MoM (Model Management) Challenge 2025

Rakshit Mittal[1], Sylvain Guérin[2], Dominique Blouin[3], Moussa Amrani[4], Salvador Martinez[2], Arne Lange[5], and Thomas Weber[5]

[1] University of Antwerp - Flanders Make, Belgium
rakshit.mittal@uantwerpen.be
[2] IMT Atlantique, France
(sylvain.guerin, salvador.martinez)@imt-atlantique.fr
[3] Télécom Paris, Institut Polytechnique de Paris, France
dominique.blouin@telecom-paris.fr
[4] University of Namur, Belgium
moussa.amrani@unamur.be
[5] Karlsruhe Institute of Technology, Germany
(arne.lange, thomas.weber)@kit.edu

Abstract: This challenge is intended to allow the demonstration of Model Management (MoM) techniques and enable the comparison of submissions, and hence framework/language/tools and their capabilities. The MoM community is invited to respond to this challenge with submissions describing solutions to this challenge expressed in a technology of their choice. Authors should emphasize the merits and limitations of their solution according to the criteria defined in this challenge description. This challenge aims to lay down the foundations of the theory, common techniques, and identify commonly proposed, highly expected, and still not proposed features for MoM approaches and tools.

1 Motivation

1.1 Introduction to Model Management (MoM)

Model-Based Systems Engineering (MBSE) is a technique for systematically specifying, designing, developing, implementing and maintaining increasingly complex systems at lower costs with better quality throughout their entire life cycle, which is at the heart of the International Council on Systems Engineering (INCOSE) *Systems Engineering Vision 2035* [9]. These (cyber-physical) systems often combine multi-physics systems (mechanical, electrical, hydraulic, biochemical, etc.) with computational systems (control, signal processing, etc.), often with human stakeholders in the loop [16]. In this view, Multi-Paradigm Modeling (MPM) [14], which proposes to model every part and aspect of such complex systems *at the most appropriate level(s) of abstraction while using the most appropriate formalism(s), and using explicitly modelled workflows and architectures* offers a foundational framework for gluing the several multi-disciplinary components together in a consistent way.

Consequently, MBSE supported by MPM requires many heterogeneous models to cover the many aspects or domains of systems, and the many required levels of abstraction. Extended enterprises, which consist of a network of companies combining their economic output to provide products and services, also require the manipulation

of multiple models. Similarly, the Industry 4.0 and the emergence of Digital Twins also involve many models.

Model Management (MoM) is, simply put, the set of activities undertaken in order to plan the design and maintenance of, organize, and control the different models that contribute to the development and maintenance of a system throughout its life-cycle.

Model consistency is a critical challenge that this so-called MoM aims to solve; but MoM encompasses many other activities [3] beyond preserving model consistency, such as managing model views, model validity, model versions, model changes (especially in the context of collaborative modelling), and development workflows, which orchestrate activities performed on these models.

While several approaches, such as megamodels [5], model unification, view-based modelling [13], multi-level modelling [17] and model federation [4] have been explored, there is yet no unified theory or approach to MoM. Due to the urgent need for MoM expressed by the industry (for example, the keynote at the first MoM workshop, by Thierry Le Sergent from ANSYS), several tools are currently emerging from industry [8], [6], [12], [7], [15], which are trying to solve the problem in an ad-hoc way.

1.2 Motivations for this challenge

The challenge aims not only to compare MoM approaches, frameworks, and tools, but also to lay the foundations of the theory, common techniques, and identify commonly proposed, highly expected, and novel features for MoM. The challenge does not aim to be a competition but a forum to gather interested stakeholders in MoM while providing an opportunity for them to discuss aspects of their approach to MoM, with a level playing field.[6]

2 Case Description - Satellite Configuration

Different artefacts contribute to the development and management of complex engineered (cyber-physical) systems. We are inspired by a recent visit to the Royal Belgian Institute for Space Aeronomy (BIRA-IASB) in the design of the artefacts of this challenge. The artefacts revolve around a satellite design and configuration case study. Examples associated with the configuration of aeronautical systems have also been used before, for example, in the tutorials for the Ontological Modelling Language[7] [7].

2.1 Artefacts

The artefacts for the satellite configuration model management challenge are available in this public GitHub repository[8].

[6] It would be remiss not to acknowledge that we are inspired and informed by two existing challenges of the MBSE/MDE community: the MULTI Challenges [2] on multi-level modelling and TTC (Transformation Tool Contest) [1] for comparing and benchmarking model transformation tools.

[7] `https://www.opencaesar.io/oml-tutorials/`

[8] `https://github.com/mom-challenge/satellite-config`

In complex systems engineering – designs, requirements, analyses, and reports are often spread across various files and formats, with semantics expressed in different (modelling) languages. Keeping these artefacts consistent, traceable, and up-to-date is a significant challenge. This set of example artefacts is designed to highlight these issues.

- **Part catalogue** (ontology.owl): We provide an OWL[9] (Web Ontology Language) file defining a vocabulary for satellite components. It specifies types of components (e.g., '*ThrusterComponent*', '*SolarPanelComponent*') and their properties (e.g., '*hasMass*' and '*componentID*'). It also contains individuals (specific instances of components) with their assigned mass values. The ontology (and corresponding knowledge graph) acts as a product/part catalogue and additionally establishes a common terminology for the various stakeholders of the system, and for the current challenge.
- **System Decomposition / Architecture** (system_config.json): We provide a JSON[10] file describing a specific satellite configuration, named '*SatAlphaV1*'. It lists the components used in this configuration, their quantities, and their individual masses (which should be consistent with the part catalogue file). It also includes a '*calculated_total_mass_kg*' property which is the sum of all parts that a system or subsystem is composed of. This specification essentially defines a 'bill of materials' for a specific system instance and aggregates properties like total mass, while also describing the architectural system decomposition.
- **Requirements Specification** (requirements.csv): We provide a CSV (Comma Separated Values) file outlining design requirements for the satellite system. For example, it specifies a maximum allowable total system mass ('*REQ001*') and a maximum allowable mass for any single component ('*REQ002*'). Such a specification serves to describe constraints that the system design must satisfy.
- **Generated Report** (report.md): We provide a Markdown file summarizing the '*SatAlphaV1*' system's properties and the verification results of the specified requirements from the requirements specification. The report is a means to provide a human-readable format of the results of validation of the system design for the non-technical stakeholders of the engineering organization.

Note that we aimed to provide the artefacts in the simplest format with the semantics as described above. If your tool or framework does not support the above formats, you are free to re-implement the above artefacts in a format/notation/language of your choice, while respecting the same semantics. We invite you to contribute such new artefacts to the challenge Git repository via a pull request.

In the following section, we describe the model management scenarios of this challenge, without committing to any specific file/artefact format.

2.2 Model Management Scenarios

1. **Change Impact & Consistency**

[9] `https://www.w3.org/TR/owl2-overview/`
[10] `https://www.json.org/json-en.html`

The mass of a component defined in the part catalogue changes. In this scenario, the mass of the thruster component changes, because of a change in fuel that is needed for longer flights. Therefore, the '*MainThruster001*' (with ID '*T001*'), originally 150.5 kg, is found to be heavier, now weighing 165.0 kg.

- Consequently, the '*mass_kg_per_unit*' for component '*T001*' in the architecture specification needs to be updated to 165.0 .
- Consequently, the '*calculated_total_mass_kg*' in the architecture specification needs to be recalculated and updated.

2. **Unified Querying & Validation**
   We need to verify that the current '*SatAlphaV1*' configuration, as defined in the architecture specification, meets the mass requirement '*REQ001*' (total system mass $\leq$ 350 kg) described in the requirements specification.
   - The report should reflect this verification.
   - If the '*calculated_total_mass_kg*' in the architecture specification is (automatically or manually) updated to 360.0 kg (due to the component change from the previous scenario) the requirement is "*NOT SATISFIED*". This needs to be updated in the report.
3. **Generation of Views**
   The architecture decomposition artefact presented in the previous section describes the complete set of systems, subsystems, and components in the satellite payload. However, engineers typically focus on their specific views, and their individual view-point-specific contributions contribute to the whole satellite. Lets say there is a communications engineer who is only concerned with the communication subsystem, and would like to only 'see' the 'part' of the satellite associated with the communications viewpoint.
   - For the UI of the communications engineer, the irrelevant subsystems should be abstracted out and only the relevant ones presented.
   - If the communications engineer modifies a component in the communications subsystem (lets say, adds a new "*AntennaComponent*" with ID "*ANT002*"), from the communications view, that change needs to be reflected in the total system architecture.
4. **Versioning**
   The versioning scenario consolidates some and depends on the previously described scenarios:
   The update to '*MainThruster001*'s mass (from 150.5 kg to 165.0 kg in the product catalogue) is a significant design change. This implies that, '*product_catalogue_v1*' effectively becomes '*product_catalogue_v2*'. This also triggers a change in the architecture specification as described in the change impact scenario (scenario 1). This also potentially impacts and necessitates an update to the report (scenario 2). Consequently the view may also need to be updated if the '*MainThruster001*' component for the viewpoint of a hypothetical propulsion engineer (scenario 3).
   - The lineage of different versions of the same artefact ('*product_catalogue_v1*' becomes '*product_catalogue_v2*', '*architecture_specification_v1*' becomes '*architecture_specification_v2*', '*report_v1*' becomes '*report_v2*') should be tracked.

- The reasons for creation of a new version of the dependent artefacts (in this case, the architecture specification and report) should be tracked as well (i.e. why was a new version '*architecture_specification_v2*' and '*report_v2*' created?).

5. **Collaboration**
   Lets say there are 2 communications engineers working concurrently on the satellite design and they concurrently (unaware of each other's changes) make conflicting changes to the satellite communications subsystem. Engineer 1 renames the id of one of the '*Antenna Component*'s from '*ANT001*' to '*ANT003*', whereas Engineer 2 deletes the '*Antenna Component*' with ID '*ANT001*'.
   - The engineers need live collaboration to visualize changes made by each other in real-time (given network latency is low), in such a scenario possibly these conflicts will not occur in the first place (since each participant sees the other's changes before making another change, and changes are no longer concurrent).
   - However, if conflicting (offline, concurrent) changes are made by the engineers, they need to be handled and possible reconciliation strategies should be suggested.

3 Solution Presentation Requirement

MoM challenge responses must discuss their solution with respect to the model management aspects listed below. Note that not all dimensions have to be implemented for a submission to this challenge to be accepted. While we appreciate and favour real demonstrations with tools, it is sufficient for a submission to discuss the features of the approach w.r.t. the proposed dimensions without actually implementing them. Responses with real demonstrations must provide a replication package along with the submission. This can be done via a public Git repository with detailed instructions on how to run the software, or a Docker image.

3.1 High-level Dimensions and Functions of MoM

Main approach used to manage a collection of models (required). Explain the general approach followed to allow the management of the set of models of the challenge. In this sense, responses should place themselves (while justifying the rationale for doing so) into the model management categories defined by ISO 14258 [10]. They are *integration*, *unification*, *federation* [3], or hybrid (approaches combining more than one general approach) solutions. Solutions that do not fall into either category are also welcome, but the responses must explain how their approach differs.

Technological Spaces (required). Explain how the approach deals with heterogeneity in technological spaces [11]. Indeed, the MoM challenge includes operations which rely on information belonging to artefacts represented in different technological spaces (owl, json, csv, markdown, etc.). Note that responses may decide to not deal with heterogeneity and translate all artefacts to the same technological space. If this is the case, responses should also discuss the limitations of their solution.

Relationships (required). Different relationships may exist between artefacts and the information contained in them. Discuss how the challenge response deals with these relationships, including representation, semantics, and management.

Operations (required). Responses must discuss how they deal with operations. This includes consistency constraints, synchronization operations, queries, etc. Dimensions to be discussed include the type of supported operations, formalisms & languages, execution modes (e.g., batch, reactive, lazy), etc. How (if) do they keep track of dependencies between artefacts? How do they allow the specification of queries from different artefacts (possibly multi-formalism) through a unified language?

Views. The interaction (between the user and MoM system) can be directly on the models (when they form a synthetic description of the system) or via views in a projective approach, or a combination of both. Additionally, views may combine different models and display specific or arbitrary filtering or aggregations. Discuss how (if) the challenge response provides view-based visualization and/or operations on models.

Versions and Variants. Responses must discuss how (if) they provide the possibility of tracking different versions or variants of models.

Collaboration. Responses must discuss how (if) they provide the ability for online (live) or offline collaboration (by tracking conflicting versions and providing strategies for reconciliation).

Further Instructions. Additionally, responses to this challenge should highlight limitations (notably, requirements the solution does not address, if any.), list key advantages and drawbacks, describe extensions or adaptations that may have been made to the challenge description, and include a discussion on lessons learned and their implications for future work.

3.2 Scenario-specific attributes

1. **Change Impact and Consistency**
   - How does your tool detect the change in the part catalogue?
   - What mechanisms or processes ensure that the '*mass_kg_per_unit*' for component '*T001*' in the architecture specification is updated to 165.0?
   - Consequently, how does your tool recalculate and update the '*calculated_total_mass_kg*' in the architecture specification?
2. **Unified Querying & Validation**
   - How does your tool automate the check that the '*calculated_total_mass_kg*' from the architecture specification complies with the relevant constraint in the requirements specification?
   - When the '*calculated_total_mass_kg*' in the architecture specification is updated to 360.0 kg, how does your tool update the report to show that '*REQ001*' is now "*NOT SATISFIED*"?
3. **Generation of Views**

  - How are the irrelevant subsystems abstracted out and only the relevant ones presented to the communications engineer, in your tool?
  - If the communications engineer modifies a component in the communications subsystem, from the communications view, is that change reflected in the total system architecture? If so, how does your tool do it?
4. **Versioning**
  - Does your tool track the lineage of different versions of the same artefact? If yes, how does it do so?
  - Does a new version of an artefact trigger a new version of dependent artefacts? Are these dependencies and reasons for creation of a new version of the dependent artefacts tracked in your tool?
5. **Collaboration**
  - First of all, how (if) does your tool support live collaboration between the engineers – do they visualize changes made by the other user in real-time? In such a scenario where engineers work concurrently online, the scope for inconsistencies is lesser.
  - Is your tool able to track different versions of models concurrently by branching (or another strategy)?
  - Does your tool support reconciling differences between conflicting versions, and does it propose possible merge solutions? What are the solutions proposed by your tool in merging the inconsistent versions of Engineer 1 and 2?

4 Conclusion

Responses should cover:

- Requirements the solution does not address, if any.
- Any extensions that may have been made to the case description or evaluation aspects.
- Advantages and drawbacks of the presented solution.
- Advantages and drawbacks of the presented MoM approach that may not be evident in the solution to the challenge but are worth mentioning.
- Lessons learned and their implications for future work.

References

1. *Post Proceedings of the STAF 2023 Workshops*, volume 3620. CEUR-WS.org, Jan. 2024.
2. J. P. A. Almeida, A. Rutle, M. Wimmer, and T. Kühne. The multi process challenge. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 164–167, 2019.
3. M. Amrani, R. Mittal, M. Goulão, V. Amaral, S. Guérin, S. Martínez, D. Blouin, A. Bhobe, and Y. Hallak. A survey of federative approaches for model management in mbse. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, pages 990–999, 2024.

4. J.-C. Bach, A. Beugnard, J. Champeau, F. Dagnat, S. Guérin, and S. Martínez. 10 years of model federation with openflexo: Challenges and lessons learned. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, MODELS '24, page 25–36, New York, NY, USA, 2024. Association for Computing Machinery.
5. J. Bézivin, F. Jouault, and P. Valduriez. On the need for megamodels. In *proceedings of the OOPSLA/GPCE: best practices for model-driven software development workshop, 19th Annual ACM conference on object-oriented programming, systems, languages, and applications*, pages 1–9. Citeseer, 2004.
6. J. Cederbladh, D. Krems, and A. Cicchetti. Extending magicgrid to support virtual prototyping for early system performance validation and verification. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, MODELS Companion '24, page 287–298, New York, NY, USA, 2024. Association for Computing Machinery.
7. M. Elaasar, N. Rouquette, D. Wagner, B. J. Oakes, A. Hamou-Lhadj, and M. Hamdaqa. opencaesar: Balancing agility and rigor in model-based systems engineering. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 221–230, 2023.
8. Y. Hallak, D. Blouin, L. Pautet, L. Saab, B. Laborie, and R. Mittal. Model management at renault virtual simulation team: State of practice, challenges and research directions. MODELS Companion '24, page 1005–1014, New York, NY, USA, 2024. Association for Computing Machinery.
9. International Council on Systems Engineering (INCOSE). Systems engineering vision 2035: Engineering solutions for a better world. Technical report, INCOSE, San Diego, CA, 2022.
10. International Organization for Standardization. Iso:14258:2014: Industrial Automation Systems: Concepts and Rules for Enterprise Models, 2014.
11. I. Ivanov, J. Bézivin, and M. Aksit. Technological spaces: An initial appraisal. In *4th International Symposium on Distributed Objects and Applications, DOA 2002*, 2002.
12. R. Jongeling, A. Cicchetti, F. Ciccozzi, and J. Carlson. Towards boosting the openmbee platform with model-code consistency. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, MODELS '20, New York, NY, USA, 2020. Association for Computing Machinery.
13. R. Mittal, D. Blouin, A. Bhobe, and S. Bandyopadhyay. Solving the instance model-view update problem in aadl. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, MODELS '22, page 55–65, New York, NY, USA, 2022. Association for Computing Machinery.
14. P. J. Mosterman and H. Vangheluwe. Computer automated multi-paradigm modeling: An introduction. *SIMULATION*, 80(9):433–450, 2004.
15. No Magic, Inc. Magicdraw, 2024. Version 2024x R1.
16. B. Tekinerdogan, R. Mittal, R. Al-Ali, M. Iacono, E. Navarro-López, S. Bandyopadhyay, K. Vanherpen, A. Barišić, and K. Taveter. Chapter 3 - a feature-based ontology for cyber-physical systems. In B. Tekinerdogan, D. Blouin, H. Vangheluwe, M. Goulão, P. Carreira, and V. Amaral, editors, *Multi-Paradigm Modelling Approaches for Cyber-Physical Systems*, pages 45–65. Academic Press, 2021.
17. T. Weber, M. Ojha, M. Sadeghi, L. König, M. Armbruster, A. Lange, E. Burger, and C. Atkinson. Towards deep reactions in multi-level, multi-view modeling. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, MODELS Companion '24, page 760–769, New York, NY, USA, 2024. Association for Computing Machinery.