Towards a Taxonomy of Inconsistency Patterns in Multi-View Modelling

Vadim Zaytsev[1]

[1]Formal Methods & Tools, EEMCS, University of Twente, The Netherlands

Abstract

Multi-view modelling relies on consistency across heterogeneous views, yet the literature still lacks a compact, example-backed taxonomy of the inconsistency patterns that we keep seeing across practice and research prototypes. Existing surveys classify approaches and tools, but they do not stabilise the recurring defect patterns themselves in a form that is directly useful for benchmarking, evaluation, and tool-scope claims. This paper presents a literature-based evidence map and a seven-category taxonomy of inconsistency patterns in multi-view modelling. A seed corpus of 31 papers was built from foundational and survey literature. Then, from 19 sources, 46 inconsistency examples were extracted and coded. After a stabilisation pass using explicit tie-break rules, 40 examples were retained as core evidence and 6 as support-only examples. The resulting taxonomy covers: structural/allocation mismatch, interface/data contract mismatch, behavioural contradiction, requirements satisfaction/coverage gap, naming/terminology divergence, traceability disruption, and temporal/version skew. Primary and secondary codes are distinguished, and temporal skew is treated as an overlay unless drift is itself the main defect. The taxonomy provides a concise vocabulary for describing consistency problems, a reusable evidence map for future ICMM work, a basis for more precise claims about what checking and repair approaches do and do not cover, and in general is aimed to serve as a start of the discussion.

Keywords

multi-view modelling, inconsistency management, taxonomy, consistency checking, traceability, model evolution

1. Introduction

Complex software and systems can rarely be described by a single model. Instead, multiple views capture complementary concerns such as structure, behaviour, interfaces, requirements, deployment, testing, and implementation. This makes consistency management central to multi-view modelling: once a system is decomposed across views, engineering quality depends not only on the correctness of each individual view, but also on the validity of the relations between them [1, 2]. Existing reviews confirm the importance of the problem, but also show that the field remains fragmented in terminology, uneven in semantic support, and biased toward a subset of inconsistency types [2, 3, 4].

This situation creates a practical problem for both researchers and tool builders. A consistency-checking or repair approach may claim to handle "inconsistencies", while in fact targeting only one narrow family, such as stale trace links, interface mismatches, or structural desynchronisation. Conversely, some patterns that matter strongly in practice, such as behavioural contradictions or requirement coverage failures, often remain under-specified or are treated as secondary effects rather than first-class inconsistency categories [3, 4]. Without a stable taxonomy of recurring patterns, it is difficult to compare approaches, design reusable benchmarks, or argue convincingly that a technique covers the consistency challenges faced in multi-view settings.

This paper addresses that gap with an explicitly modest goal: not a new checking algorithm, and not another full systematic literature review, but a *literature-based evidence map* that stabilises the recurring inconsistency patterns reported in multi-view modelling papers so far. Our starting point was a seed corpus built from surveys, foundational papers, and targeted primary studies. From that

Joint Proceedings of the STAF 2026 Workshops: AgileMDE, GCM, ICMM, LLM4SE, TTC. Rennes, France, June 29-July 3, 2026
vadim@grammarware.net (V. Zaytsev)
https://grammarware.net (V. Zaytsev)
0000-0001-7764-4224 (V. Zaytsev)
© 2026 Copyright © 2026 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-26

CEUR-WS.org/Vol-4338/ICMM-paper1.pdf

corpus, concrete inconsistency examples were extracted, quoting their original wording, and iteratively coded into a compact taxonomy.

The paper makes four contributions:

- a curated evidence map of **46** extracted inconsistency examples from **19** sources, including **40** core examples used for category counts;
- a stabilised seven-category taxonomy of recurring inconsistency patterns in multi-view modelling;
- explicit coding rules for borderline cases, using primary and secondary codes, temporal overlays, and evidence-status filtering; and
- a discussion of implications for evaluation, benchmark construction, and tool-scope claims in ICMM-style research.

The complementary website **https://circus.github.io/tip** visualises the dataset, the evidence map and the taxonomy in a publicly browsable form.

The paper is structured around three research questions: **(1)** What recurring inconsistency patterns appear across published multi-view modelling case material? **(2)** How can these patterns be organised into a compact taxonomy spanning structural, behavioural, naming, traceability, requirement, and temporal dimensions? **(3)** What emphases and blind spots are visible in this curated evidence map, and what do they suggest about the kinds of inconsistencies currently foregrounded in the literature?

2. Positioning and Study Design

2.1. Related Work

This paper is motivated by, but distinct from, several established lines of work. First, systematic reviews have already mapped the broad landscape of multi-view modelling and cross-domain consistency management. Cicchetti et al. have already reported fragmented terminology, limited industrial evidence, and weak support for semantic consistency in multi-view approaches [2]. Torres et al. have classified cross-domain consistency checking tools and identify families such as behavioural, interface, refinement, and requirement inconsistencies, while also noting significant coverage gaps [3]. Jongeling et al. have documented the industrial reality of consistency management in continuous model-based development and show that practitioners need more automation, especially under frequent change [4].

Second, formal and conceptual work has clarified the nature of multi-view consistency without consolidating the recurring defect patterns reported in case material. Reineke et al. have framed consistency as a central problem of multi-view modelling [1], while surveys such as Knapp and Mossakowski's on UML consistency and Stevens' work on networks of models, have helped to place the problem in broader MDE settings [5, 6]. These contributions are essential background, but they have not yet provided a compact, example-backed taxonomy that can be applied across papers and domains, which is the next step that would be nice to take.

Closest in spirit is the formalised classification schema of Kühn et al. [7]. Their work gives a precise model-element-based account of consistency relations and classifies such relations along dimensions including abstraction, metalevel, position, observed property, development phase, quantification, and gradual consistency. Our taxonomy addresses a different unit of analysis. Rather than classifying the consistency relation itself, this paper focuses on recurring ways in which such relations are reported to fail in multi-view modelling case material. In that sense, their schema and the one from this paper are complementary: theirs describes the shape of a consistency relation; this one describes the defect pattern observed when the relation is broken.

The intended outcome of this project is pragmatic. This paper provides a classification artefact and a reusable evidence map. At some point this should evolve into a taxonomy that authors can use to state which inconsistency families they cover, reviewers can use to assess those claims, and future benchmark builders can use to design challenge cases with better category coverage.

2.2. Corpus Construction

The seed corpus was constructed by purposive and snowball sampling, rather than by an exhaustive systematic literature review protocol. This choice follows the goal of the paper: to build and stabilise a taxonomy of recurring inconsistency patterns, not to estimate population frequencies in the literature. The starting point was a review and foundation spine consisting of surveys and conceptual papers on multi-view modelling, UML consistency, cross-domain consistency checking, behavioural consistency, consistency in networks of models, and industrial consistency management. These papers led to the vocabulary, known gaps, and initial search directions.

From this spine, candidate primary papers were added through backwards and forwards snowballing, complemented by targeted searches for terms such as multi-view consistency, viewpoint inconsistency, UML consistency checking, traceability maintenance, behavioural consistency, model-code consistency, model repair, multi-model evolution, and industrial model-based consistency management. A paper was retained in the seed corpus when it satisfied two conditions:

- it addressed consistency relations across views, models, artefacts, phases, or engineering tools; and
- it appeared likely, from its abstract, examples, case descriptions, or survey treatment, to contain extractable inconsistency patterns rather than only a generic statement that consistency is important.

The resulting 31 papers can be group into four categories by the roles they played: eight survey or foundation papers used to frame the field; ten classic primary or example-rich papers on viewpoints, UML, multiple-view development environments, and traceability; eight papers on evolution, repair, collaboration, preservation, and behavioural consistency; and five industrial or recent MBSE papers used to add realistic system-model, code, and cross-tool cases. The full seed list, including the role assigned to each paper, is included in the accompanying data artefact.

The corpus was then expanded and balanced in three targeted passes, focusing successively on behavioural and traceability-heavy sources, and then on requirement/coverage and refinement-heavy sources. This balancing step was necessary because an initial pass naturally overemphasised structural and model-to-code cases.

Selection followed a simple protocol aimed at taxonomy construction rather than exhaustive retrieval. Papers were included when they (i) addressed multi-view, multi-model, or cross-artefact consistency relations, and (ii) contained either concrete inconsistency examples or sufficiently detailed case material from which a recurring inconsistency pattern could be extracted. This led to preferring papers with explicit examples, industrial cases, or worked inconsistency scenarios, and to excluding papers that only mentioned consistency at a high level without enough detail to code a specific pattern.

The final extraction matrix contains 46 rows. The unit of analysis is not a paper, but *an inconsistency example*. A single paper may therefore contribute several rows when it reports multiple distinct patterns. After the stabilisation pass described below, 40 rows were retained as *core* evidence for category counts, and 6 as *support-only* rows useful for interpretation but excluded from quantitative summaries. Support-only rows were typically didactic, constructive, or defect-indirect rather than direct case evidence. Once additional papers mainly yielded examples already covered by the existing categories or only weak support-only variants, and a final recoding pass no longer changed the top-level category set, the expansion was stopped.

2.3. Extraction Schema and Coding Procedure

Each row in the matrix records the following fields: paper identifier, citation, domain, views involved, artefacts involved, original wording, example summary, cause/mechanism, observed consequence, candidate taxon, optional secondary taxon, confidence, and notes. Retaining the authors' original wording was deliberate because terminology itself is part of the problem: different papers often describe related issues using different labels.

Coding proceeded in two passes. In the first pass, descriptive coding was performed: what views were involved, what relation was violated, and whether the reported defect was static or evolution-induced. In the second pass, taxonomic codes were assigned and then stabilised using explicit tie-break rules.

The main stabilisation rules were as follows:

1. **Primary-code rule.** The primary category follows something identified as the *main violated inter-view relation*, which is not necessarily the first symptom reported by the authors.
2. **Secondary-code rule.** A second category is retained only when it captures a substantial additional mechanism rather than a trivial consequence.
3. **Temporal-overlay rule.** Temporal/version skew is primary only when drift or propagation lag is itself the defect; otherwise it is recorded as an overlay on another category.
4. **Evidence-status rule.** Ambiguity due to weak evidence is treated separately from taxonomic ambiguity. Generic or constructive examples may remain in the dataset as support-only rows without forcing category proliferation.

These rules proved especially useful for three recurring overlaps: structural vs refinement/allocation mismatch, requirement coverage vs traceability rupture, and any category vs temporal skew. A key design choice was *not* to introduce a top-level "cross-cutting" category. In practice, such a category would have become a catch-all for examples that were better handled by primary/secondary coding.

3. Taxonomy of Inconsistency Patterns

Table 1 summarises the stabilised taxonomy. During stabilisation, our initial distinction between structural correspondence mismatch and refinement/allocation mismatch consistently collapsed into the same underlying violated relation: an expected correspondence between higher-level and lower-level artefacts was missing, incomplete, or incompatible. These were therefore merged under **C1**.

The categories in Table 1 should not be read as alternatives to formal relation classifiers such as Kühn et al.'s schema [7]. A single example can, in principle, be described both ways. For instance, a traceability disruption may be vertical, inter-model, inter-phase, qualitative, and strict as a consistency relation, while still being classified as **C6** in our defect-pattern taxonomy. Conversely, a behavioural contradiction is not merely a relation that observes behavioural properties; it is a particular failure mode in which the behaviours admitted by related views cannot jointly hold. This separation between relation properties and failure patterns is what allows the taxonomy to support tool-scope and benchmark-coverage claims.

3.1. C1: Structural Mismatch

C1 captures the broad family of cases where views are expected to align through some *correspondence or refinement relation* but do not. Examples include a design-level call relation that is added without a matching implementation update, a missing subsystem in a lower-level model, and failure of a one-to-one model-to-code decomposition mapping [8, 9, 10]. The same family also subsumes many formal refinement cases where a lower-level model changes the realised structure in a way that no longer preserves the intent of the higher-level one [11].

This category became the largest in our corpus. This is to be interpreted cautiously: it likely reflects both genuine prevalence and a publication bias toward mismatches that are visible and comparatively easy to formalise.

3.2. C2: Interface Contract Mismatch

C2 is about *interfacing between components*, it isolates boundary-level disagreements. Typical examples include incompatible call signatures, wrong connector direction, inconsistent parametrisation

Table 1
Stabilised taxonomy of inconsistency patterns. Counts are based on the 40 core examples.

Code	Label	Count	Definition
C1	Structural mismatch	13	An expected correspondence, allocation, or refinement relation between views is missing, extra, or incompatible.
C2	Interface contract mismatch	4	Views disagree at a boundary on signatures, ports, parameter sets, types, units, directions, or equivalent exchanged values.
C3	Behavioural contradiction	3	Views admit conflicting protocols, orderings, pre/postconditions, state combinations, or jointly unsafe behaviour.
C4	Requirement satisfaction gap	7	A requirement is not adequately realised, linked, tested, or accompanied by the artefacts needed to justify satisfaction.
C5	Terminology divergence	3	Corresponding concepts are named differently, or the same label is used for non-equivalent concepts across views.
C6	Traceability disruption	7	Explicit cross-artefact links are missing, stale, ambiguous, incomplete, or insufficiently maintained for navigation or impact analysis.
C7	Temporal skew	3	Views are individually plausible but inconsistent because they reflect different points in evolution, propagation, or branching history.

of the same physical quantity, or mismatched equivalent values in aligned models [8, 9, 12]. **C2** was separated from **C1** because many inter-view defects arise *not* from missing correspondence, but from correspondence that exists yet disagrees on the contract at the boundary.

3.3. C3: Behavioural Contradiction

C3 covers cases where the problem lies in the *jointly admitted behaviour* rather than in naming or structure. In our corpus, the clearest examples come from behavioural multi-modelling: different models compose into globally unsafe traffic-light states, or one model allows progress that is inconsistent with the state assumed by another [13]. There were also cases where refinement examples had lower-level protocols altering the nondeterministic or allowed behaviour of a higher-level one [11].

Although **C3** is one of the smallest categories in our counts, this should not be interpreted as lack of importance. Quite the opposite: behavioural contradictions are repeatedly described by others as difficult to detect, expensive to formalise, and under-supported by available tools [3]. In general, they are the focus of research on feature interaction, typically done with formal methods, such as (bi)simulation of a formal specification and a system execution trace [14].

3.4. C4: Requirement Satisfaction Gap

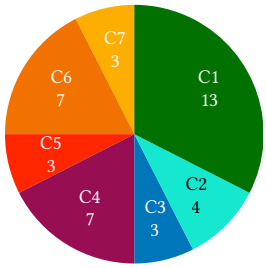
C4 covers failures to justify that *requirements are satisfied or covered.* Some examples are classical viewpoint gaps, such as a required dependent view not being created, or a formal schema lacking its required associated description [15]. Others arise later in the lifecycle, for instance when critical requirements are not traced through to tests, or when tests continue to execute but no longer cover the intended behaviour after model evolution [4, 16].

C4 was separated from traceability (**C6**) because a missing trace is not always the main problem. If the central defect is that satisfaction or coverage cannot be justified, then traceability is often the mechanism rather than the category.

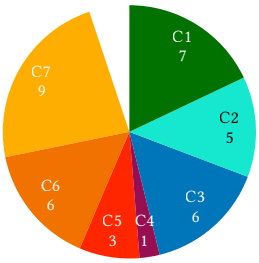
3.5. C5: Terminology Divergence

C5 captures lexical and *terminological disagreement*, be it simply naming or representative of semantic mismatches. Illustrative cases include class renaming in a library that is not propagated across related

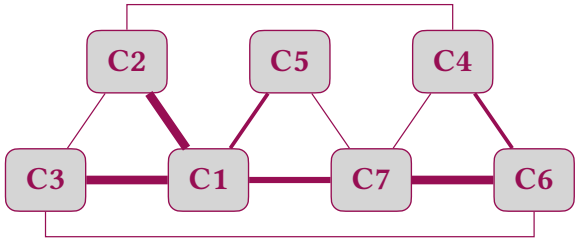
Figure 1: Visualisations of the taxonomy.



(a) Distribution of core examples across the final taxonomy, primary category only



(b) Same, but secondary category only



(c) Co-occurrence of categories in our dataset. For the sake of this visualisation, primary–secondary and secondary–primary co-occurrences are merged, and two one-time co-occurrences (**C1**–**C6** and **C4**–**C1**) are omitted. Thickness of associations encode the number of co-occurrences from 2 to 5.

artefacts, inconsistent names for linked properties, and fragile model-code correspondence rules that depend on naming conventions being followed uniformly [8, 10, 12]. Naming was retained as a top-level category because the surveys already indicate that terminology inconsistency is a real obstacle in multi-view work [2], and because naming problems often act as the first point of failure for correspondence mechanisms.

3.6. C6: Traceability Disruption

C6 covers explicit *correspondence relation failures*: stale links, incomplete link sets, missing links needed for impact analysis, or insufficient relation information for backward navigation after a test failure [17, 18, 19, 20, 21]. In this category, the main defect is that the cross-artefact relation itself is not reliably available, even when the underlying artefacts may still be individually valid.

Disruptions of the traceability chain also turned out to be a major meeting point between research prototypes and industrial practice. Published tools often handle trace links more readily than behavioural semantics, but the literature simultaneously shows how easily those links decay under evolution [17, 18].

3.7. C7: Temporal Skew

C7 captures drift caused by *asynchronous evolution*. Examples include code customised after generation while the source model remains unchanged, models staying stale for long periods while implementations evolve, and parallel versions producing conflicting states for the same method [4, 8, 10].

Temporal skew deserves its own category, but only under a restricted rule: when time-lag or version drift *is* the inconsistency. In many other rows, temporal change is better treated as an overlay on structural mismatch, trace decay, or requirement coverage loss.

4. Evidence Map and Observations

Figure 1(a) and (b) summarise the distribution of the 40 core examples in our curated evidence map. Table 2 lists all the cases by their ID (not important, but included for traceability and alignment with the data artefact), each linked to the paper where it was found, with their primary and secondary (if any) assigned categories. Figure 1(c) visualises co-occurrence of categories as primary and secondary tags. Three observations are visible in this corpus.

- **Correspondence-related problems dominate the published examples.** **C1** is the largest category, and **C2** adds another boundary-focused slice. This confirms the intuition that the literature is especially rich in examples where a correspondence relation can be stated concretely

Table 2
All cases with their IDs, primary and possibly secondary categories.

HLI19-1 [22]	C1	—	DHI19-2 [12]	C2	C1	DHI19-1 [12]	C5	—
RCM05-1 [11]	C1	C2	DHI19-4 [12]	C2	C1	DSS22-3 [10]	C5	C1
RCM05-5 [11]	C1	C2	IMM98-3 [8]	C2	C1	IMM98-1 [8]	C5	C7
RCM05-2 [11]	C1	C3	CCM19-2 [9]	C2	C3	ATD11-1 [23]	C6	—
RCM05-3 [11]	C1	C3	BCM23-1 [13]	C3	C1	BCM23-3 [13]	C6	C3
RCM05-4 [11]	C1	C3	BCM23-2 [13]	C3	C2	MBT12-1 [21]	C6	C4
RCM05-6 [11]	C1	C3	BCM23-4 [13]	C3	C6	ITC16-1 [20]	C6	C7
DHI19-3 [12]	C1	C5	FER94-1 [15]	C4	C1	OAT18-1 [18]	C6	C7
FER94-2 [15]	C1	C5	DHI19-5 [12]	C4	C2	ATM12-1 [17]	C6	C7
DSS22-1 [10]	C1	C6	FER94-3 [15]	C4	C2	CAE08-1 [19]	C6	C7
IMM98-2 [8]	C1	C7	ABB09-1 [16]	C4	C6	ARC22-1 [4]	C7	C1
CCM19-1 [9]	C1	C7	FER94-4 [15]	C4	C6	IMM98-4 [8]	C7	C5
ARC22-3 [4]	C1	C7	MME21-1 [24]	C4	C6	DSS22-2 [10]	C7	C6
			ARC22-2 [4]	C4	C7			

and checked against explicit artefacts [8, 10, 12]. It also helps explain why many consistency tools emphasise structural and interface-level checks.

- **Assurance-oriented inconsistencies are substantial and should not be collapsed into traceability alone.** C4 and C6 each account for seven core examples. This is methodologically important: papers often move quickly from "missing requirement coverage" (C4) to "missing trace" (C6), but our coding shows that they are not exactly the same phenomenon. In one case the central question is whether satisfaction or coverage can be justified; in the other it is whether the relations needed for navigation and impact analysis are maintained [16, 17, 21].

- **Low counts for behaviour, naming, and temporal skew should not be read as low importance.** Behavioural contradiction (C3), naming divergence (C5), and temporal/version skew (C7) are all smaller in the final counts. Yet surveys and industrial studies suggest that these areas are precisely where tooling and practice struggle most [2, 3, 4]. The low counts are to be interpreted at least partly as a *reporting and extraction effect*: these categories are less often made explicit in papers, harder to state in a compact rule, or treated as context rather than as the named problem.

4.1. Cross-Cutting Patterns

A recurring question during stabilisation was whether to add a top-level cross-cutting category as a catch-all for all ambiguities and bridging projects. It was decided against that move for two reasons. First, most ambiguous rows were *ambiguous in a structured way*. As we can see from Figure 1(c), the main overlaps were C1 vs C2, C4 vs C6, and any of those vs C7. Such cases are ideally handled by formulating precise conditions to break ties, or, when impossible, adding optional secondary codes, instead of moving the entire exhibit into a cross-cutting bucket. Second, at least some borderline rows were not taxonomically exotic at all; they were simply weaker evidence, such as constructive papers motivating traceability or didactic examples used to explain a concept. Adding new categories for such rows would have mixed evidence quality with conceptual structure.

In other words, *cross-cuttingness is a coding property, not necessarily a category.* For our dataset, having a primary code, an optional secondary code, and a temporal overlay seemed to have proven sufficient.

5. Implications for Inconsistency Research in Modelling

The proposed taxonomy can be seen as immediately useful in several ways:

- **More precise evaluation claims.** Consistency papers should state which categories they handle. A tool that checks aligned port types and parameter names should not claim broad support for inconsistency management without clarifying that its scope is primarily **C2** (interface contract mismatch), perhaps with some **C1** (structural mismatch) support. Likewise, a repair approach centred on stale links should not silently imply coverage of requirement satisfaction or behavioural contradiction. Ideally, such claims should report both the failure pattern addressed here and the relation type addressed by formal schemas such as Kühn et al.'s [7]: e.g., not only "traceability disruption", but traceability disruption for vertical, inter-phase, inter-model relations.
- **Better challenge and benchmark design.** Reusable case studies in inconsistency management in modelling may risk over-representing the easiest-to-check classes. Our evidence map suggests that future benchmarks should deliberately include at least one representative case for each category, and at least some cross-cutting cases where temporal overlay interacts with another primary defect. This is particularly important for **C3** (behavioural contradiction) and **C4** (requirement satisfaction gap), which are practically important but less frequently made explicit in current benchmark-style papers.
- **Cleaner distinction between symptoms and causes.** The primary/secondary coding scheme helps separate *what is wrong* from *how it became wrong*. For instance, a stale trace caused by evolution is naturally **C6** (traceability disruption) with temporal overlay of **C7**, while an uncovered requirement discovered through missing traces is naturally **C4** (requirement satisfaction gap) with **C6** (traceability disruption) as secondary. This distinction matters for repair: the right remediation for a structural mismatch is not necessarily the same as for a requirement coverage gap, even if both are detected through missing relations.

6. Threats to Validity

The most important limitation is scope. This study is *not* a full systematic literature review. It was intentional to build a curated, balanced seed corpus aimed at taxonomy construction rather than exhaustive coverage. That choice improves feasibility and category exploration, but it also means the final distribution should be read as evidence-guided rather than population-representative. Since the seed corpus was built by purposive and snowball sampling, it should be read as a coverage-oriented corpus for taxonomy construction, not as a reproducible estimate of how often different inconsistency patterns occur in the literature.

A second threat concerns publication and reporting bias. Structural and interface inconsistencies are often easier to articulate in papers than behavioural contradictions or long-term temporal skew. Our counts may therefore under-represent categories that are common in practice but less frequently published as crisp examples.

A third threat lies in coding granularity. The unit of analysis is an example, not a paper, and multiple examples may come from the same source. This is appropriate for taxonomy building, but it means the counts are not independent observations in a statistical sense.

A fourth threat is coder subjectivity. The extraction and stabilisation decisions reported here were not validated through a formal inter-rater agreement study, so some borderline assignments could reasonably be debated. This was mitigated by preserving original wording, recording borderline cases explicitly, and applying the same tie-break rules across the full matrix during a final review pass.

These threats were mitigated in four ways: by seeding from established surveys and foundational papers [2, 3, 4]; by targeted balancing of under-represented areas; by preserving original wording during extraction; and by using explicit stabilisation rules, support-only rows, and primary/secondary coding to avoid category inflation.

Declaration on Generative AI

In preparation of this work, the authors used ChatGPT 5.4 Thinking to support literature triage,

extraction-table structuring, and taxonomy wording alternatives. After the usage, the authors reviewed, edited and adjusted all the content as needed, and take full responsibility for the publication's content.

7. Conclusion

This paper presents a literature-based evidence map and a stabilised taxonomy of inconsistency patterns in multi-view modelling. Starting from a seed corpus of **31** papers, **46** inconsistency examples were extracted and **40** retained as core evidence after stabilisation, disregarding six candidates from four sources [21, 24, 25, 26]. The resulting taxonomy has **seven** categories: structural mismatch, interface contract mismatch, behavioural contradiction, requirement satisfaction gap, terminology divergence, traceability disruption, and temporal skew. A supporting website allows anyone to browse through the taxonomy, the evidence map that supports it, or download the dataset: **https://circus.github.io/tip**.

The main methodological result is not only the final category set, but also the way it was stabilised: by separating category ambiguity from evidence-quality ambiguity, avoiding a catch-all cross-cutting class, and handling overlap through primary/secondary coding plus temporal overlays. The main substantive result is that the examples in our corpus cluster strongly around correspondence, requirement/traceability, and evolution problems, while behavioural and naming issues appear comparatively under-reported despite their acknowledged importance.

For consistency preservation research, the implication is straightforward: instead of claiming generic support for "consistency management", papers should instead state explicitly which inconsistency categories they address, which they leave outside scope, and whether temporal skew is treated as a primary defect or only as context. Likewise, reusable challenge problems and benchmarks should span the full category set rather than concentrating on structural and interface cases alone, no matter how attractive they are as a focus point. On that basis, this taxonomy is proposed as a concrete coordination device for future work: a shared vocabulary for tool-scope claims, a checklist for benchmark coverage, and a way to make cross-paper comparisons more precise. The next steps can be to expand the evidence map, turn the categories into reusable benchmark cases, and test whether existing approaches still justify their scope claims when evaluated against the broader spread of identified inconsistency patterns. A natural extension is to enrich the evidence map with orthogonal relation-level descriptors, for example those of Kühn et al. [7]: their dimensions can describe the formal shape of the consistency relation, while the taxonomy proposed here describes the recurring pattern by which that relation fails.

References

[1] J. Reineke, C. Stergiou, S. Tripakis, Basic Problems in Multi-View Modeling, Software and Systems Modeling (2019). doi:10.1007/s10270-017-0638-1.

[2] A. Cicchetti, F. Ciccozzi, A. Pierantonio, Multi-View Approaches for Software and System Modelling: A Systematic Literature Review, SoSyM (2019). doi:10.1007/s10270-018-00713-w.

[3] W. Torres, M. G. J. van den Brand, A. Serebrenik, A Systematic Literature Review of Cross-Domain Model Consistency Checking by Model Management Tools, Software and Systems Modeling (2021). doi:10.1007/s10270-020-00834-1.

[4] R. Jongeling, F. Ciccozzi, J. Carlson, A. Cicchetti, Consistency Management in Industrial Continuous Model-Based Development Settings: A Reality Check, Software and Systems Modeling (2022). doi:10.1007/s10270-022-01000-5.

[5] A. Knapp, T. Mossakowski, Multi-View Consistency in UML: A Survey, in: Models, Modules and Transformation Systems, 2018. doi:10.1007/978-3-319-75396-6_3.

[6] P. Stevens, Maintaining Consistency in Networks of Models, Software and Systems Modeling (2020). doi:10.1007/s10270-019-00736-x.

[7] T. Kühn, D. Fuchß, S. Corallo, L. König, E. Burger, J. Keim, M. Mazkatli, T. Sağlam, F. Reiche, A. Koziolek, R. Reussner, A Formalized Classification Schema for Model Consistency, Technical Report 1, Karlsruher Institut für Technologie (KIT), 2023. doi:10.5445/IR/1000161127.

[8] J. Grundy, J. Hosking, W. B. Mugridge, Inconsistency Management for Multiple-View Software Development Environments, IEEE TSE (1998). doi:10.1109/32.730545.

[9] R. Jongeling, Work in Progress towards Consistency Management for Industrial Model-based Development, in: A. Etien (Ed.), SATToSE, CEUR Workshop Proceedings, 2019. URL: https://ceur-ws.org/Vol-2510/sattose2019_paper_9.pdf.

[10] R. Jongeling, J. Fredriksson, J. Carlson, F. Ciccozzi, A. Cicchetti, Structural Consistency between a System Model and Its Implementation: A Design Science Study in Industry, Journal of Object Technology 21 (2022) 3:1–16. doi:10.5381/jot.2022.21.3.a6.

[11] H. Wehrheim, Refinement and Consistency in Multiview Models, in: Language Engineering for Model-Driven Software Development, volume 4101 of *Dagstuhl Seminar Proceedings*, 2005, pp. 1–11. doi:10.4230/DagSemProc.04101.13.

[12] S. Feldmann, Diagnosis and Handling of Inconsistencies in Heterogeneous Models of Automated Production Systems, Ph.D. thesis, TUM, 2019. URL: https://d-nb.info/119840244X/34.

[13] T. Kräuter, H. König, A. Rutle, Y. Lamo, P. Stünkel, Behavioral Consistency in Multi-Modeling, Journal of Object Technology 22 (2023) 2:1–15. doi:10.5381/jot.2023.22.2.a9.

[14] P. Shaker, A Feature-Oriented Modelling Language and a Feature-Interaction Taxonomy for Product-Line Requirements, Ph.D. thesis, University of Waterloo, 2013. URL: http://hdl.handle.net/10012/8100.

[15] B. Nuseibeh, J. Kramer, A. Finkelstein, A Framework for Expressing the Relationships between Multiple Views in Requirements Specification, IEEE Transactions on Software Engineering 20 (1994) 760–773. doi:10.1109/32.328995.

[16] F. Abbors, D. Truscan, J. Lilius, Tracing Requirements in a Model-Based Testing Approach, in: First International Conference on Advances in System Testing and Validation Lifecycle (VALID 2009), IEEE Computer Society, 2009, pp. 123–128. doi:10.1109/VALID.2009.15.

[17] P. Mäder, O. Gotel, Towards Automated Traceability Maintenance, Journal of Systems and Software (2012). doi:10.1016/j.jss.2011.10.023.

[18] M. A. Javed, F. U. L. Muram, U. Zdun, On-Demand Automated Traceability Maintenance and Evolution, in: Proceedings of the 17th International Conference on New Opportunities for Software Reuse (ICSR), 2018, pp. 111–120. doi:10.1007/978-3-319-90421-4_7.

[19] L. G. P. Murta, A. van der Hoek, C. M. L. Werner, Continuous and Automated Evolution of Architecture-to-Implementation Traceability Links, Automated Software Engineering 15 (2008) 75–107. doi:10.1007/s10515-007-0020-6.

[20] A. Demuth, R. Kretschmer, A. Egyed, M. Maes, Introducing Traceability and Consistency Checking for Change Impact Analysis across Engineering Tools in an Automation Solution Company: An Experience Report, in: ICSME, 2016. doi:10.1109/ICSME.2016.50.

[21] M. George, K.-P. Fischer-Hellmann, M. Knahl, U. Bleimann, C. Atkinson, Traceability in Model-Based Testing, Future Internet 4 (2012) 1026–1036. doi:10.3390/fi4041026.

[22] R. Jongeling, How to Live with Inconsistencies in Industrial Model-Based Development Practice, in: MoDELS Companion (Doctoral Symposium), 2019, pp. 642–647. doi:10.1109/MODELS-C.2019.00098.

[23] G. Buchgeher, R. Weinreich, Automatic Tracing of Decisions to Architecture and Implementation, in: Proceedings of the 2011 Ninth Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE Computer Society, 2011, pp. 46–55. doi:10.1109/WICSA.2011.16.

[24] P. Stünkel, H. König, A. Rutle, Y. Lamo, Multi-Model Evolution through Model Repair, Journal of Object Technology 20 (2021) 1:1–25. doi:10.5381/jot.2021.20.1.a2.

[25] P. Valderas, V. Pelechano, Introducing Requirements Traceability Support in Model-Driven Development of Web Applications, Information and Software Technology 51 (2009) 749–768. doi:10.1016/j.infsof.2008.09.008.

[26] R. N. Taylor, N. Medvidović, E. M. Dashofy, Software Architecture: Foundations, Theory, and Practice, Wiley, 2009. Examples from authors' slides at https://ics.uci.edu/~taylor/classes/221/13_Analysis_of_Software_Architectures.pdf.