}$ ensures that all involved computations yield the same result, i.e., only values consistent across all sub-expressions are retained.

Example 7. Consider the observables o^{mot}, o^{geo}, and o^{mat} from Ex. 3 together with o^{ch} providing the total weight limit $w_{\max}$. We aim to derive the total weight and ensure it stays within the allowed maximum $w_{\max}$. For this, we define a compound observable using two function symbols:

– σ_{weight}, with interpretation $\llbracket \sigma_{\mathrm{weight}} \rrbracket_{\mathcal{A}}(\nu, \rho, \omega) = \nu \cdot \rho + \omega$ used to compute the total weight;
– id, with interpretation $\llbracket \mathrm{id} \rrbracket_{\mathcal{A}}(x) = x$, used to propagate the weight limit.

We define a compound observable over index sets $\mathcal{C} = \{\{\mathrm{geo}, \mathrm{mat}, \mathrm{mot}\},\ \{\mathrm{ch}\}\}$ with the corresponding functions $\sigma_{\{\mathrm{geo},\mathrm{mat},\mathrm{mot}\}} = \sigma_{\mathrm{weight}}$ and $\sigma_{\{\mathrm{ch}\}} = \mathrm{id}$. Writing $\mathrm{gmm} = \{\mathrm{geo}, \mathrm{mat}, \mathrm{mot}\}$, the compound observable $\sigma^\bullet_{\mathcal{C}}[o^{\mathrm{geo}}, o^{\mathrm{mat}}, o^{\mathrm{mot}}, o^{\mathrm{ch}}]$ evaluates as follows:

$$\begin{aligned}
\sigma^\bullet_{\mathrm{gmm}}[o^{\mathrm{geo}}, o^{\mathrm{mat}}, o^{\mathrm{mot}}](m) &= \left\{ \nu \cdot \rho + \omega \ \middle|\ \begin{array}{l} \nu \in o^{\mathrm{geo}}_{\mathrm{gmm}}(m_{\mathrm{gmm}}), \\ \rho \in o^{\mathrm{mat}}_{\mathrm{gmm}}(m_{\mathrm{gmm}}), \\ \omega \in o^{\mathrm{mot}}_{\mathrm{gmm}}(m_{\mathrm{gmm}}) \end{array} \right\} &&= \{v_{\mathrm{geo}} \cdot v_{\mathrm{mat}} + v_{\mathrm{mot}}\} \\
\sigma^\bullet_{\{\mathrm{ch}\}}[o^{\mathrm{ch}}](m) &= \{v \mid v \in o^{\mathrm{ch}}_{\mathrm{ch}}(m_{\mathrm{ch}})\} &&= [0, w_{\max}]
\end{aligned}$$

and their intersection yields $\sigma^\bullet_{\mathcal{C}}[o^{\mathrm{geo}}, o^{\mathrm{mat}}, o^{\mathrm{mot}}, o^{\mathrm{ch}}](m) = \{\nu \cdot \rho + \omega\} \cap [0, w_{\max}]$.

This observable returns a singleton $\{w\}$ if and only if the computed total weight does not exceed $w_{\max}$, and $\emptyset$ otherwise. In other words, the models are consistent with the total weight constraint if: $\nu \cdot \rho + \omega \leq w_{\max}$, which provides an observable-level formulation of the semantic requirement.

Compound observables generalize to MMS observable semantics (Def. 11) for $(o^k)_{k \in K}$. Regardless of the input observable semantics, the following holds:

Proposition 3. *The semantics of compound observables are MMS observable semantics.*

Proof. The compound observable $\sigma^\bullet_{\mathcal{C}}[o^K]$ maps to subsets of $\llbracket s \rrbracket_{\mathcal{A}}$, meaning that it has proper typing and only the coherence condition from eq. (3) needs to be checked. The condition follows from permuting intersections as the observables are interpreted using their MMS observable semantics (either directly or in the form of Ob. 1).

5.4 Encoding Requirements in Observable Semantics

A central feature of our observable semantics framework is its flexibility in handling system requirements. Constraints such as “the total weight must not exceed a threshold” can be captured in different ways.

Requirements via Observable Values (Value Filtering) The most direct way to encode requirements is by *restricting the value set* of the observable to those values meeting the requirement’s constraints. This encoding keeps the constraint implicit, resulting in a lack of transparency. Due to the hardcoded monolithic integration, domain-experts engineers cannot easily update the encoding.

Requirement via Compound Observables (Computational Filtering) A requirement can also be *represented as a compound observable*, allowing the constraint to depend on multiple models. This encoding is also implicit but can be made explicit using auxiliary expressions for the observable, as shown in Ex. 7. The compound observable filters outputs based on conditions specified by a model, allowing domain experts to formalise the validation of global properties; however, the requirement remains entangled with the observables.

Requirement as First-Class Models (Model-based Filtering) Finally, requirements can be made *first-class citizens of the modelling space* making them (part of) a *requirements model* within the V-SUM defining the parameters of a condition (e.g., the weight threshold) and the condition itself. The requirement model thus provides the syntactic material needed in the Σ-algebra $\mathcal{A}$, i.e., the function used to combine observable semantics into compound semantics. Within our framework, this would require a slight adaption of the compound observable definition to handle observables whose value set contains function symbols and base the pointwise lifted interpretation on the associated function from $\mathcal{A}$. This approach modularises the constraint specification and supports better traceability, reuse, and validation of requirements at the cost of providing a less narrowed space for domain-specific engineers to work in. Additionally, complex information must be encoded in the requirement model that should then be carefully edited.

5.5 Meta-model Spanning Observables Are Also Simple

Meta-model spanning observables allow our formalisation to be much closer to actual engineering practice than an approach based purely on simple observables. However, what we gain in naturality, we lose in conceptional elegance, and the proofs of many properties become much more involved. Hence, instead of giving a direct proof that any semantics-induced consistency relation (as in [20]) can be induced by compound observables, we show that any compound observable can be equivalently represented as a simple observable at the cost of pushing complexity into its value space. Here, we consider observables equivalent if they induce the same consistency relation.

Definition 14 (Equivalence of observables). *Two observables are equivalent, written $\approx$, if they yield the same consistency relation, i.e. for o, o' in O, $o \approx o'$ if and only if $CR_o = CR_{o'}$.*

By definition, two observables $o \approx o'$ if and only if $o_I(m) \neq \emptyset \iff o'_I(m) \neq \emptyset$ for all models $m \in \prod M$. This property is considerably weaker than equality (or even isomorphism) on the semantic functions – it is enough that they accept the same models as consistent. For compound observables, pushing the complexity into the value space is done by mimicking the construction from Sect. 5.1.

Proposition 4 (Tuple encoding). *Given a compound observable $\sigma^{\bullet}_{\mathcal{C}}[o^K]$ built over observables $(o^k)_{k\in K}$, the tuple encoding*

$$\Big(\mathfrak{T}(\sigma^{\bullet}_{\mathcal{C}}[o^K])_i \colon M_i \to \prod_K \mathscr{P}(V_{o^k})\ ;\ m_i \mapsto \{v_K \in \prod_K o^k_i(m_i) \mid \bigcap_{\mathcal{C}} [\![\sigma_c]\!]_{\mathcal{A}}(v_K) \neq \emptyset\}\Big)_{i\in I}$$

of o is an observable equivalent to $\sigma^{\bullet}_{\mathcal{C}}[o^K]$.

Proof. For $m \in \prod M$, $\mathfrak{T}(\sigma^{\bullet}_{\mathcal{C}}[o^K])_I(m) \neq \emptyset$ iff there exists $v_K \in \prod_K o^k_I(m)$ such that $\bigcap_{\mathcal{C}} [\![\sigma_c]\!]_{\mathcal{A}}(v_K) \neq \emptyset$, i.e., $v_K \in \sigma^{\bullet}_{\mathcal{C}}[o^K]_I(m)$, that is $\sigma^{\bullet}_{\mathcal{C}}[o^K]_I(m) \neq \emptyset$. Thereafter $\sigma^{\bullet}_{\mathcal{C}}[o^K] \approx \mathfrak{T}(\sigma^{\bullet}_{\mathcal{C}}[o^K])$.

Of course, while useful, this result should not be taken to mean that compound observables are unmotivated. The encoding can be mechanically computed from the component observables and σ_C alone. Still, for practical applications, it will usually introduce drawbacks (in particular, for observables with finite value sets, it vastly increases the set's cardinality compared to the equivalent compound). Instead, the encoding provides an alternative view of the same structure, and implementations may choose one or the other depending on their own priorities.

We can even perform a similar construction to also reduce arbitrary MMS observable semantics to simple observable semantics, even without requiring knowledge of them as compounds, i.e., by interpreting them purely in terms of the models they affect:

Proposition 5 (Generalised encoding). *Given a meta-model spanning observable o, the generalised encoding*

$$\Big(\mathfrak{G}(o)_i \colon M_i \to \prod M\ ;\ m_i \mapsto CR_o \cap p_{i,\{m_i\}}\Big)_{i\in I}$$

of o is an observable equivalent to o.

Proof. Let $m \in \prod M$. Then $\mathfrak{G}(o)_I(m) = \bigcap_{i\in I} CR_o \cap p_{i,\{m_i\}} = CR_o \cap \{m\}$. Thus, $\mathfrak{G}(o)_I(m) \neq \emptyset \iff m \in CR_o \iff o_I(m) \neq \emptyset$. Thereafter $o \approx \mathfrak{G}(o)$.

However, this now requires additional knowledge of the modelling domains (in particular, we need to have the entire set of instances of the meta-models $(M_i)_{i\in I}$ available) in order to perform the latter encoding, limiting its applicability to real-world scenarios or implementations, as enumerating all possible model instances may be impractical or impossible.

6 Discussion and Future Work: Consistency Management

The previous sections presented the conceptual and set-theoretic foundations of observables in MDD, but we also want to validate their potential within engineering workflows. We identified the following key application potentials:

Informal discussion: Observables serve as a shared conceptual interface that enables interdisciplinary dialogue about system behaviour and the impact of modelling decisions, facilitating early detection of inconsistencies across modelling artefacts, even before formal analysis.

Formal semantic alignment: Once formalised, observables support precise reasoning about consistency, unambiguously identifying the source of inconsistencies across heterogeneous models.

Implementation-level integration: Observables can be made into first-class implementation artefacts and integrated into consistency checking, monitoring, or repair tools.

We opted for a qualitative validation approach in the form of an interdisciplinary workshop involving computer scientists as well as mechanical and electrical engineers. The workshop introduced observables and explored their role in semantic consistency management in CPS development. The central case study discussed for the workshop involved a model inconsistency in the automotive domain, based on a real-world scenario [1]. The case focused on a high-end hybrid vehicle with unsatisfactory braking behaviour, which was identified shortly before product release. The root cause was an inconsistency introduced in an early design phase arising from a mismatch in the braking phases between the mechanical friction brake and the electrical brake for recuperative braking via the electric motor operating as a generator. The unsatisfactory braking behaviour could not be detected within the driver simulation software. The interdisciplinary exchange yielded insights that strengthened our vision of the potential of observables:

Adequacy: The observable *braking force profile*, a function representing applied braking force over time, was suggested to serve as a common semantic anchor across domains. It (1) raised awareness of potential inconsistencies (informal discussion), (2) provided a precise interdisciplinary discussion basis for domain experts (formal discussion), and (3) enabled the implementation of automated consistency analysis tools.

Heterogeneity: The modelling artefacts for the electrical and mechanical braking systems differed significantly. Friction-based braking models rely on empirical data from test runs, while electrical brakes can be modelled mathematically. Braking force profiles, however, are semantically meaningful in both domains and could provide a common point of reference – illustrating the value of observables in bridging modelling heterogeneity.

Extensibility: The possibility of inconsistency between the braking drivers' software was previously unidentified and not modelled explicitly, highlighting the necessity of maintaining a collection of observables rather than a single universal semantics. Observables not only reveal inconsistencies but also help identify missing requirements, especially when expectations span multiple modelling domains.

6.1 Future Work

Currently, observables induce consistency by requiring all models in a V-SUM to admit a common value. For some modelling contexts, one might require refinement or inclusion (e.g., one value set is a subset of another) rather than requiring agreement on a value. This generalisation may involve enriching the algebraic constructions in Sect. 5.3 with logical connectives or relational operations. Once inconsistencies can be identified and described using observables, an obvious question arises: Can these observables also be used to resolve inconsistencies? We are currently developing a formal framework and implementation for a *consistency repair* mechanism that relies on observables to describe the effect of model transformations. Finally, in collaboration with domain experts, we aim to explore further the roles and potential applications of observables within design workflows, from informal validation to formal tooling. As more concrete modelling artefacts become available, we intend to refine the automotive example into a comprehensive case study.

7 Conclusion

We introduced *observables* as a novel and semantically grounded approach to consistency management between heterogeneous models, especially in cyber-physical systems. By abstracting system properties as measurable quantities, observables provide a unified and intuitive framework for assessing model consistency. We showed that any consistency relation may be represented by an observable. We further presented two principled extensions – *meta-model spanning semantics* and *compound observables* – that enable modular construction of complex observable semantics. We proved that both extensions can be reduced to simple observable semantics, retaining the related results. This design enables separation of concerns and extensibility as new observables can be added incrementally as consistency requirements evolve. Beyond theoretical expressiveness, observables offer practical flexibility: constraints may be expressed externally, derived through observable computation, or embedded in models. As such, observables serve as a bridge between modelling practice and formal reasoning and offer a promising foundation for tool-supported consistency management in engineering workflows.

Acknowledgments. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – CRC 1608 – 501798263.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Albers, A., Völk, T.A., Pfaff, F., Nowak, K.: Inconsistency situations in engineering of cyber-physical systems (Apr 2025). https://doi.org/10.5281/zenodo.15267521
2. Ambler, S.W.: The Object Primer: Agile Model-Driven Development with UML 2.0. Cambridge University Press, Cambridge, 3 edn. (2004). https://doi.org/10.1017/CBO9780511584077
3. Bowman, H., Steen, M., Boiten, E., Derrick, J.: A Formal Framework for Viewpoint Consistency. Formal Methods in System Design **21**(2), 111–166 (Sep 2002). https://doi.org/10.1023/A:1016000201864
4. Calegari, D., Mossakowski, T., Szasz, N.: Model-Driven Engineering in the Heterogeneous Tool Set. In: Braga, C., Martí-Oliet, N. (eds.) Formal Methods: Foundations and Applications. pp. 64–79. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-15075-8_5
5. Chen, K., Sztipanovits, J., Abdelwahed, S.: A Semantic Unit for Timed Automata Based Modeling Languages. In: 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06). pp. 347–360 (Apr 2006). https://doi.org/10.1109/RTAS.2006.8
6. Chen, K., Sztipanovits, J., Abdelwalhed, S., Jackson, E.: Semantic Anchoring with Model Transformations. In: Hartman, A., Kreische, D. (eds.) Model Driven Architecture – Foundations and Applications. pp. 115–129. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11581741_10
7. Chen, K., Sztipanovits, J., Neema, S.: Toward a semantic anchoring infrastructure for domain-specific modeling languages. In: Proceedings of the 5th ACM International Conference on Embedded Software. pp. 35–43. EMSOFT '05, Association for Computing Machinery, New York, NY, USA (Sep 2005). https://doi.org/10.1145/1086228.1086236
8. Chen, K., Sztipanovits, J., Neema, S.: Compositional Specification of Behavioral Semantics. In: Automation & Test in Europe Conference & Exhibition 2007 Design. pp. 1–6 (Apr 2007). https://doi.org/10.1109/DATE.2007.364408
9. Dibowski, H., Massa Gray, F.: Applying Knowledge Graphs as Integrated Semantic Information Model for the Computerized Engineering of Building Automation Systems. In: Harth, A., Kirrane, S., Ngonga Ngomo, A.C., Paulheim, H., Rula, A., Gentile, A.L., Haase, P., Cochez, M. (eds.) The Semantic Web. pp. 616–631. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_36
10. Fradet, P., Le Métayer, D., Périn, M.: Consistency checking for multiple view software architectures. SIGSOFT Softw. Eng. Notes **24**(6), 410–428 (Oct 1999). https://doi.org/10.1145/318774.319258, https://dl.acm.org/doi/10.1145/318774.319258
11. Grönniger, H., Ringert, J.O., Rumpe, B.: System Model-Based Definition of Modeling Language Semantics. In: Lee, D., Lopes, A., Poetzsch-Heffter, A. (eds.) Formal Techniques for Distributed Systems. pp. 152–166. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02138-1_10
12. Gurevich, Y.: Evolving algebras 1993: Lipari guide. In: Specification and Validation Methods, pp. 9–36. Oxford University Press, Inc., USA (Sep 1995)
13. Hailpern, B., Tarr, P.: Model-driven development: The good, the bad, and the ugly. IBM Systems Journal **45**(3), 451–461 (2006). https://doi.org/10.1147/sj.453.0451

14. Hennicker, R., Bidoit, M.: Observational Logic. In: Haeberer, A.M. (ed.) Algebraic Methodology and Software Technology. pp. 263–277. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-49253-4_20
15. Kamburjan, E., Fiorini, S.R.: On the Notion of Naturalness in Formal Modeling. In: Ahrendt, W., Beckert, B., Bubel, R., Johnsen, E.B. (eds.) The Logic of Software. A Tasting Menu of Formal Methods: Essays Dedicated to Reiner Hähnle on the Occasion of His 60th Birthday, pp. 264–289. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-08166-8_13, https://doi.org/10.1007/978-3-031-08166-8_13
16. Klare, H., Kramer, M.E., Langhammer, M., Werle, D., Burger, E., Reussner, R.: Enabling consistency in view-based system development — The Vitruvius approach. Journal of Systems and Software **171**, 110815 (Jan 2021). https://doi.org/10.1016/j.jss.2020.110815
17. Knapp, A., Mossakowski, T.: Multi-view Consistency in UML: A Survey. In: Heckel, R., Taentzer, G. (eds.) Graph Transformation, Specifications, and Nets: In Memory of Hartmut Ehrig, pp. 37–60. Lecture Notes in Computer Science, Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-75396-6_3
18. Lee, E.A.: CPS foundations. In: Proceedings of the 47th Design Automation Conference. pp. 737–742. DAC '10, Association for Computing Machinery, New York, NY, USA (Jun 2010). https://doi.org/10.1145/1837274.1837462
19. Lucas, F.J., Molina, F., Toval, A.: A systematic review of UML model consistency management. Information and Software Technology **51**(12), 1631–1645 (Dec 2009). https://doi.org/10.1016/j.infsof.2009.04.009, https://www.sciencedirect.com/science/article/pii/S0950584909000433
20. Pascual, R., Beckert, B., Ulbrich, M., Kirsten, M., Pfeifer, W.: Formal foundations of consistency in model-driven development. In: 12th international symposium on leveraging applications of formal methods, verification and validation (ISoLA 2024). Lecture notes in computer science (Oct 2024)
21. Perin, M., Wouters, L.: Using Ontologies for Solving Cross-Domain Collaboration Issues. IFAC Proceedings Volumes **47**(3), 7837–7842 (Jan 2014). https://doi.org/10.3182/20140824-6-ZA-1003.01575
22. Rumpe, B.: Formale Methodik Des Entwurfs Verteilter Objektorientierter Systeme. Informatik, Utz, Wiss. Verlag (1996)
23. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theoretical Computer Science **249**(1), 3–80 (Oct 2000). https://doi.org/10.1016/S0304-3975(00)00056-6
24. Spanoudakis, G., Zisman, A.: Inconsistency management in software engineering: survey and open research issues. In: Handbook of Software Engineering and Knowledge Engineering, pp. 329–380. World Scientific Publishing Company (Dec 2001). https://doi.org/10.1142/9789812389718_0015, https://www.worldscientific.com/doi/abs/10.1142/9789812389718_0015
25. Stachowiak, H.: Allgemeine Modelltheorie. Springer (1973)
26. Stünkel, P., König, H., Lamo, Y., Rutle, A.: Comprehensive Systems: A formal foundation for Multi-Model Consistency Management. Formal Aspects of Computing **33**(6), 1067–1114 (Dec 2021). https://doi.org/10.1007/s00165-021-00555-2