A Survey of Federative Approaches for Model Management in MBSE

Moussa Amrani, Mittal Rakshit, Miguel Goulão, Vasco Amaral, Sylvain Guérin, Salvador Martínez, Dominique Blouin, Anish Bhobe, Yara Hallak

A Survey of Federative Approaches for Model Management in MBSE

Moussa Amrani
Moussa.Amrani@unamur.be
Faculty of Computer Science / NaDI,
University of Namur
Namur, Belgium

Rakshit Mittal
Rakshit.Mittal@uantwerpen.be
University of Antwerp
FlandersMake @ UAntwerpen
Antwerp, Belgium

Miguel Goulão
Vasco Amaral
mgoul@fct.unl.pt
vma@fct.unl.pt
FCT-UNL / NOVA LINCS
Lisbon, Portugal

Sylvain Guérin
Salvador Martínez
sylvain.guerin@imt-atlantique.fr
salvador.martinez@imt-atlantique.fr
IMT Atlantique
Brest, France

Dominique Blouin
Anish Bhobe
dominique.blouin@telecom-paris.fr
anish.bhobe@telecom-paris.fr
Télécom Paris
Palaiseau, France

Yara Hallak
yara.hallak@telecom-paris.fr
Télécom Paris
Palaiseau, France
yara.hallak@renault.com
Renault Group
Guyancourt, France

Abstract

Model-based Systems Engineering (MBSE) advocates the use of models in every stage of development, leading to large number of models that need coordination, collaboration, and discipline management. Model Management (MoM) is a possible approach to manage inter-related *collections* of models among which Model Federation (MF) provides unique capabilities, like independence of development in individual modelling domains. There is currently a lack of studies about commonalities, variabilities, and gaps in MF approaches. In this paper, we propose a survey and a critical discussion of carefully selected papers about MF. From 59 contributions collected by experts in MoM, we selected, and classified, 23 papers. We extract the main trends we observed, according to our Classification. We then critically review the Classification, and discuss important gaps found in our corpus. The survey results and artefacts are all available online.

CCS Concepts

• **General and reference → Surveys and overviews**; • **Software and its engineering → Multiparadigm languages**; • **Information systems → Information integration**.

Keywords

Model Management, Model Federation, Model-Based Systems Engineering, Feature Model, Literature Review

1 Introduction

Models are increasingly prevalent in Systems Engineering (SE), replacing traditional document-based approaches. In a nutshell, a model is an abstraction of a system (or its components) based on specific goals and properties of interest. Using models, systems engineers can systematically develop a system, optimising the effort and cost of development. This contrasts with document-based approaches where engineers write natural-language documents describing the system, making validation very difficult and time-consuming, especially with complex systems-of-systems.

Model-Based Systems Engineering (MBSE) is an approach that advocates building models for all stages of a product's life cycle, including requirements, design, analysis, verification, and validation. As more and more companies adopt MBSE to develop increasingly complex and heterogeneous systems, there is a significant loss of consistency and traceability in information *between* models in different domains, built and updated by *different engineers* at *different points of time*, for the same or different systems. Maintaining this consistency becomes especially important for safety-critical systems with inter-dependencies and relations between different models in the context of Multi-Paradigm Modelling (MPM) [14, 50].

As complexity of systems increases and MBSE is being adopted, the industry is spending more and more effort to manage large *collections* of models from across domains, concerns and levels of abstraction, with different syntaxes and semantics. For instance, as communicated by research partners at the virtual simulation group at Renault to some of the authors of this paper, hundreds of simulation platforms, each of which using hundreds of simulation models must be managed to cover the different engineering domains involved when building vehicles, and the different levels of abstraction required by each phase of development. Therefore Model Management (MoM) is essential to bridge this gap by proposing dedicated approaches, and related techniques, to (i) manage and extract information from a collection of non-/inter-related models, (ii) propagate changes from model evolution at various abstraction levels, and (iii) globally interact with and manipulate the collection of models in its entirety, instead of acting on single models and manually managing the consequences.

One of the earliest mentions of MoM [6] proposed relational operators between database schema to manage the data abstracted as models. Later, proposals for MoM with approaches like megamodels, macromodels, modelverse, etc. [13, 27, 43, 55] introduced an interesting perspective by emphasising the necessity to provide appropriate foundations, and dedicated tooling, to support these kinds of activities. While many of these approaches rely on merging, unification, or collection of the models at various meta-levels,

Model Federation (MF) is a technique that ensures that the involved modelling approaches and frameworks can continue development unhindered while still offering all advantages that come with a MoM framework.

We argue that it is the right time to collect all of these techniques in the literature, analyse their commonality and variability, and identify literature gaps that may form the basis for future research directions in MoM. To our knowledge, such a survey does not exist in the literature yet. Hence, in this paper, we propose an initial survey of the federative approaches to support MoM. This review aims to serve as the initial stage of a larger exhaustive systematic literature review on MoM, that we plan to perform. Note that while the motivation for our study was rooted in SE for Cyber-Physical Systems (CPSs), our notion of MoM is generalised to be applicable to models from various other domains.

The paper is organised as follows: Section 2 presents concepts and definitions related to MoM, MF, and Feature Models, which we use to present our classification. Section 3 describes the methodology employed in conducting the review, which resulted in our 23-paper corpus. Section 4 presents our Classification for MoM *federative* approaches present in the corpus, followed by the results, i.e. trends and gaps observed when classifying the corpus contributions in Section 5. section 6 discusses Related Work, and section 7 presents the concluding remarks. All review artefacts are provided as a Zenodo repository at https://doi.org/10.5281/zenodo.12704828.

2 Background

As preliminary material, this section describes the central notion of MoM and its relationship to MF, and offers an overview of feature modelling, as the Domain-Specific Language (DSL) used to describe our classification.

2.1 Model Management

The notion of MoM is so general that the MBSE community has yet to reach a clear agreement on its meaning. As the name suggests, the expression targets all data structures and operations (beyond simple elementary CRUD operations targeting model elements), aimed at managing a collection of models. These operations can consist of creating, retrieving (through appropriate queries), updating (according to specific needs or predefined events), deleting, and in general manipulating (to perform analysis, generate code, etc.) *collections* of models. That information may be spread over such collections of models, typically representing the same system, which makes it more complex than dealing with a single model.

Note that in MBSE, a so-called "model" may take multiple forms: it may be expressed textually or diagrammatically [56]; it may relate to instance information (i.e. directly reflecting, albeit in an abstract and simplified way, a particular system) or to typing information (what we often refer to as "metamodels" in MDE); it may represent arbitrarily complex data structures, e.g., architectural designs, complex computations (including real-time, continuous behaviours), or any of their combinations. As a result, we will in this paper, use the generic term "*artefact*" to designate models or model elements, agnostic to their form or purpose.

This section briefly reviews how the notion of MoM has been discussed in other fields of Computer Science, before focusing on general approaches for manipulating interrelated collections of models.

2.1.1 What is Model Management? The Encyclopedia of *Information Systems* discusses the need to build MoM systems that emerged, among other things, from the central use of data and decision models, which both needed dedicated tools and workflows to extract valuable information efficiently[7]. The *Business Process* community also discussed the same need, although naturally more oriented towards "dynamic" models representing processes, decisions, analyses, etc., and emphasising the crucial need for collaboration [17].

A prominent field that has extensively studied MoM is *Databases*. We find in [6, 41] (among many other contributions) that "*Model Management comprises technologies and mechanisms to support the integration, transformation, evolution, and matching of models*". This citation captures the most represented use of MoM and emphasises the importance of dealing with evolution and change.

Later on, "*a Model Management System (MMS) has to provide definitions for* models *[...],* mappings *(i.e. relationships between different models), and* operators *(i.e. operations that manipulate models and mappings)*". Here, we find a structural definition of MoM, that is similar to the high-level features of our proposed classification (cf. §4). Their "*models*" translate into our generic notion of *artefacts*; "*mappings*" into our notion of Links; and their "*operators*" are reflected in our Operational feature.

Two other crucial points are mentioned [41]. First, "*models and mappings are considered as first-class objects, and [...] operations should address them as a whole and not only one model element at a time*". This directly translates into our idea of MoM in MBSE: pairing "models" (aka. Artefacts) ***with*** "relationships" (aka. Links) enables efficiently manipulating *collections* of *interrelated* artefacts. They should, therefore, be treated as *first-class citizens*, with appropriate support for describing them through dedicated data structures and manipulating them using dedicated "operations".

Second, "*mappings*" are not just decorative, but should rather capture an actionable meaning: "*Mappings are in practice very complex and therefore, a rich mapping language is required in an MMS. The MMS must also be able to reason about the mappings. Furthermore, the MMS should not only enable the definition of a mapping, but support also its application*". For MoM in MBSE, these "operations" should act at the *collection* level (what we call the Federation), as opposed to finer-grained, classical model operations that act on model elements only: this is what enables new perspectives for integration, interoperability, global analyses, and disciplined evolution. Furthermore, because "relationships" quickly gain in complexity, this calls for dedicated ways of describing them precisely: in MBSE, this is often tackled by using DSLs.

Finally, the field of *Model-Driven Engineering* introduced the idea of dynamically maintaining *repositories* of models [4, 30, 55]: these models may represent a physical or computational reality with different scales of fidelity, and are consequently, often updated along real-life evolutions. In turn, engineers (or other models) may query these models, and perform operations and analyses on the repository. This vision is complementary with the other, as it adds an orthogonal dimension to the views proposed in the other fields.

In summary, the four fields we surveyed emphasise four main points: (i) the centrality of "links", that need to be properly modelled;

(ii) the introduction of a shift in how "operations" should work on collections, instead of only the models; (iii) the idea of collecting artefacts in a properly designed repository that may be queried, and serviced with repository-wide operations; and (iv) the highlight of important challenges associated with evolution and inevitable changes of the collection, as well as collaboration among those who manipulate it.

2.1.2 Approaches to Manage Collections of Models. Managing *collections* of interrelated *artefacts* is more complex than simply manipulating (even very large) models, especially when the *artefacts* differ in nature (meta-models vs. models; textual vs. diagrammatic), and are expressed in various formalisms (often chosen to appropriately fit the task at hand, as advocated by Multi-Paradigm Modelling [1]). For that purpose, and with interoperability in mind, the ISO Standard ISO-14258 describes several, complementary approaches [28]:

Integration All models constituting a system are integrated via the same language; in other words, each model is instantiated from a common "union-language", often obtained by merging all initial languages.

Unification All models are unified through a *pivot language*, or "super-language", that represents all the knowledge about the system. In other words, each model is related to the pivot language through bi-directional transformations, ensuring integrity and consistency between each "view" extracted from the model and the pivot language itself. Note that this approach may be called differently depending on how large the pivot language spans (cf. the notion of "System Model" in [9, 10]).

Federation All models stay in their own world and do not directly interfere with the language specification of the others (non-intrusiveness); each world is related to the others through meaningful *relations* that explicit *how* and *why* models are related to each other.

Note that the term "model" in the ISO standard [28] applies indifferently to any "kind" of models (since "everything is a model" [13]), regardless of what they represent, similar to what we described in §2.1 for the term "model" in the MBSE community. We chose to use the term *artefact* instead of " model", in our classification, to avoid confusion.

Each of the three approaches presents advantages and suffers from drawbacks, depending on the purpose of the collection (i.e. *why artefacts* are collected). We discuss them for three concerns. First, how much "knowledge" do *artefacts* need to know about the others to communicate and fulfil their purpose efficiently? Second, how are *artefacts* kept consistent when other artefacts evolve, triggering adaptation to reflect the new reality? Third, due to the previous points, how hard is it for engineers to adapt the tools associated with each *artefact* to retrieve relevant information from other *artefacts* and fulfil tasks?

Integration poses challenges immediately at the level of the union-language specification because the very semantics of the union are difficult to properly describe: not only does it require standardised languages, and associated technologies to ensure interoperability, but it is sometimes impossible to implement due to fundamental incompatibilities (e.g., semantics that are not reconcilable, such as continuous vs. discrete time domains). Depending on the number of languages in the union-language, it may become challenging to understand, track, and even represent large union-languages. Furthermore, any fundamental change in one of the integrated languages invariably triggers changes in the union-language, requiring complicated adaptations in all related artefacts. This also directly impacts all associated tools (debuggers, test and code generators, analysers, etc.) The advantages are pretty clear: only dealing with the union-language considerably simplifies artefact manipulation, at the expense of heavily preventing the independent evolution of artefacts. Similarly, there is no need to establish relationships between models, because they are specified by the union-language.

Unification partially overcomes the drawbacks of Integration, in the sense that *some* languages are kept untouched at the expense of specifying (often complex) transformations to keep track of evolution and ensure proper synchronisation between artefacts. This heavily alleviates the need to maintain the associated tools. However, a similar (if not higher) complexity is hidden in the bi-directional transformations between artefacts, by forcing engineers to think about *how* and *when* these transformations have to be triggered, i.e. enforcing a development *life-cycle* at the transformations-level, which may invariably result in inconsistencies, and possibly introduce synchronisation loops. Furthermore, a crucial question resides in the choice of a pivot meta-model that should maintain an appropriate level of abstraction to be able to represent information from other worlds without (too much) information loss. Note, finally, that *Unification* may degrade into *Integration* when the pivot meta-model covers all unified meta-models exhaustively. To compensate its drawbacks, the Unification could possibly consist of *several* (instead of a single unique) pivot meta-models: this, however, introduces extra complexity to explicitly manage relationships covering semantic gaps between the pivot meta-models.

Finally, *Federation* proposes to explicitly decouple languages to keep them fully separated and to introduce explicit relationships between them. As an obvious advantage, an MF preserves the tooling associated with each meta-model. However, a major drawback resides in managing the aforementioned relationships: they need dedicated tooling. This relationship-level tooling would likely rely on existing tools, delegating some tasks to the tools already available for each artefact. Furthermore, creating, maintaining and evolving the relationship-level tooling would likely reuse the methodologies and techniques readily available at the artefact level. Federation also possesses a strength that the other approaches cannot easily offer: since *artefacts* stay independent of each other and are connected only through the existing links, it becomes possible to dynamically manipulate the federation as a whole, therefore adding or removing new *artefacts* when necessary.

2.2 Feature Models

Feature models were first proposed to model the commonality and variability of software products in product lines [18]. The common and variable, functional and non-functional features of the software (in the product line) are modelled as a tree-graph, where the *features* constitute the nodes (starting from a *root*), and the edges represent the decomposition of the *features* with special Boolean conditions on the connected *features*.

Relations between features in the tree represent the decomposition of the features into sub-features with special Boolean conditions like AND (if the super-feature is selected, the sub-feature must compulsorily be selected), OR/XOR (at least/exactly one sub-feature can be selected). Other complex conditions (mutual exclusion, cardinalities, etc.) can be specified as *cross-tree constraints*.

A valid *feature configuration* is a single product in the product line, an 'instantiation' of the feature model that satisfies all Boolean and numerical conditions of the feature model in its selection of the various *features* from the tree.

Feature Models have a simple graphical representation with a clean semantics [44]. We therefore believe it is an appropriate language to model the characteristics and classification of the MoM approaches reviewed in this paper.

3 Survey Methodology

The goal of this paper is to identify some of the key literature on MF, then clarify the trends and extract valuable lessons in this topic. As mentioned in section 2, since no clear consensus emerged in MBSE about what MF exactly is, we needed a preliminary phase to agree on a common definition extracted from the key contributions in the literature. As a result, this paper presents an early, preliminary study on the topic of MF, as a central and important approach to MoM. It is built on a corpus of contributions manually selected by experts among the authors, as opposed to a more rigorous Literature Survey based on a systematic search of online repositories (cf. among others, [40]). We still followed the classical steps in later stages (e.g., cross-reading, snowballing) to complement the initial selection, and improve the representativeness of our corpus. The various stages are illustrated by Figure 1, and documented in an Excel sheet publicly available at https://doi.org/10.5281/zenodo.12704828.

In a first stage, a group of experts (among the authors) were tasked with selecting papers they knew were contributing to the general topic of MoM. We applied basic exclusion criteria: papers not peer-reviewed, not written in English, or under 5 pages were immediately excluded, as well as duplicate entries proposed by different experts. This resulted in an initial set of 59 papers.

The second stage had two objectives in mind: first, reaching a consensus on what MF is (at least, identifying key characteristics that could help in reading papers, and extract valuable features), as opposed to other MoM approaches (cf. §2.1.2); and second, emerging with a subset of papers that truly describe MF. To reach the first objective, authors rapidly scanned papers and described their content in natural language. A panel discussion then identified the main dimensions (as described by the ISO standard in §2.1.2, and translated as high-level features in our Classification). For the second objective, we randomly assigned all the papers to three different reviewers who were tasked to assess the relevance of the paper with the MF approach. Conflicts were resolved by discussions between reviewers, and escalated to plenary meetings when no consensus for a specific paper was reached. As a result, 27 papers were discarded for one of the following reasons: (i) the paper is not related to MoM; (ii) it discusses a MoM approach that is not Federation; or (iii) the paper does not provide a technical contribution (typically, a literature survey).

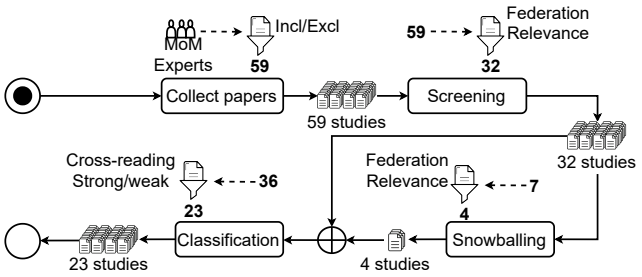

Figure 1: Search & Classification Methodology

While the first two stages were dedicated to paper selection and screening, the third and last stages were dedicated to the classification that iteratively emerged from reading the papers. Each paper was assigned to two different reviewers who classified it independently, according to the feature model described in section 4. Regular plenary meetings were conducted for reviewers to propose changes in the feature model, based on the information extracted from the papers. At this stage, 12 papers were further discarded for two reasons: (i) some were assessed to be non-relevant when read in detail; and (ii), some were describing similar contributions than already included stronger papers (typically, workshop vs. journal papers). This resulted in 20 papers that constitutes the basis of our corpus.

The fourth and final stage consisted of snowballing our base corpus (twice) to find relevant papers missed in the first stage: this resulted in 7 papers, which were again submitted to the same screening and classification processes as earlier (with still two different reviewers), among which 3 papers were finally included in the final corpus. One of these papers superseded a paper from the basic corpus of 20. As a result, our corpus for exploring the topic of MF consists of 23 papers.

4 Classification

MF brings the focus on the *links*, or *relations*, between *artefacts* (as well as the elements they contain), making them more explicit, structured, dynamic, reactive, and ultimately more easily manipulable. Furthermore, with MF, artefacts exist untouched in their own original technical and conceptual environment. Artefacts also acquire an improved semantics thanks to the relations between artefacts, which are non-existent without MF (or at least, more difficult to recreate with artefacts alone, in other approaches). Links provide semantics in multiple ways: a single artefact may contribute to several links, and thus enter in a relation with several other artefacts; while several artefacts may well contribute to the specification of the same given link.

As a result, at a very high level, an MF can simply be seen as a *graph* of links connecting artefacts, that can be studied w.r.t. three views (cf. Figure 2):

From the *structural* viewpoint are studied the graph's constituents (the nature of *vertices*; and how *edges* are organised as well as what they represent), as well as the whole graph.

From the *operational* viewpoint, are studied the processes and operations that are possible, explicitly or implicitly defined, and how they compare with each other.

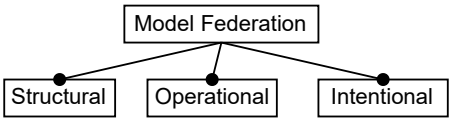

Figure 2: High-level features of Model Federation

From the *intentional* viewpoint, are studied the *goals* or *purposes* of creating the MF.

Directly inspired by Guerin's Ph.D. thesis [26], our classification refines its precursor w.r.t. three points: (i) it adds purely cosmetic, high-level features that corresponds to the viewpoints mentioned above, which will guide our presentation of the feature model in the following sub-sections, (ii) the Intentional feature is new and tries to extract the *intent* or *goal* of each MF in our corpus, (iii) it identifies additional new sub-features, and rearranges some others.

This section is a contribution of our paper, describing our classification for MF, while illustrating important features when needed, with contributions from our corpus. Classifying one of the 23 reviewed contributions using our feature model leads to a valid feature configuration (cf. §2.2) where the absence of an optional feature indicates that we were not able to extract the related information from the paper. For the complete feature model, the reader is referred to https://doi.org/10.5281/zenodo.12704829.

Note that regarding the other approaches for Model Management detailed in section 2, we do not claim that this Classification operates well on other approaches (namely, Unification and Integration): we strongly believe that these approaches are different in nature, and deserve their own classifications (although they may partially overlap with the one for MF).

4.1 Structural Features

From a structural viewpoint, we see a *Federation* as having a graph structure (cf. Figure 3):

- *Vertices* correspond to Artefacts, as well as the elements they contain; while
- *Edges* correspond to Links; and
- The structural features of the whole MF *graph* as a singular entity are captured by the Federation feature

4.1.1 Artefacts. We classify the Artefacts composing an MF according to four sub-features:

(1) the Formalism(s) used to specify the Artefacts. [32] proposes a list of Formalisms commonly used for CPS Engineering that we use as a basis for our description.
(2) the Domain(s) the Artefacts are intended to be used in. In [51], a feature model describing CPS Domains is available. We extend it with domains outside of CPS.
(3) the technological space (Tech_Space) as defined by [29] is the working context with its set of associated concepts, tools, and possibilities, in which the Artefacts are defined. In the context of MBSE, it crudely refers to the "format" (eg., the MOF meta-meta-model, RDF, textual grammars, etc.) that the Artefact(s) belong to.
(4) the Serialisation format(s) used for persisting the Artefacts

Note that the two latter sub-features are optional, as they are not always explicitly mentioned in the case of papers describing high-level/abstract MF approaches. The sub-features of the former two features are already discussed in the literature, hence they are omitted.

4.1.2 Links. The Links (a.k.a. Relations) appearing in an MF are classified according to six dimensions:

(1) the Links' Arity, which refers to the number of artefacts defining, and participating in a Link: we simply distinguish between common Binary and N-ary Links.
(2) the Granularity at which Links work. It refers to the specific 'level' of decomposition of the linked Artefacts: a Link may exist at the Model-level, meaning that its target Artefacts are whole models, or it may exist at the Element level, meaning that structural properties (or model-elements) of the Artefacts are actually linked. In the latter case, the Link may even give access to the Element's Content (i.e. the *type*, in case of a meta-model; or the *value*, in case of a model), which may be useful for e.g. to propagate values changes throughout interconnected models, or the Link may simply Reference the involved Elements, e.g. to indicate that they may be impacted by a change.
(3) the Semantics associated with the Links. This feature describes the 'meaning' of the Links, i.e. why, and for which purpose, the Artefacts are related by the Links. The Links' Semantics may be characterised by a general-purpose Typology in which we consider the following classical relation types from [21]: *IsRepresentationOf* (μ), IsDecomposedIn (δ), IsElementOf (ϵ), *conformsTo* (χ), IsTransformedIn (τ); which we extend with the classical IsTracedTo (π) type that represents trace Links. This Typology is only descriptive and non-exhaustive, hence the X_Type feature is present to make the Typology complete and extensible. A step further in describing the Semantics of Links would be to use *explicitly* Meta-modelled Links: this allows Links' types and relations to be freely and consistently created. Some tools may also allow Decomposable Link Semantics, meaning that it is possible to define Links at a given Granularity, and (semi-)automatically compute and create Links at another level of Granularity.
(4) the Links' Persistence feature characterises MF contributions where Links are persisted, meaning that it is possible to retrieve the Links independently of a modelling session, or after it has ended.
(5) the Mixed_Level feature indicates that it is possible to agnostically define Links between Artefacts from any meta-modelling-level (linking together models, meta-models, or models with meta-models).
(6) the Specification feature describes the way Links are defined. This can be fully manual, by linking individual Artefacts for example, or follow a more complex process in which correspondences are defined Manual-ly at the meta-level and then concrete links are Semi-Automatic-ally generated through a matching process. This case indeed relates to the Decomposable-ity of the Link Semantics.

Note that the last three sub-features of Links are optional, since many of the reviewed papers do not describe these details .

4.1.3 Federation. We classify an MF based on two features:

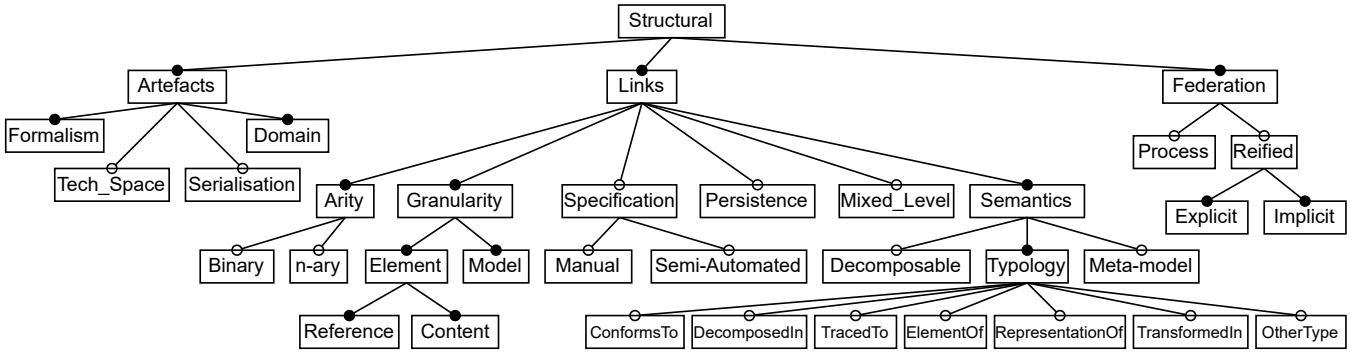

Figure 3: Structural features for Model Federation, following a graph-like decomposition in vertices (Artefacts), edges (Links), and the whole graph (Federation) (cf. §4.1).

(1) Whether the MF is Reified, meaning that the whole MF is manipulable (Artefacts *and* Links, *together*). When it is the case, the reification may be Explicit, meaning that there exists another single Artefact that explicitly encodes the entire MF, as opposed to the Implicit approach where modellers need to retrieve pieces of Artefacts they need, by themselves. Note that these two features are interdependent: a Feature Model that needs to be explicitly used for creating valid products would require an explicit so-called *cross-tree* «requires» constraint [44].
(2) We specifically classify those approaches and tools that prescribe a given development Process, meaning that creation and manipulation of the MF should follow a predefined workflow.

4.2 Operational Features

Building an MF is only useful when it enables execution of Operations and (complex) processes on it, to leverage the (large) graph of Artefacts. Usually, an MF tool, is not responsible for creating, deleting, or updating the Artefacts themselves: rather, each Artefact has dedicated domain-specific tooling for that purpose. The MF tool is however responsible for creating, deleting, and updating the Links between these Artefacts, and maintaining them in a coherent state. Therefore, usual elementary CRUD operations for MF often focus on the Links, rather than on the Artefacts. In our classification (cf. Figure 4), we focus on *where* and *which* Operations are specified, and *how* and *when* they are executed.

4.2.1 Location. The Location feature describes *where* the operations take place: they can be attached to the Links, to the Federation, or be completely External to the MF.

4.2.2 Operation. The Operation feature describes *which* Operations are executed on the MF. Our classification lists some common Operations. Representing, Querying, and Constraining Operations on MFs are used to visually display and retrieve models on and from the MF. Syncing an MF aims at resolving inconsistencies between Artefacts. V&Ving generically designates operations for validating and verifying properties over the MF. Note that this list of operations is generalised and not exhaustive.

4.2.3 Trigger. The Trigger is important as it specifies *when* an Operations shall be executed. An Operation may be triggered Manual-ly and on-demand, or may triggered Reactive-ly to specific events on some parameters from the concerned Artefacts or the MF itself (typically, modifications in the MF).

4.2.4 Execution Mode. (a.k.a Exec_Mode) specifies *how* the Operation works. It may work in an Incremental fashion, meaning that whenever it is executed, only the new changes (typically, Links) are computed and/or updated, as opposed to a full Rebuild where everything is recomputed from scratch at each operation invocation.

4.3 Intentional Features

An MF is rarely built without a particular application usage in mind, that requires reasoning about the MF as a whole: investing the effort in developing and maintaining an MF needs to be justified by substantial gains. However, it is often difficult to extract this motivation directly from the contributions: these goals are rarely explicitly mentioned, and inferring this information from the papers is not an exact science (unless the authors are asked directly). Furthermore, considering the MF field is still in its infancy, it is difficult to define "*intents*" for designing MF systems unlike the field of Model Transformations where *purpose* is characterised via a set of *properties* [34].

On the other hand, we believe that understanding *why* an MF system is built, is essential to evaluate and compare different contributions. Understanding these objectives which, in turn, could be amenable to further characterisation through appropriate properties, paves the way to precise comparisons. Our classification presents such an initial attempt (towards a fully-fledged list of intents) at listing all the possible Intentional features of MF that we encountered in our corpus:

Traceability refers to the ability to track and link various stages of the software or system development life-cycle. It includes various activities, such as requirement assignment and verification, impact analysis, compliance and auditing, and project management.
Model consistency checking and repair are essential practices in MBSE. Model consistency checking involves verifying that different models do not contradict each other and adhere to predefined rules and constraints, while model repair focuses on automatically or semi-automatically modifying inconsistent model to restore overall consistency.

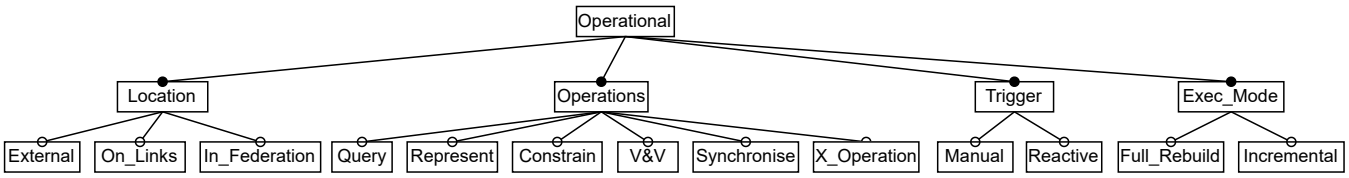

Figure 4: Operational features for Model Federation: where they are defined, and how and when they are executed.(cf. §4.2).

Unified transformation management refers to all tool chains used in software or system production (code generation, reverse engineering, model-to-model transformations). The persistence of transformations enables fine-tuned, incremental management and production traceability of artefacts in a production pipeline.

Model composition is a useful modelling activity. For example, in MBSE, where the aim is to build a model of a system by combining existing sub-system models even when the sub-system models are well-developed within their heterogeneous modelling domain(s) or paradigm(s).

Cross-domain analysis involves examining and integrating concepts, methodologies, or technologies from multiple domains to resolve or optimise cross-functional concerns, such as cyber-security, usability, process metrics optimisation, scenario testing, etc.

Artefact co-design corresponds to the co-construction of one or more models and their associated meta-model(s), where the model-to-meta-model conformance relationship is not hard-coded in a tool but meta-modelled itself.

Model edition allows to edit a model using another model: a graphical editor, for example, allows to build a model using a diagram and graphical shapes (another kind of model).

Conceptual elicitation / reverse engineering Such activities also involve several models, where the aim is to build a conceptual model from various technical artefacts.

5 Results

5.1 Threats to Validity of Methodology

Our study has three main threats to validity: the *creation of the corpus*, the *elaboration of the classification*, and the *screening phase*.

We tried our best to carefully select contributions representative of MF. However, the screening phase still discarded some initially proposed papers (because they described MoM approaches other than *Federation*, cf. §2.1.2 and 3 for details), essentially due to a lack of clear definition of MF. The initial selection was by no means systematic, but rather *ad hoc*, based on the opinion of expert authors: one just completed his Ph.D. on the very same topic [26] after having worked for about 8 years on an MF System called OpenFlexo [5]; while three authors have 10+ years of experience in MDE, and studied MoM and MF for several years. We mitigated this issue by iteratively adding new contributions and systematically snowballing papers to retrieve missing contributions, as they illustrated novel approaches.

The classification proposed in §4 is based on [26]: it was heavily discussed, tested over sample contributions early on, and eventually improved (by moving features to more appropriate locations) and completed (with features retrieved from this survey). Our comparison with other fields (cf. §2.1) seems to support a similar view on at least how MF is structured.

Finally, the screening phase may have rejected relevant papers, or included non-relevant ones, because the papers classification occurred concurrently with building the feature model. We mitigated this issue by cross-reading, followed by cross-classification, both resulting in discussion (either between readers/reviewers, or in plenary sessions, cf. §3). We provide all the stage-wise artefacts for public review and scrutiny at https://doi.org/10.5281/zenodo.12704829.

5.2 Classification of Literature

In light of the threats to validity mentioned earlier, an *orthogonal* analysis (i.e. crossing the results of several dimensions of the Classification) does not make much sense. We instead offer a (preliminary) *vertical*, high-level analysis, focusing on general tendencies.

5.2.1 Structural Features. The Artefacts feature was the most surprising: we were expecting to find contributions and techniques specialised to specific Formalisms and Domains, but almost all contributions can deal with any formalism (assuming it belongs to the appropriate Tech_Space), and are agnostic of domains. A few contributions explicitly mention Cyber-Physical Systems as a target domain [42, 54], with approaches easily extensible to others domains. Furthermore, the vast majority targets the standardised OMG EMOF (and its EMF implementation Ecore [46]) as their Tech_Space, at the exception of two (FTG+PM [38], and DesignSpace [19, 42], both using *ad hoc*, internal representations based on graphs). Note that some approaches also offer so-called "*connectors*", i.e. adaptors intended to CRUD and communicate with other models in different formats and tech spaces [22, 33, 42, 45]. For Serialisation, many contributions rely on XMI/XML: either because they use Ecore, for which it is the default format, although it can be easily changed, or because they use a modelling tools also serialising in XML (e.g., DrawIO/Diagrams for FTG+PM [38]). A few contributions use specific serialisation formats: Epsilon [33] relies on a dedicated, domain-specific format; and NAOMI [20] also uses XMI, paired with an explicit version control. Some contributions are totally agnostic from serialisation, such as OpenFlexo [22–24].

The Links feature is perhaps the most heterogeneous in our classification. Less than half of the contributions use n-ary Links. Surprisingly, less contributions allow Links to connect any type of models than those that specialise in a particular type (meta-model, or model), because most modelling tools allow nowadays to manipulate meta-models just like any models. This requires further investigation. Most contributions also allow a fine-grained manipulation of model elements and among those, a vast majority offer value manipulations to such elements. In our understanding, this

is particularly important in light of model synchronisation/repair processes (cf. [12, 22, 42, 45], among others). The semantics and processes associated to Links is not always explicitly stated, or illustrated, so it was often difficult to extract the relevant information from the papers. Most of the contributions use explicitly Meta-modelled, and Decomposable Links. One approach includes the Links specification for the whole MF [43]; and [20] only offers a monolithic Links type. Many contributions allow to persist Links with several variations. For example, some only persist their specification, but not the actual Links [8, 9]; and another [12] stores Links inside a weaved model. We could not retrieve the information for two contributions [37, 52]. Only [33] imposes full Links rebuild at each modelling session .

For the Federation feature, only one contribution explicitly enforces a development Process [39] (with one [52] not clearly stating it).

5.2.2 Operational Features. The Operational features describe what, how and when operations on Artefacts, and on the MF, are specified and performed. It was difficult to extract the set of operations available for each tool/approach of our corpus: the contributions' presentation does not necessarily follow this angle, or focus on this viewpoint. We noticed that a vast majority of operations were performed with classical MDE model transformations languages, or general-purpose programming languages. Aside from CRUD operations, the most represented ones target *synchronisation*, *consistency checking*, and *traceability*. In Epsilon [33], a Domain-Specific Language is used for linking model elements through constraints that may be used to trigger specific operations. In [54], the authors propose an original approach for managing consistency between heterogeneous models based on contracts.

Operations are specified almost equally on Links and as External resources, with a few approaches allowing specifications on Links and on Federation altogether [23, 24, 31, 55]. The way they are triggered is not always clearly defined, but for those that are, they are also almost equally distributed between Manual and Reactive, with a few standing contributions allowing both (cf. [8, 24, 31, 38, 45, 55] for the most significant contributions). Finally, we notice a clear tendency towards Incremental execution modes, with Full_Rebuild contributions [39, 49] focused on providing traceability and interoperability between Artefacts from different domains.

5.2.3 Intentional Features. The Intentional feature focuses on methodological aspects and purposes of the MF. As mentioned in §4.3, extracting this information was difficult, since most contributions do not explicitly state them, and are often generic so that they can address multiple purposes.

Unsurprisingly, the majority of contributions address Traceability needs (*e.g.*, methodological aspects, impact analysis, software production) [2, 23, 24, 42, 49]. Model consistency checking (often combined with repair features) is also covered by a large number of contributions [33, 42, 54]. The need for independence of federated model life-cycles means that we need to take an interest in the life-cycle of the whole federated model. This feature, described as Unified transformation management, is treated in some approaches, which have in common the fact that the links carry one or more transformations, such as [22, 38, 39]. Multi-model design, described as Model composition, are explored by numerous contributions, especially in the context of MBSE. These include for example DesignSpace [19], NAOMI [20], OpenFlexo [22], FTG+ PM [38], Reactive Links [42].

Cross- domain analysis is less represented as classical approaches do not require write-access to models, and therefore does not impose MF scheme (unless analysis is not coupled with refactoring/optimising actions which obviously require data modification). This intentional feature is generally coupled with some other intentions [37, 38]. Model/meta-model co-design is illustrated in [26] with *Free Modelling* but is poorly represented in our corpus. Model edition is covered by numerous approaches, where models are indirectly edited by other models thanks to reactive behaviours [42] or by the design of virtual models such as in EMF Views [12] or Openflexo [24]. This is often a secondary or indirect feature, coupled with some other usage requirements, such as hiding complexity to users or simplifying edition. Finally Conceptual elicitation is illustrated in [23, 24], where plain documentation serves as a support to extract conceptual and categorised information.

5.3 Trends from Literature

We observe that the Structural specification of MF as a *graph* (cf. §4.1) is indeed present in our corpus, although many contributions do not extract full value of their MF. Contrary to our belief, no contributions have specific Formalisms or Domains they specifically target. Since MBSE leverages software language engineering techniques from MDE, models are manipulated through their metamodel specification, regardless of their domain, as long as they fit the appropriate Tech_Space. Serialisation is an important issue to persist Artefacts *and* Links, which is not particularly an issue for models. However the conceptual and technical aspects dedicated to Links still need progress to reach a standardised approach. As we suggested in the Classification, a possible approach would be to *model* Links explicitly, which closes the loop.

As we expected however, MF as an approach still presents challenges and issues, and still has room for many improvements to fully leverage an MF system and the development efforts it requires: Reified MFs are still minor and not Explicit, meaning that the early visions for MF [13, 27] are not a widespread reality, even 15 years later. MF could also radically improve with a more disciplined approach towards operations specification: having operations working as *services* over the whole MF, supported by, and Triggering Links operations performing more elementary processing, could be an interesting architectural design.

5.4 Identified Gaps in Literature

As system complexity is increasing, new MoM-related needs emerge, which are not well covered by the approaches we studied. For instance, *model versioning* and *access control* are only addressed by a few of the approaches such as DesignSpace [42]. In a more general perspective, the *Authoritative Source of Truth* (ASoT)[1] must be managed. It involves managing rules to proclaim a digital artefact (model) is valid and originates from a legitimate source, and covers several aspects such as the authorisation to access, analyse, and use valid digital artefacts for diverse organisations often at distributed locations, and serves in conflict resolution.

[1] https://www.omgwiki.org/MBSE/doku.php?id=mbse:authoritative_source_of_truth

Needs for model validity management are also emerging. As mentioned in the introduction, thousands of simulations models are used and it must be ensured that what is being modelled meets the model's approximations, and that the model provides the required accuracy for the given development phase [36].

Finally, the emergence of Digital Twins imposes managing models at runtime. More complex MoM operations may be required such as switching from one model to another, without interrupting the running system, when a model is no longer valid due to a change in the state of the system or its environment.

6 Related Work

Several surveys related to MoM have been contributed in the last decade. In [3], the authors discuss realisation techniques for multi-view specification environments. They focus on the nature of the views (e.g., projective vs. synthetic) and correspondences (implicit vs. explicit) and provide a few examples of canonical approaches. In [53], the authors contribute a systematic literature review focused on the consistency capabilities of cross-domain MoM tools, identifying a number of shortcomings such as lack of maturity. Focusing on model view management, the authors of [11] provide a feature model to classify the approaches of a survey. Similarly, but taking into account realisation techniques such as in [3], the authors of [15] provide another feature model, which is used as as a taxonomy for the classification of model-view contributions in a systematic review, and includes additional dimensions for the assessment of the approaches w.r.t. used research methods, obtained evidence and limitations. Finally, conceived as a tertiary study (that is, they are based on some of the aforementioned surveys and systematic reviews) and focused on the application of model management for model repair, yet another feature model is proposed in [48].

With respect to the aforementioned research works, our survey extends the discussion by: 1) specially focusing on federative approaches; 2) adding additional classification dimensions such as process and intention, which were not developed in previous work; 3) including more recent contributions uncovered in previous work.

Note that we consider general techniques such as bidirectional and multi-directional transformations [16, 35, 47] as enablers of model management but not as model management proper and thus, we do not discuss works related to these techniques here. Similarly, we view model management as distinct from (while being an enabler of) multi-formalism modelling [25].

7 Conclusion

In this paper, we presented a classification of Model Federation (MF) approaches as a feature model. Our classification is split into structural, operational, and intentional features of MF. We reviewed and classified a corpus of 23 papers on MF, based on the proposed feature model. Our study shows that MF is still in its infancy and that while model consistency, model views are widely present, other important needs such as model version, access control, ASoT and validity are poorly addressed. This will help define new research directions in MoM.

Future work is aimed at conducting a full-scale systematic literature review of MoM, by re-using the feature model developed in this paper and the lessons learnt, while also considering, when available, approaches used by industry.

References

[1] Moussa Amrani, Dominique Blouin, Robert Heinrich, Arend Rensink, Hans Vangheluwe, and Andreas Wortmann. 2020. Multi-Paradigm Modelling for Cyber-Physical Systems: A Descriptive Framework. *Software and Systems Modelling* 20, 1 (2020), 611–639.

[2] Hazeline U Asuncion and Richard N Taylor. 2009. Capturing custom link semantics among heterogeneous artifacts and tools. In *2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*. IEEE, 1–5.

[3] Colin Atkinson, Christian Tunjic, and Torben Möller. 2015. Fundamental Realization Strategies For Multi-View Specification Eanvironments. In *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*. IEEE, Adelaide, SA, Australia, 40–49.

[4] Önder Babur, Loek Cleophas, and Mark van den Brand. 2022. SAMOS-A framework for model analytics and management. *Science of Computer Programming* 223 (2022), 102877.

[5] Jean-Christophe Bach, Antoine Beugnard, Joel Champeau, Fabien Dagnat, Sylvain Guérin, and Salvador Martínez. 2024. 10 Years of Model Federation with OpenFlexo: Challenges and Lessons Learned. In *International Conference on Model Driven Engineering Languages and Systems (MoDELS)*.

[6] Philip A. Bernstein. 2003. Applying Model Management to Classical Meta Data Problems. In *Conference on Innovative Data Systems Research*. https://api.semanticscholar.org/CorpusID:5859503

[7] Robert Blanning. 2003. *Encyclopedia of Information Systems*. Academic Press, Chapter Model Building Process, 181–191.

[8] Dominique Blouin, Yvan Eustache, and Jean-Philippe Diguet. 2014. Extensible Global Model Management with Meta-model Subsets and Model Synchronization. https://doi.org/10.13140/2.1.1947.2329

[9] Artur Boronat, Alexander Knapp, José Meseguer, and Martin Wirsing. 2009. *What Is a Multi-modeling Language?* Springer Berlin Heidelberg, 71–87. https://doi.org/10.1007/978-3-642-03429-9_6

[10] Manfred Broy, Maria Victoria Cengarle, and Bernhard Rumpe. 2006. *Semantics of UML – Towards A System Model for UML: The Structural Data Model*. Technical Report TUM-I0612. Technische Universtität München.

[11] Hugo Bruneliere, Erik Burger, Jordi Cabot, and Manuel Wimmer. 2019. A feature-based survey of model view approaches. *Software & Systems Modeling* 18 (2019), 1931–1952.

[12] Hugo Bruneliere, Jokin Garcia Perez, Manuel Wimmer, and Jordi Cabot. 2015. *EMF Views: A View Mechanism for Integrating Heterogeneous Models*. Springer International Publishing, 317–325. https://doi.org/10.1007/978-3-319-25264-3_23

[13] Jean Bézivin. 2005. On The Unification Power Of Models. *Journal of Software & Systems Modeling* 4 (2005), 171–188.

[14] Paulo Carreira, Vasco Amaral, and Hans Vangheluwe (Eds.). 2020. *Foundations of Multi-Paradigm Modelling For Cyber-Physical Systems*. Springer, Cham, Switzerland.

[15] Antonio Cicchetti, Federico Ciccozzi, and Alfonso Pierantonio. 2019. Multi-view approaches for software and system modelling: a systematic literature review. *Software and Systems Modeling* 18 (2019), 3207–3233.

[16] Anthony Cleve, Ekkart Kindler, Perdita Stevens, and Vadim Zaytsev. 2018. Multidirectional Transformations and Synchronisations (Dagstuhl Seminar 18491): report from Dagstuhl Seminar 18491. *Dagstuhl Reports* 8, 12 (2018).

[17] Fred A. Cummins. 2016. *Building the Agile Enterprise, With Capabilities, Collaborations and Values*. Morgan Kaufmann.

[18] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. 2005. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice* (2005).

[19] Andreas Demuth, Markus Riedl-Ehrenleitner, Alexander Nöhrer, Peter Hehenberger, Klaus Zeman, and Alexander Egyed. 2015. DesignSpace: an infrastructure for multi-user/multi-tool engineering. In *Symposium on Applied Computing* (Salamanca, Spain). ACM, New York, NY, USA, 1486–1491. https://doi.org/10.1145/2695664.2695697

[20] Trip Denton, Edward Jones, Srini Srinivasan, Ken Owens, and Richard W. Buskens. [n. d.]. *NAOMI – An Experimental Platform for Multi–modeling*. Springer Berlin Heidelberg, 143–157. https://doi.org/10.1007/978-3-540-87875-9_10

[21] Jean-Marie Favre and Tam NGuyen. 2005. Towards a Megamodel to Model Software Evolution Through Transformations. *Electronic Notes in Theoretical Computer Science* 127, 3 (2005), 59–74.

[22] Fahad R. Golra, Antoine Beugnard, Fabien Dagnat, Sylvain Guerin, and Christophe Guychard. 2016. Addressing modularity for heterogeneous multi-model systems using model federation. In *Companion Proceedings of the 15th International Conference on Modularity* (Málaga, Spain) *(MODULARITY Companion 2016)*. Association for Computing Machinery, New York, NY, USA, 206–211. https://doi.org/10.1145/2892664.2892701

[23] Fahad R. Golra, Antoine Beugnard, Fabien Dagnat, Sylvain Guerin, and Christophe Guychard. 2016. Continuous Requirements Engineering Using Model

Federation. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*. 347–352. https://doi.org/10.1109/RE.2016.42

[24] Fahad Rafique Golra, Fabien Dagnat, Jeanine Souquières, Imen Sayar, and Sylvain Guerin. 2018. *Bridging the Gap Between Informal Requirements and Formal Specifications Using Model Federation.* Springer International Publishing, 54–69. https://doi.org/10.1007/978-3-319-92970-5_4

[25] Marco Gribaudo, Marco Gribaudo, and Mauro Iacono. 2013. *Theory and Application of Multi-Formalism Modeling* (1st ed.). IGI Global, USA.

[26] Sylvain Guerin. 2023. *FML: A Model Federation Language For Semantic Interoperability of Heterogeneous Information Sources.* Ph. D. Dissertation. École Nationale Supérieure de Techniques Avancées Bretagne.

[27] Regina Hebig, Andreas Seibel, and Holger Giese. 2011. On The Unification of Megamodels. In *International Workshop on Multi-Paradigm Modeling (MPM)*.

[28] International Organization for Standardization. 2014. ISO:14258:2014: Industrial Automation Systems: Concepts and Rules for Enterprise Models.

[29] Ivan Ivanov, Jean Bézivin, and Mehmet Aksit. 2002. Technological spaces: An initial appraisal. In *4th International Symposium on Distributed Objects and Applications, DOA 2002*.

[30] Alireza Khalilipour, Fatma Bozyigit, and Moharram Challenger. 2024. Intelligent Model Management based on Textual and Structural Extraction-An Exploratory Study. In *2024 10th International Conference on Web Research (ICWR)*. IEEE, 165–174.

[31] Heiko Klare, Max E. Kramer, Michael Langhammer, Dominik Werle, Erik Burger, and Ralf Reussner. 2021. Enabling consistency in view-based system development — The Vitruvius approach. *Journal of Systems and Software* 171 (2021), 110815. https://doi.org/10.1016/j.jss.2020.110815

[32] Stefan Klikovits, Rima Al-Ali, Moussa Amrani, Ankica Barisic, Fernando Barros, Dominique Blouin, Etienne Borde, Didier Buchs, Holger Giese, Miguel Goulao, Mauro Iacono, Florin Leon, Eva Navarro, Patrizio Pelliccione, and Ken Vanherpen. 2019. COST IC1404 WG1 Deliverable WG1.1: State-of-the-art on Current Formalisms used in Cyber-Physical Systems Development. (2019). https://zenodo.org/record/2533455

[33] Dimitrios Kolovos, Richard Paige, and Fiona Polack. 2008. Detecting and Repairing Inconsistencies across Heterogeneous Models. In *International Conference on Software Testing, Verification, and Validation*. 356–364. https://doi.org/10.1109/ICST.2008.23

[34] Levi Lúcio, Moussa Amrani, Jürgen Dingel, Leen Lambers, Rick Salay, Gehan Selim, Eugene Syriani, and Manuel Wimmer. 2016. Model Transformation Intents and Their Properties. *Journal of Software And Systems (SoSyM)* 15 (2016), 647–684. https://doi.org/10.1007/s10270-014-0429-x

[35] Rakshit Mittal, Dominique Blouin, Anish Bhobe, and Soumyadip Bandyopadhyay. 2022. Solving the instance model-view update problem in AADL. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems* (Montreal, Quebec, Canada) *(MODELS '22)*. Association for Computing Machinery, New York, NY, USA, 55–65. https://doi.org/10.1145/3550355.3552396

[36] Rakshit Mittal, Raheleh Eslampanah, Lucas Lima, Hans Vangheluwe, and Dominique Blouin. 2023. Towards an Ontological Framework for Validity Frames. In *International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 801–805. https://doi.org/10.1109/MODELS-C59198.2023.00128

[37] Jörg Niemöller, Leonid Mokrushin, Konstantinos Vandikas, Stefan Avesand, and Lars Angelin. 2013. Model Federation and Probabilistic Analysis for Advanced OSS and BSS. In *International Conference on Next Generation Mobile Apps, Services and Technologies*. 122–129. https://doi.org/10.1109/NGMAST.2013.30

[38] Randy Paredis, Joeri Exelmans, and Hans Vangheluwe. 2022. Multi-Paradigm Modelling For Model Based Systems Engineering: Extending The FTG + PM. In *Annual Modeling and Simulation Conference (ANNSIM)*. 461–474. https://doi.org/10.23919/ANNSIM55834.2022.9859391

[39] Oscar Pastor, Giovanni Giachetti, Beatriz Marín, and Francisco Valverde. 2013. *Automating the Interoperability of Conceptual Models in Specific Development Domains.* Springer Berlin Heidelberg, 349–373. https://doi.org/10.1007/978-3-642-36654-3_14

[40] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update. *Information and software technology* 64 (2015), 1–18.

[41] Christoph Quix. 2018. *Encyclopedia of Database Systems.* Springer, Chapter Model Management, 2288–2293.

[42] Cosmina-Cristina Rațiu, Wesley KG Assunção, Edvin Herac, Rainer Haas, Christophe Lauwerys, and Alexander Egyed. 2024. Using reactive links to propagate changes across engineering models. *Software and Systems Modeling* (2024), 1–27.

[43] Rick Salay, John Mylopoulos, and Steve Esterbrook. 2009. Using Macromodels to Manage Collections of Related Models. In *International Conference on Advanced Information Systems Engineering*. Springer-Verlag, Amsterdam, The Netherlands, 141–155.

[44] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. 2007. Generic Semantics Of Feature Diagrams. *Journal of Computer Networks* 51, 2 (2007), 456–479.

[45] Andreas Seibel, Stefan Neumann, and Holger Giese. 2009. Dynamic hierarchical mega models: comprehensive traceability and its efficient maintenance. *Software & Systems Modeling* 9, 4 (Dec. 2009), 493–528. https://doi.org/10.1007/s10270-009-0146-z

[46] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. 2008. *EMF: Eclipse Modeling Framework.* Addison-Wesley.

[47] Perdita Stevens. 2017. Bidirectional transformations in the large. In *International Conference on Model Driven Engineering Languages and Systems (MoDELS)*. IEEE, 1–11.

[48] Patrick Stünkel, Harald König, Adrian Rutle, and Yngve Lamo. 2021. Multi-model evolution through model repair. (2021).

[49] Masoumeh Taromirad, Nicholas Drivalos Matragkas, and Richard F. Paige. 2013. Towards a Multi-Domain Model-Driven Traceability Approach. In *Proceedings of the 7th Workshop on Multi-Paradigm Modeling co-located with the 16th International Conference on Model Driven Engineering Languages and Systems, MPM@MoDELS 2013, Miami, Florida, September 30, 2013.* 27–36.

[50] Bedir Tekinerdogan, Dominique Blouin, Hans Vangheluwe, Miguel Goulão, Paulo Carreira, and Vasco Amaral (Eds.). 2021. *Multi-Paradigm Modelling Approaches For Cyber-Physical Systems.* Academic Press, London, United Kingdom.

[51] Bedir Tekinerdogan, Rakshit Mittal, Rima Al-Ali, Mauro Iacono, Eva Navarro-López, Soumyadip Bandyopadhyay, Ken Vanherpen, Ankica Barišić, and Kuldar Taveter. 2021. Chapter 3 - A feature-based ontology for cyber-physical systems. In *Multi-Paradigm Modelling Approaches for Cyber-Physical Systems.* Academic Press, 45–65. https://doi.org/10.1016/B978-0-12-819105-7.00008-8

[52] Martin Törngren, Ahsan Qamar, Matthias Biehl, Frederic Loiret, and Jad Elkhoury. 2014. Integrating viewpoints in the development of mechatronic products. *Mechatronics* 24, 7 (Oct. 2014), 745–762. https://doi.org/10.1016/j.mechatronics.2013.11.013

[53] Weslley Torres, Mark GJ Van den Brand, and Alexander Serebrenik. 2021. A systematic literature review of cross-domain model consistency checking by model management tools. *Software and Systems Modeling* 20 (2021), 897–916.

[54] El Hadji Bassirou Toure, Ibrahima Fall, Alassane Bah, and Mamadou Samba Camara. 2018. *Megamodel Consistency Management at Runtime.* Springer International Publishing, 257–266. https://doi.org/10.1007/978-3-319-72965-7_24

[55] Yentl Van Tendeloo and Hans Vangheluwe. 2017. The Modelverse: A tool for Multi-Paradigm Modelling and simulation. In *Winter Simulation Conference (WSC)*. 944–955. https://doi.org/10.1109/WSC.2017.8247845

[56] Manuel Wimmer and Gerhard Kramler. 2005. Bridging Grammarware And Modelware. In *Satellite Events at the MoDELS 2005 Conference*, Vol. 3844. Springer-Verlag, Montego Bay, Jamaica, 159–168.