10 years of Model Federation with Openflexo: Challenges and Lessons Learned

Jean-Christophe Bach, Antoine Beugnard, Joël Champeau, Fabien Dagnat, Sylvain Guérin, Salvador Martínez

10 years of Model Federation with Openflexo: Challenges and Lessons Learned

Jean-Christophe Bach
jc.bach@imt-atlantique.fr
IMT Atlantique
Lab-STICC, UMR 6285
Brest, France

Antoine Beugnard
antoine.beugnard@imt-atlantique.fr
IMT Atlantique
Lab-STICC, UMR 6285
Brest, France

Joël Champeau
joel.champeau@ensta-bretagne.fr
ENSTA Bretagne
Lab-STICC, UMR 6285
Brest, France

Fabien Dagnat
fabien.dagnat@imt-atlantique.fr
IMT Atlantique
Lab-STICC, UMR 6285
Brest, France

Sylvain Guérin
sylvain.guerin@imt-atlantique.fr
IMT Atlantique
Lab-STICC, UMR 6285
Brest, France

Salvador Martínez
salvador.martinez@imt-atlantique.fr
IMT Atlantique
Lab-STICC, UMR 6285
Brest, France

ABSTRACT

In the context of complex system development, heterogeneous modeling responds to the need to integrate several domains. This need requires the use of the most appropriate formalism and tooling for each domain to be efficient. Model federation promotes the semantic interoperability of heterogeneous models by providing the means to reify correspondences between different model elements, add custom behaviors and bridge the gap between technological spaces. As such, it can be used as an infrastructure to address many different software engineering problems. We have been doing so for over a decade in a tight collaboration between a small software engineering startup and academia. This paper reports on this experience.

Concretely, we discuss the context, ambitions, and challenges that led to the inception of our practice of model federation, and we present five use cases experiences, stemming from real industrial and academic needs, and elaborate on lessons learned. In addition, we also report on challenges and lessons learned regarding the development and maintenance of a model-driven model federation tool, the Openflexo framework. Finally, we set up a road map for the future of model federation and Openflexo.

CCS CONCEPTS

• **Software and its engineering → Model-driven software engineering**; **Abstraction, modeling and modularity**; **Interoperability**.

KEYWORDS

Model federation, Model management, Experience report

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Jean-Christophe Bach, Antoine Beugnard, Joël Champeau, Fabien Dagnat, Sylvain Guérin, and Salvador Martínez. 2024. 10 years of Model Federation with Openflexo: Challenges and Lessons Learned. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

It has long been acknowledged that the description, and then the development, of complex systems often requires the integration of different points of view [32]. These different points of view may correspond to different aspects of the system (*e.g.*, structure, behavior, cost) and/or to different *domains* (mechanics, electronics, chemistry, computer science, etc.). In both cases, effective mechanisms must be in place so that the different views can be developed by the corresponding experts using the most appropriate tools while their work contributes to the description and development of the system as a whole.

Multi-view modeling tries to provide an answer to this problem [2, 13] by proposing different approaches and mechanisms for the creation and management of systems, together with their views and correspondences (*i.e.*, links between elements in different views). However, when views are heterogeneous (in the sense of views in different technological spaces [37]), additional mechanisms are required to bridge the gap between the view system (containing the correspondences) and its multiple views. In this regard, in [32], model federation is identified as a solution to this problem. From the multi-view modeling perspective, *model federation* is a synthetic approach (*e.g.*, the system is built from the views) with explicit (reified) correspondences (see [12] and [29] for other approaches in this category). Additionally, it promotes the *autonomy* of heterogeneous views, the federated models for us, which remain in their technological space and follow their own processes (connectors are used to access these views).

We advocate *model federation* provides the means to design complex systems which involve different stakeholders, including non-IT ones, while preserving the practices and processes of a given domain or enterprise. In this sense, this paper presents the experiences of applying the model federation approach to deal with complex multi-view use cases, both in industry and academia, for more than

a decade. Concretely, we elaborate on the challenges we have faced all along this adventure, give details on five representative use cases and discuss lessons learned. Our practice of model federation is tightly coupled with the development of the Openflexo framework, an MDE tool. Therefore, we also report experiences, challenges, and lessons learned regarding the development and maintenance of this tool.

This paper follows a tradition of experience reports on modeling in real and industrial scenarios [6, 46, 33, 14]. It does so from the perspective of a small startup and academic partners trying to apply model federation, a lesser-known MDE approach, in the wild. In this respect, we are convinced that this is a contribution of interest for the MDE community. The rest of the paper is organized as follows. Section 2 offers an historical context including the challenges we faced. The basic principles of model federation are introduced in Section 3 followed by a detailed description of selected use cases in Section 4. Lessons learned are explored in Section 5. We end the paper by discussing related work in Section 6 and presenting conclusions and future challenges in Section 7.

2 CONTEXT & CHALLENGES

The objective of this section is twofold. First, we briefly introduce the historical background that led to the inception of our practice of model federation, the development of the Openflexo framework as a supporting tool, and its associated company. Second, we present the high-level challenges we have faced.

2.1 A bit of history

"Bridging the gap between business and IT" was the motto of the original company[1] where the Openflexo project took its roots. The original aim was to make IT and digital tools more accessible to as many people as possible, in other words to help bridge the gap between various business areas and IT. Capturing business expertise is at the heart of these concerns and is based on the increased abstraction and use of models, by and for non-specialists. Our starting point was the proposition that any source of information can be considered as a model [8], leaving the various players free to choose the paradigms, data, and tools best suited to their vision, skills and culture. From this context emerges the need for coherent management of multiple models (or information sources interpreted as models), each with its own autonomy and life cycle, which we call semantic interoperability. This raises many major scientific issues (such as interpretation, heterogeneity, and dynamicity).

We started from the idea that it would be useful for scientific research to feed into the development of a commercial product, and conversely for industrial use cases to feed into research. Openflexo SCIC company was built around this association between academic researchers, professional developers, and industrial customers. From the company's point of view, the activity was concentrated around a technical and commercial response to very concrete business needs, while the researchers worked on building abstractions to solve the "meta-problem" that lays behind each problem encountered. The result is an original software infrastructure, based both on pragmatic solutions and very generic abstractions. It also offers a very clear separation between technical issues (intrinsically shared) and business issues (intellectual property that needs to be protected). Openflexo SCIC company has since been wound up.

[1] Openflexo SCIC

2.2 Challenges

In the following, we elaborate on the challenges we faced by applying the model federation approach to various use cases where semantics interoperability of heterogeneous information sources was required. Other authors identify challenges in a multi-modeling context. For instance, T. Denton et al. [19] identify three challenges for their multi-modeling platform: "Capturing Multi-Model Interdependencies", "Maintaining Multi-Model Consistency", and "Semantic Precision of Inter-model Data Exchange". Their challenges focus on technological aspects and ignore process or methodological issues. We share technical challenges, even if we organize them a bit differently, but our challenges take more explicitly into account the autonomy and diversity of modeling processes.

Challenge 1: preserving existing practices

The first challenge stems from the idea that it is not up to users to adapt to tools, but rather the opposite. Existing practices around a given concern often result from a consensus within the community dealing with this aspect. These concerns are embodied in specific data, paradigms, representations, vocabularies, processes and tools.

Under the pretext that we want to bring together different concerns and professions, we must not break or disrupt what already exists and works. The challenge here is to be able to manage the semantic interoperability of these various business worlds while *preserving all existing practices*. This implies being able to manage a certain amount of heterogeneity. The heterogeneity we are talking about here is multiple. First, it is technical and syntactic: semantic interoperability obviously requires the ability to connect diverse technologies, which pertain to semi-structured or unstructured paradigms (with no metamodel). It is also conceptual and semantic, since it involves linking different conceptual and/or technical universes (often constructed in various contexts and with different intentions). From the point of view of pragmatics and processes, heterogeneity is finally linked to different domain-specific businesses, such as engineering departments that do not work in the same way and do not necessarily understand each other. Preserving the independence of existing things means linking things that were not meant to be together, including in terms of life cycle, while keeping them autonomous. We also need to be able to adapt to the flexibility of existing practices without interfering with them.

Challenge 2: seeing everything as a model

Today, there is a clear consensus around the use of modeling. Modeling enables an increase in abstraction, making it possible to use, analyze, and design complex software and systems, by reducing their complexity and bringing it within reach of human cognition. This approach is sometimes explicit, as shown by the model-driven approaches in certain engineering fields. Other communities manipulate models in less explicit ways, using, for example, spreadsheets, drawings, or slides. More generally, experts organize themselves by domain and use the *most appropriate formalism* [51] and tools [38] for the task in hand. They do not necessarily give importance to the existence of metamodels or to conformance issues.

Our second challenge is to propose that any source of information (called resources below) should be considered as a model. In

the world of MOF-like models or metamodels, there is generally no ambiguity related to model interpretation[2]. But when modeling takes place in a less formalized framework, the semantics can be implicit. Sometimes, it is the tool used to create and manipulate resources that provides their semantics. In such a case, interpreting these resources from outside the tooling poses problems (it may need retro-engineering). Notice that non-standard usage of "notations" also happens in practical engineering. Consequently, we need a way to explicitly define the semantics we want for such external resources. Furthermore, this semantic should be defined in a contextual way because several uses of the same resources can require different interpretations.

Challenge 3: conceptualizing and organizing

In the context of heterogeneous modeling, we face several issues when keeping the various models independent and maintaining them in their technological space (as presented in the challenge 1).

First, the elements of the various models need to be identified and referenced when defining the links between them. This must be true even for elements of different technological spaces. For instance, elements from EMF models are not designated and accessed in the same manner as cells in a spreadsheet or variables in an Event-B specification. Heterogeneous modeling therefore requires to consider the targeted technologies when establishing correspondences between model elements. But this must be done in a uniform and neutral way when creating the links.

Then, we need a modeling space to contain these links. This modeling space must remain both technologically and semantically neutral with respect to the heterogeneous models (as detailed in challenge 2). Furthermore, this space may contain a large network of links between the elements of the various models. Organizing this space to manage this potentially large number of links becomes crucial. This modeling space must therefore offer abstraction, modularity and classification mechanisms. Lastly, the elements reifing the links, called correspondences, vary in nature. They range from concrete values to pure abstractions, and are frequently hybrid entities (part concrete and part abstract, typically *clabjects* [1]).

These various aspects highlight the essential role of a conceptual space where the links between the heterogeneous models are reified. This conceptual space needs structuring constructs and ways to link elements of the heterogeneous models. The strategic issues surrounding this conceptual space are significant, given the time-consuming nature of modeling activities. Therefore, maximizing reusability becomes imperative, with a focus on modularity and genericity to facilitate on-demand specialization. Notice that here, reusability concerns both the elements of the conceptual space and the technical elements in charge of connecting to the various technological spaces of the heterogeneous models.

Challenge 4: mastering evolution

Working with multiple models requires to focus not only on structural features, but also on behavioral features. As mentioned above, preserving existing practices is critical and implies the preservation of the life cycle of these models. First, it is crucial to be able to detect (and potentially react to) the evolution of a model element. Second, unless we are in a read-only context where we only support analysis such as inconsistencies detection, many use cases require interacting with the model elements and modifying them. Some other use cases (such as model synchronization) would require explicit element modifications triggered from the correspondences (within the conceptual space). Mastering coherence means supporting several strategies to deal with inconsistencies: from detecting and ignoring them (for a shorter or longer period of time), to proposing automatic propagation of inconsistency repairs, to including semi-automatic inconsistency resolution, thanks to human and interactive decisions.

[2]Except when ambiguity is a feature (*e.g.* "semantic variation points" of UML).

3 THE MODEL FEDERATION APPROACH

The aim of this section is to provide a quick overview of the model federation approach, including the language and framework used to support the use cases presented in the section 4.

Model federation, as defined in ISO-14258 [31], introduces principles that complement the integration and unification approaches to interoperability. The purpose of this standard is to characterize possible strategies to define correspondences between viewpoints, or languages associated with different engineering concerns. In this standard, the federation approach keeps the models (and their metamodels) unchanged and independent. The semantic equivalence is encoded within a new element, the federation, that holds the correspondences between the model elements. The federation approach is summarized by figure 1. A user may keep its usual practice and tools (right part of the figure), while an additional autonomous model, the federation model is introduced to encode the dependencies and their semantics. The federation model can offer new additional services (left part of the figure).

In our approach to model federation, we rely on a domain-specific language, FML (Federation Modeling Language) to define federation models. This language supports the definition of both the structure of conceptual models and their behaviors. It has an object-oriented flavor. Our Openflexo tool is then able to execute federations. The central element of the structural part of FML is the *FlexoConcept*, which enables the reification of a business concept. A *FlexoConcept* defines both structural properties and behaviors. *FlexoConcept*s are structured within *VirtualModels* that contain them and are organized according to multiple inheritance semantics.

Structural functionalities, also known as properties or *FlexoProperty*, are used to define or reference data. These properties may be internal to the federation or external (*i.e.*, refer to one of the federated models). A specialized kind of property, *FlexoRole* is used to implement these latter external dependencies. A link to an external data is made through a *ModelSlot* (see figure 1). The definition of *ModelSlot* is based on a *Technological Adaptor*, which provides an API dedicated to interaction with the technological space of the target element. The behavior of a *FlexoConcept* is defined by executable methods called *FlexoBehaviour*. Such behaviors encode the semantics of the concept such as their creation and modification but also more specific business behaviors that form the basis of the federation services.

As an example, a *VirtualModel* System could contain a *FlexoConcept* Component. This concept could gather information about a component from three federated models: its configuration described in a spreadsheet, its internal structure described within a SysML *Block Definition Diagram* and its detailed description from

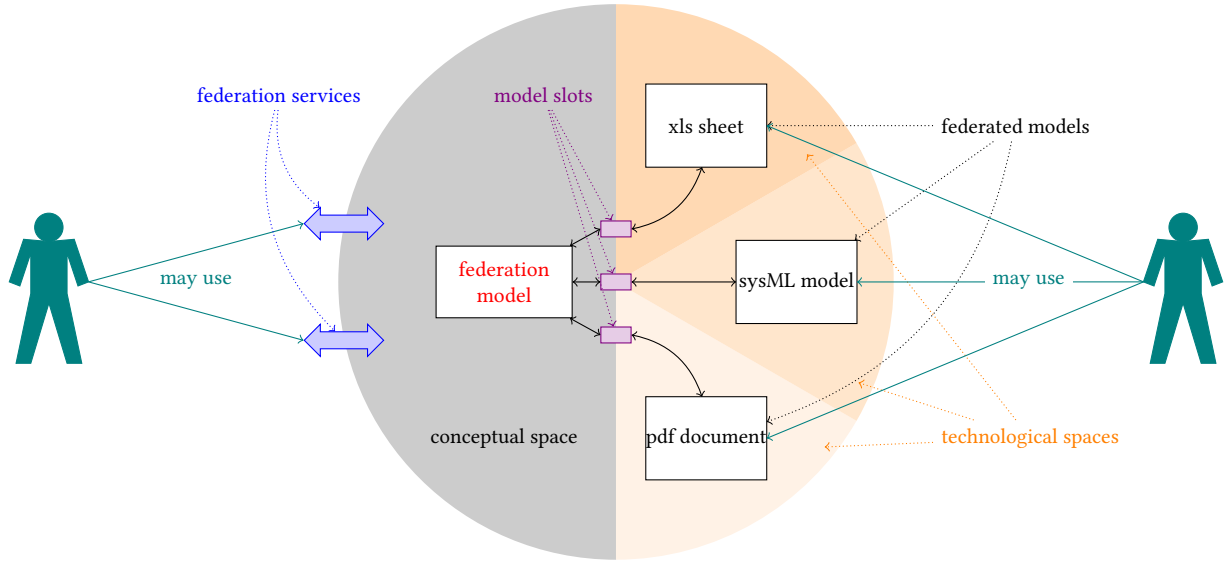

Figure 1: The principle of federation

a PDF document. Three *FlexoRole*s would use three *ModelSlot*s to interact with these external elements. The System could have a behavior to create a new instance of the concept Component for each instance found in the federated architecture model (expressed in a SysML diagram). A Component could have a behavior to support the modification of one of its configuration parameter by updating both the configuration spreadsheet and the documentation with the PDF document. Similarly, a component can offer users the service of synthesizing information about its behavior and initial configuration. It may also trigger the associated behavior to update the Component's configuration or maintain it unchanged (as shown in the left part of the figure, the arrows represent the connections between the users and the federation model).

The FML language is integrated in the Openflexo framework[3], which provides a language IDE (editors, debugger, etc.), a language interpreter and a library of technological adapters that allows addressing the technological spaces of the Use Cases exposed in the next section.

*ModelSlot*s and libraries of *Technological Adaptor*s provide some kind of proxy, that realize a bidirectional (read/write) connection to an external source, giving it a contextual semantic/interpretation. This helps to preserve usual practices (challenge 1), but, moreover, allows giving to external sources a contextual semantic/interpretation that increases "simple" unstructured or semi-structured source of information to the status of the model (challenge 2).

*VirtualModel*s, *FlexoConcept*, *FlexoBehavior*, and *FlexoRole*s provide constructs for organizing and reusing model federations. Models themselves are reified thanks to *VirtualModel* and links (correspondences) thanks to *FlexoRole*. When reified, both can have behaviors and be reused. These constructs in FML help to improve the organization of model space and its semantics (challenge 3).

Evolution and consistency (challenge 4) are highly dependent on organization, but the key is the linking mechanism that supports crossing technological boundaries. Correspondences defined between model elements may evolve over time and, since processes are independent, lead to broken links. However, since the links are reified, they can be handled as usual data and can be updated by specific behaviors. Finally, being interpreted, FML can perform on-the-fly interpretation and smoothly handle broken links.

4 SELECTED USE CASES

We present here a selection of 5 use cases from all the projects carried out over the last 10 years. This selection results from the will to cover all challenges, several application domains and several contexts of realization. Figure 2 summarize how these use cases cover the challenges of section 2.2. Some other projects are published; the reader could be interested in a multi-level modeling challenge [28], security modeling in MBSE [11], or process modeling [25] for instance.

4.1 Formose

Formose was a research project[4] aiming to propose a formally grounded, model-based requirements engineering (RE) method for critical systems. The main partners were ClearSy, LACL, Institut Mines-Telecom, Openflexo, and Thales. Our role in the project was to provide the open source tool core of the proposed RE method [23]. This tool, Formod, relies on the model federation paradigm to connect the various elements needed during a RE process.

More precisely, Formod federates (1) documents (*e.g.* Word file) providing the informal requirements, (2) a domain model describing the essential concepts for the system under study using a graphical DSL based on ontologies [52], (3) a goal model (SysMLKaos [44]) structuring the requirements in a tree refining abstract goals into concrete requirements affected to elements of the system and (4)

[3]https://openflexo.org/, https://github.com/openflexo-team/

[4]https://formose.lacl.fr/

	Preserving existing practices	Seeing everything as a model	Conceptualizing and organizing	Mastering evolution
Formose	✓	✓	✓	✓
SSE4Space	✓	✓	✓	
FME		✓	✓	
Model mapping	✓			✓
PDF labels		✓	✓	✓

Figure 2: Uses cases challenge covering matrix

formal descriptions of these requirements in Event-B together with their formal refinements.

Formose was a complex project that faced all the challenges described in section 2. More precisely:

- *Challenge 1:* Eliciting requirements, analyzing a domain, refining and allocating requirement to subsystems and conducting proof are four very different engineering done by specialists. For example, within Formod, the operation on B code was done from within the Atelier B, a mature prover for Event-B. The federated B code could evolve in its technological space and the federation takes in charge the evolution management in the related goal model.
- *Challenge 2:* The elicitation and justification parts of the method use both the document view that interprets a text document as a model. Requirements can be linked to parts of the document (*e.g.* text, figure). For this, a document is interpreted as a model containing linkable elements. The B code is also viewed as a model and connected to the elements of the goal model.
- *Challenge 3:* A requirement is a multi-faceted element with information coming from the informal documents but also from the goal and domain models and the B code. A central concept of requirement federates all these concrete elements. Furthermore, as the proposed method is iterative and incremental, the conceptual space was highly structured, based on several virtual models, to support various concrete practice.
- *Challenge 4:* The model federation at the core of Formod was in charge of maintaining consistency. For example, an evolution made to B code is propagated to the goal models. Similarly, decomposing abstract goal was done using SysML-LKaos but was directly reflected on the B code. Similarly, modification done in Word to a requirement document is kept in consistence with the model of requirements.

The main result of the project is the Formod tool[5]. It federates the previously described elements and provides four business views of figure 3 to build a central requirements model:

(1) A *document view* where the user elicits and justify requirements.
(2) A *goal view* where users structure and refine requirements, allocate them to agents.
(3) A *domain view* gathering knowledge about the future operational context of the system under study.

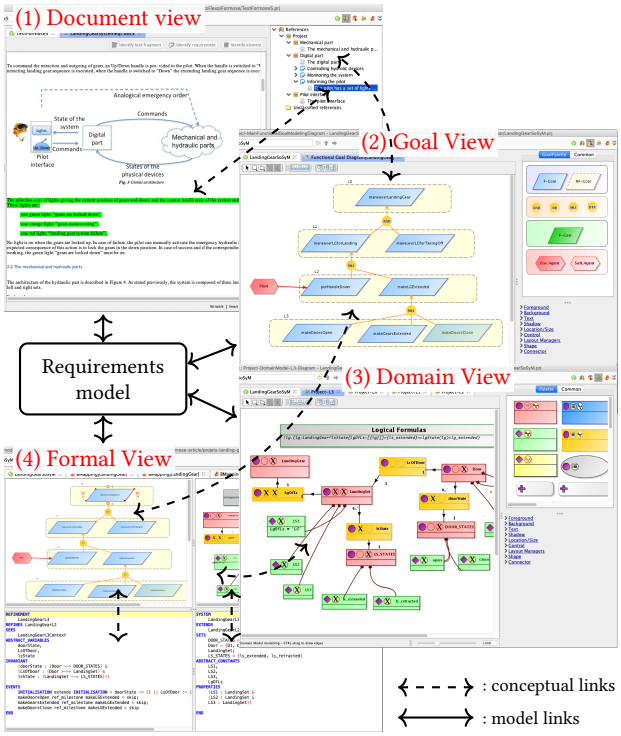

Figure 3: Formod: four views for a requirements model

(4) A *proof view* where the overall refinement level of models is translated into Event-B and the proofs can be conducted.

Formod is implemented as a model federation containing 19 virtual models defining in total 177 concepts for around 8000 lines of FML. It relies on height technology adapters, one for each dedicated tooling and its language or data source. The complete metamodel can be browsed as it is available in the tool[6].

4.2 SSE4Space

The goal of the SSE4Space (Secure Systems Engineering for Space) project was to develop a complex project management framework

[5]https://github.com/openflexo-team/formod

[6]The `.fml` files of https://github.com/openflexo-team/formod/tree/master/formod-rc/src/main/resources.

for secure systems engineering. We brought our expertise in software engineering, architecture, and modeling to Telindus, a company with expertise both in cybersecurity and in tooling that integrates security aspects into project management. With our assistance, Telindus provided the SSE4Space tool built upon the Openflexo framework for the European Space Agency (ESA).

In the space industry, projects are usually long (more than 10 years) and complex due to the nature of the products (costly Cyber-Physical Systems in constrained environments). Thus, they require a strict and careful management. The tool helps to monitor a project and to better control and integrate cybersecurity risks. It provides global traceability to support auditing and certification. To achieve this, it federates several tools and technologies. The core of the tool uses Openflexo to orchestrate information from multiple sources and enable specific processing within models. More precisely, the federation model integrates workflow models, risk analysis models, and requirement models.

SSE4Space met three out of the four challenges of section 2:

- *Challenge 1:* The preservation of existing practices and tools was at the core of this multidisciplinary project: the framework had to support project management by integrating expertise in system engineering, risk analysis, cybersecurity (*e.g.* pentesting). For example, risk analysts use the Secure Engineering Support Tool [7], a risk assessment tool that uses the MEHARI methodology[8]. System engineers can use SysML which is supported by various tools such as Papyrus, Rational Rhapsody, MagicDraw... Pentesters use commercial scanners and custom scripts in addition to manual methods to produce their analyzes. All these business domains had to be integrated into the project workflow, each task producing artifacts and tasks feeding it.
- *Challenge 2:* Models are built from heterogeneous sources (user interactions, files, or data extracted from tools). The workflow, which is defined in terms of precedence constraints, orchestrates the tasks and the artifacts. They are fed by elements of the risk model, which in turn are fed by security requirements or requirements that have an impact on security. For example, updating a vulnerability database can trigger a risk assessment task. Risk analysts may request a vulnerability scan or a specific check from a pentester to feed their threat model and their vulnerability model, resulting in a pentesting task whose result will feed into the risk assessment task.
- *Challenge 3:* Each task and each artifact (pentest analysis, risk analysis, system constraint, etc.) can be used as input to the workflow, leading to its recalculation. Therefore, there was a need to structure the concepts of risks and requirements so that each element could be integrated into the workflow, leading to three main models: workflow, risk analysis, and requirements.

The main outcome of the project is the SSE4Space tool developed by Telindus for ESA. According to ESA policy, it will eventually be published as an open-source tool[9]. This work has been published at an industrial conference of the space domain [39]. Through fruitful exchanges with the Telindus engineers, this project also enabled us to push Openflexo's development towards the next major version. The published version 2.99 introduced the textual syntax FML and its associated toolset.

[7]SEST: https://github.com/mmerialdo/Secure-Engineering-Support-Tool
[8]MEHARI: http://meharipedia.org/home/
[9]https://gitlab.space-codev.org/sse4s/sse4space-proximus

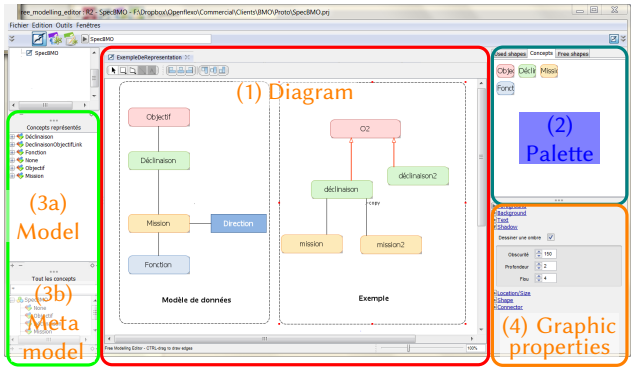

Figure 4: Free Modeling Editor.

4.3 Free Modeling Editor

The Free Modeling Editor (FME) is an internal research project dedicated to understanding the modeling process [7]. The starting point of this approach is based on the observation that, in the context of DSL development, identifying the metamodel is not always straightforward. In this context, the aim is to develop an approach and a tool for graphically editing models and their metamodels concurrently. The metamodel does not take precedence over the models, although it remains an option. Thus, models and metamodels can be co-constructed [24]. This tool has been used in many projects to help define domain-specific metamodels: local administration organization [7] or emergency department [20] for instance. Note that the co-construction of model and metamodels is also explored in other flexible modeling approaches [53, 26].

In the FME tool, instances or model elements may exist without a definition of the corresponding concept. In such cases, these instances exist without any formal definition, but can be utilized to define new abstractions or concepts. Alternatively, these instances can be identified as instances of an existing concept after their creation. Concepts can also be created directly through a generalization process. Classically in this case, a concept can be used to create new instances associated with the graphical representation of the concept.

Figure 4 shows a screenshot of the FME editor. It comprises four main regions: (1) A modeling view where concrete instances (model elements) or concepts are drawn. This region is not tied to a dedicated abstraction level and depends only on the modeling needs. (2) This view contains several tabs, including a tab for the palette of concepts (metamodel) and a tab for defining shapes to create instances. (3) A list of instances classified by concepts (or without a particular definition) is displayed in the view (3a), along with a list of concepts created (metamodel definition) (3b). (4) A view dedicated to the customization of graphical properties.

The Free Modeling Editor met two out of the four challenges previously identified:

- *Challenge 2:* FME allows us to explore the relationship between an abstraction and its concretizations. We consider that any modeling element can be abstracted, concretized, or even duplicated at the same level of abstraction. In this manner, we remove any reference to the modeling level while associating one or more graphical representations with the elements. In many projects, we used the Free Modeling Editor, coupled with the ability to address external tooling, to launch reflection from examples, sometimes provided in a PowerPoint document [24], sometimes from textual description [20]. In doing so, we have successfully defined metamodels for DSML, supported by a prototype tool built simultaneously.
- *Challenge 3:* The Free Modeling Editor is a conceptual space editor. In that sense, it provides the capacity to create concepts and their instances in the conceptual space independently of any federated models.

FME is a pure Java application embedded in Openflexo, and distributed as a desktop application on the Openflexo website[10]. It appears it is a useful environment for testing modeling activities.

4.4 Model Mapping

The Openflexo SCIC company was commissioned by a major defense contractor to design a tool for mapping business DSMLs used in very different contexts. An important aspect of this contract was the very high level of sensitivity of the domains involved, with classified data, in particular. The tool developers did not have access to the models. Furthermore, on the customer's side, very few experts had knowledge of the two business areas to be linked. The business rules relating to the mapping were not directly accessible to the developers but only to the tool's expert users, which meant that the Openflexo developer's team had to work at a high level of abstraction to provide the tool.

To overcome the problem of data sensitivity, Openflexo worked on and delivered a version of the tool based on desensitized metamodels (modeling a city using EMF technology). The "CityMapping" prototype was built to demonstrate the model mapping facilities over heterogeneous EMF models[11]. Figure 5 shows an example of such a model mapping metamodeling. The two DSMLs appear on either side of the figure. The two metamodels overlap partially. The correspondences between the business concepts common to each of the two DSMLs are metamodeled using the FML language. These correspondences are based on n-ary links that connect entities of different granularity (*e.g.* EMF class or EMF attribute). `House` type is implemented, for example, with a `HouseType` attribute in the left metamodel (`Model1`) while it is implemented by a specialization in the right metamodel (`Model2`).

Model mapping activity faced all the challenges previously identified, but mainly focuses on managing consistency while preserving existing engineering practices (challenges 1 and 4).

[10]https://openflexo.org/downloads

[11]Details on this prototype can be found in https://openflexo.org/docs/Tutorials/Tutorial7-WorkingOnModelMapping.

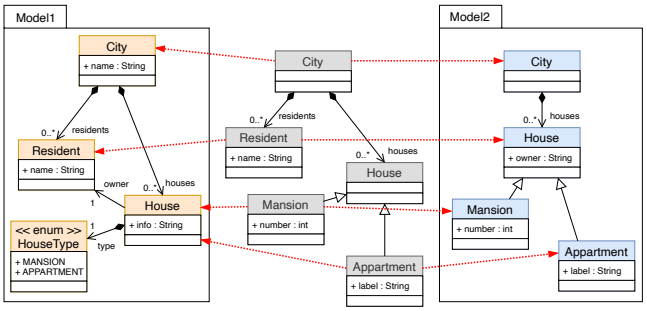

Figure 5: Model federation for model mapping

- *Challenge 1:* This preservation is at the heart of this modeling activity, in particular because the experts in each domain do not have prior knowledge of the other domain to be linked.
- *Challenge 4:* After an initial phase of model alignment and elimination of any inconsistencies that need to be resolved manually, the aim is to be able to maintain the mapping over time, as the two federated models evolve.

4.5 PDF labels

This last use case has been selected to illustrate a very unusual scenario, but reflects real use cases that can be encountered in industry. This scenario arises in contexts where the architecture of the information system is poor or when the nominal processes have not been followed.

The Openflexo SCIC company was requested by a chemical company to help audit and update part of its information system, which contained a number of inconsistencies. The company was a subsidiary of an international group that managed a large number of chemical products in different countries under different customer references. The task was to synchronize the internal product database with the references to the products marketed. The particularity of this project was that the only source of information for the product marketed was contained in the PDF product labels. This involved several hundred references, managed by numerous independent graphic designers. Additionally, the process had to ensure that the labels display the safety recommendations on the products comply with local regulations, depending on the composition of the product described in the internal product database. This required a large number of highly specific business rules to be implemented for checking and updating various sources of information, including SQL databases, Excel spreadsheets, and PDF files.

Openflexo SCIC designed and delivered a specific tool that encoded these various business rules. The tool had to be intuitive enough to be used by the company's employees and interactive enough to explicitly manage inconsistencies and propose problem resolution. The delivered solution was based on a model federation infrastructure described in FML and executed in Openflexo.

This use case met three out of the four challenges :

- *Challenge 2:* PDF files are especially considered a primary source of information (such as safety recommendations, product composition, market, usage, etc.).
- *Challenge 3:* The conceptual space allows us to define the business concepts involved in various data sources, such as the variations of a generic product in different countries, with the associated safety rules.
- *Challenge 4:* Specific business rules are defined at conceptual space level, allowing for the detection and resolution of inconsistencies.

5 LESSONS LEARNED

The previous section presented five selected projects relying on our approach of model federation. These use cases were chosen to illustrate the challenges we faced and resolved. They also concretize the model federation approach. This section outlines six selected lessons learned over the past ten years. The first two are usage oriented, the next two are related to federation and the last two touch on modeling.

5.1 Separate business from technique

This is probably one of the most important lessons: when modeling, focus on business concepts, forget your own modeling habits, especially those depending on technical choices.

We naturally experienced this when using the free modeling editor (use case 4.3). Elaborating the metamodel while modeling instances gives users the freedom to focus on their business concepts. Additionally, we often encounter situations where users describe entities that are mid-abstract, mid-concrete, introducing multi-level modeling aspects as if it were a natural modeling paradigm.

This requires abstraction abilities, to focus on the essence of things and to forget technical details or constraints. This lesson directly meets challenge 1. Technical aspects live in their space, with experts and their own models and habits; federation lives in a tailored model with specific semantics.

The explicit separation between federated modeling spaces and the conceptual space of the federation was initially driven by technological considerations, such as interfacing with various domain tool APIs. This separation has evolved beyond its technical origins and allows the interpretation of a domain model with different semantics defining in the conceptual spaces, depending on the required point of view. This capability relies on a local interpretation at the boundary during integration into the conceptual space (using technological adapters for us) and considers different models within the conceptual space in relation to the boundary (using *ModelSlot*s in our case).

It enables the explicit separation of domain models, sometimes subject to intellectual property constraints or different life cycles, from their interpretation in the federation.

5.2 Tool maintenance & dissemination

This is probably the most worrying lesson: tool maintenance in a small research team is time consuming. We have observed numerous research tools disappear and promising ideas abandoned as a result. Indeed, with a very small team of developers, the available manpower rarely allows for community management, answering questions or updating tutorials when versions change, which hinders the adoption of the tool.

Aware of this issue, the Openflexo company has been created with a two-fold goal: first, to bridge the gap between industrial companies - driven by profit - and academic institutions - driven by recognition and reputation - and second, to organize a community. This period helped to gather a small community composed of model users and developers. In fact, it was during this period that the main architectural refactoring was carried out. In addition, this is also the period where the continuous integration and continuous development processes were installed. We used to generate different tools on demand for the three most popular OS. Alas, with time and OS evolution, we failed in maintaining this service after Openflexo SCIC wound up.

We face other technical evolution. The fast evolution of Java versions is also challenging. Openflexo still relies on Java 8. Java modules constraints, among other novelties, make the evolution to Java 11 complicated. We still encounter a few issues. As a matter of fact, the fast evolution of technology hinders the development of long-running research tools.

When developing Openflexo, we chose a reflexive and interpreted approach. Indeed, the framework relies on the interpretation of a set of models which may change their behaviors. This brought a lot of flexibility, but made it difficult to join the Eclipse ecosystem that relies heavily on code generation. As a consequence, we did not benefit from the visibility Eclipse ecosystem provides. The control of the development chain and the flexibility of interpretation (versus code generation) seem critical to us.

Recently, Openflexo has been adopted by a Luxembourg company for developing an application for the European Space Agency (use case 2). This was the first autonomous development with Openflexo, outside our team. On this occasion, we have observed that it was possible to become autonomous using Openflexo, and that we were able to provide a reasonable support.

5.3 Need for many concrete syntax

Historically, Openflexo was designed for non-software users and was focusing on capturing business concepts from various data and tools, which come with their own representations and notations. Its preferred interface was graphical with numerous windows, check boxes, menus, and so on. Behaviors (method body) were themselves defined with dialog boxes. This was ideal for small size projects. When projects became larger, or behavior became more sophisticated, graphical interface hindered development, and textual interface, with a concrete syntax was expected. Other circumstances revealed other needs, such as tabular entries when it comes to entering long lists of class instances.

The right tool at the right place is not only a matter of users, but also depends on the development phase of a project. In the early stages, the same user may need a graphical view to sketch up ideas, then a textual language to scale up, but again a graphical view to build an abstract view of too large a text, before to identify and rectify a small error. One needs the right concrete syntax at the right time. This requires a language authorizing various concrete syntax that need to be synchronized. Recent work [16, 45] links this need to *Blended Modeling* or *Hybrid Programming*.

5.4 Federation requires organization

Federation models lead to the creation of new models containing potentially many entities that make correspondences between other model entities. To do this, the entities must be designated (referred to). The designation must be flexible enough to function when entities change or move, or at least to detect designation breaks, without preventing the federation from functioning. In the Openflexo framework, this role is played by the technological adapters.

Beyond the designation requirement, the multiplicity of entities and correspondences be managed in the conceptual space show the need for organizational mechanisms. Here, Openflexo reuses standard techniques since conceptual spaces are real models which are organizational units. These can be composed, or can inherit from each other.

For instance, in a use case such as Formose (described in section 4.1), there are thirteen virtual models, linking six pre-existing models allowing making various tools and methodologies interacting with each other (SysML-Kaos, OWL, B, Word, Excel). Managing the complexity of this federation requires proper organization of the conceptual space. On the other hand, in the multi-level challenge use case [28], virtual models are used to implement, by inheritance, the sequence of specification evolution.

5.5 Modeling requires flexibility

All along our experience of doing model federation in industrial and academic scenarios we have perceived that a key factor to succeed is flexibility. This flexibility concerns both practices and tools, which should be adaptable to the problem at hand and its environment (*e.g.*, IT culture).

As already stated in other contributions (*e.g.*, see [17, 14, 41]) the strict hierarchical two-level approach presents limitations in the description of complex domains with several levels of specialization. This results in the introduction of accidental complexity which pollutes domain models. We have found similar problems and experimented with multi-level in Openflexo as explained above (see Section 5.4).

In the same vein, we have found that when building domain models, the implication of the domain experts in early stages is crucial to get the right abstractions. In this scenario, modeling sessions with domain experts become really fruitful but are not without difficulties. Indeed, we have found that domain experts often feel more comfortable starting from instances instead of abstractions. As an example, we have recently acted as consultants in the development of a tool to manage enterprise strategies. This was the procedure preferred by the domain experts involved. In this scenario, constantly building or adapting metamodels so that the creation of instances is possible becomes rapidly inefficient and expensive while the alternative of using mere drawing tools unsatisfactory, as abstractions cannot be capitalized. These difficulties inspired the *Free Modeling Editor* described in Section 4.3 which has been used since then to perform *pair modeling* with domain experts.

From a more technical point of view, it is noteworthy that Openflexo relies on interpretation instead of compilation or code generation. We think this has helped us to support the aforementioned flexibility scenarios and often validate ideas with customers and domain experts in shorter iterations resulting in better end products. The advantages of interpretation are also discussed regarding other notable model-driven engineering tools such as Epsilon and Viatra [48].

5.6 Modeling needs model capitalization

In all our projects, we have observed that keeping the domain experts practices require the production of a large number of models and metamodels. Following classical software engineering principles, reuse should be systematic to reduce the cost of complex engineering processes. Therefore model capitalization is a key to large scale modeling.

But model capitalization is a difficult and elusive goal due to a lack of flexibility of modeling tools. However, we have observed that model federation significantly facilitates capitalization. First, the conceptual space provides an environment decoupled from the domain models and each virtual model within the conceptual space is decoupled from the other virtual models. This low coupling makes it possible to reuse any portion of an existing model either by composition or inheritance mechanisms. Furthermore, when reusing an element of another model a virtual model can freely adapt its semantic interpretation. Second, the levels of abstraction in the conceptual space may not necessarily be aligned with those in a domain model, and they may differ between various domain models. For example, a virtual model may reuse any element whatever its abstraction level is.

This simplicity and flexibility of reuse greatly improve the benefit of model capitalization to lower the development cost of new models. This is especially true for metamodeling which, as already mentioned, becomes crucial when adapting to new domains. Concretely, rather than developing new metamodels from scratch with all the associated cost, we tended to build on previously developed ones. The Formod tool (presented in section 4.1) perfectly exemplifies this approach by using 19 virtual models in the conceptual space while there are only four engineering views. Furthermore, a part of these 19 virtual models are reused in other projects. For example, the goal approach proposed by the Formose project extends the KAOS method. This extension is also true at the (meta)model level. We have a virtual model defining the concepts of KAOS that is reused for the Formose methodology. In this experiment, the fact that when reusing one can adapt the semantic was needed as the constraints on the KAOS models and our requirements model were different.

6 RELATED WORK

Many different reports can be found in the literature w.r.t. the use of modeling in industrial scenarios. They generally are of two types, surveys which summarize the current status of modeling practice at a given time and concrete experience reports that describe a particular use case, its challenges and its lessons learned.

Among the first group, in [46] the authors sought to find evidence supporting the claimed benefits of modeling. They report on hurdles to its adoption, *e.g.*, immature tools and lack of specific processes, and key elements for success (*e.g.*, the use of domain-specific approaches and the integration of in-house tools). Similarly, in [30] the authors describe a number of use cases and some lessons

learned. Notably, they highlight how MDE is often about the use of DSLs when adopted in non-software engineering companies. In [3] the authors account of a survey conducted in two phases with ten years span. Although they agree with the shortcomings presented in previous studies (*e.g.*, bad tools), they also evidence an increased adoption of both rigorous and casual (*e.g.*, blackboard-based) modeling. From an embedded systems perspective, in [40] the authors conducted a survey on MBE industrial practice. They report challenges related to tool interoperability and the integration of existing processes.

In the group of concrete use-case reports, many different works have been contributed in the recent years, showing that modeling for real and/or industrial scenarios remains challenging. As examples, in [6], [47] and [33] the authors report challenges and lessons learned of respectively applying component modeling, SysML and component fault trees for safety and MBE in general. In [14] the authors discuss the model-based development of a modeling workbench using open-source modeling technologies such as EMF [50] and Epsilon [35, 36].

With respect to the aforementioned works, ours complements the discussion by contributing a report on a multiuse-case experience of applying a specific modeling approach, that is, model federation, in different scenarios for more than ten years. Additionally, we account on the concurrent development, evolution and maintenance of our supporting model-federation tool, Openflexo.

Model federation approaches

From a technical point of view, model federation can be seen as a special flavor of multi-modeling [10] or multi-view modeling [2, 13]. Concretely, it corresponds to the synthetic approach of multi-view modeling, with explicit correspondences. Canonical approaches on this category can be found in [12] and [29]. However, in our view, one notable additional feature of model federation is that models (*i.e.*, *views*) remain in their technological space with their own business process and special mechanisms are used to *connect* to them.

The principles of model federation have been well known for at least two decades [31] and early works already describe experimental approaches [21, 19, 27] that adhere (sometimes partially) to these principles. Unfortunately, these aforementioned approaches have remained in either a conceptual or prototypical stage, lacking widespread adoption and/or mature toolsets. In this sense, the increased adoption of modeling in complex and emerging scenarios requiring multimodeling such as cyber-physical systems has led to a plethora of recent contributions intending to fill this gap. We discuss these recent approaches in the following, with a focus on pragmatic approaches with corresponding implementations.

In [5, 4] the authors present Syndeia, a model federation tool around SysML. Concretely, Syndeia links a SysML system architecture model with multiple engineering models and repositories to create a so-called *Total System Model*, which is a graph of models. Syndeia provides a number of different inter-model connection patterns (from simple referencing to model-transformation and synchronization) with a granularity set at the model element/attribute level. The main difference from our approach is the central role that the SysML model plays.

Although not explicitly a model federation approach, the Epsilon [36] ecosystem provides through its diverse scripting languages (*e.g.*, see EOL [34] and EML [35]) and its connectivity layer all the building blocks to build model federations.

More similar to our approach in [49] the authors present *reactive links*, an approach developed on top of *DesignSpace [18]* aimed at maintaining synchronization among heterogeneous models (which are accessible through adapters).

7 CONCLUSIONS & ROADMAPS

In this paper, we have presented our experience, both industrial and academic, in using model federation to address software engineering problems that require the semantic interoperability of heterogeneous *models*. We have illustrated the diversity of this experience with five selected use cases and discussed a number of lessons learned. We intend to continue our practice of model federation and the support to the Openflexo framework in the future. In this sense, we present below a roadmap with future challenges and goals.

Collaboration. Model federation fosters the collaboration of different stakeholders in the development of complex systems by providing the infrastructure to achieve semantic interoperability, manage consistency, etc. However, collaboration in both the development and the management of a federation model itself has not yet been explored. This will require exploring the integration of features such as multi-user views, conflict resolution, and security [43, 42, 15].

Free Modeling. We have discussed in Sections 4 and 5 the interests of *free modeling* as a means of effectively involving experts in domain modeling. We intend to further explore this path. Among others, we want to explore the means to provide a configurable level of freedom [22], perform free-modeling in other environments than box and line diagrams (*e.g.* text documents or drawings), and mix multilevel concepts in free-modeling.

Maintainability. As we have indicated in Section 5, the maintainability of the Openflexo framework has been challenging. In this sense, we intend to explore different ways to improve it. This includes enlarging the community of users and maintainers (this is already ongoing, with different labs and a company already contributing bug descriptions and bug fixes to the framework), foster re-use of models and code within the framework itself and the automation of processes.

Digital Twins. In recent years, digital twins have emerged as a means of studying and managing complex systems and as an important application domain for model-driven engineering. This raises opportunities and challenges [9] for the MDE community. Among the challenges, the management and synchronization of heterogeneous models are of notable importance and one of the strengths of model federation. In this sense, we intend to explore the application of model federation to the domain of digital twins.

ACKNOWLEDGMENT

The authors would like to thank ANR FORMOSE project members, Telindus company, and all Openflexo contributors.

REFERENCES

[1] Colin Atkinson and Thomas Kühne. 2008. Reducing accidental complexity in domain models. *Software & Systems Modeling* 7, 3 (2008), 345–359. https://doi.org/10.1007/s10270-007-0061-0

[2] Colin Atkinson, Christian Tunjic, and Torben Möller. 2015. Fundamental realization strategies for multi-view specification environments. In *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*. IEEE, 40–49. https://doi.org/10.1109/EDOC.2015.17

[3] Omar Badreddin, Rahad Khandoker, Andrew Forward, Omar Masmali, and Timothy C. Lethbridge. 2018. A decade of software design and modeling: A survey to uncover trends of the practice. In *Proceedings of the 21th acm/ieee international conference on model driven engineering languages and systems*. Association for Computing Machinery, New York, NY, USA, 245–255.

[4] Manas Bajaj, Jonathan Backhaus, Tim Walden, Manoj Waikar, Dirk Zwemer, Chris Schreiber, Ghassan Issa, Intercax, and Lockheed Martin. 2017. Graph-Based Digital Blueprint for Model Based Engineering of Complex Systems. In *INCOSE International Symposium*, Vol. 27. Wiley Online Library, 151–169.

[5] Manas Bajaj, Dirk Zwemer, Rose Yntema, Alex Phung, Amit Kumar, Anshu Dwivedi, and Manoj Waikar. 2016. MBSE++ — Foundations for Extended Model-Based Systems Engineering Across System Lifecycle. In *INCOSE International Symposium*, Vol. 26. Wiley Online Library, 2429–2445.

[6] Vincent Bertram, Shahar Maoz, Jan Oliver Ringert, Bernhard Rumpe, and Michael von Wenckstern. 2017. Component and connector views in practice: An experience report. In *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 167–177.

[7] Antoine Beugnard, Fabien Dagnat, Sylvain Guérin, and Christophe Guychard. 2015. Des situations de modélisation pour décrire un processus de modélisation. *Ingénierie des systèmes d'information* 20, 2 (2015), 41 – 66. https://doi.org/10.3166/isi.20.2.41-66 English translation available: https://p4s.enstb.org/docs/articles/isi2015-English_translation.pdf.

[8] Jean Bézivin. 2005. On the unification power of models. *Software & Systems Modeling* 4, 2 (2005), 171–188. https://doi.org/10.1007/s10270-005-0079-0

[9] Francis Bordeleau, Benoit Combemale, Romina Eramo, Mark van den Brand, and Manuel Wimmer. 2020. Towards model-driven digital twin engineering: Current opportunities and future challenges. In *Systems Modelling and Management: First International Conference, ICSMM 2020, Bergen, Norway, June 25–26, 2020, Proceedings 1*. Springer International Publishing, Cham, 43–54.

[10] Artur Boronat, Alexander Knapp, José Meseguer, and Martin Wirsing. 2009. What is a multi-modeling language?. In *Recent Trends in Algebraic Development Techniques: 19th International Workshop, WADT 2008, Pisa, Italy, June 13-16, 2008, Revised Selected Papers 19*, Andrea Corradini and Ugo Montanari (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 71–87.

[11] Chahrazed Boudjemila, Fabien Dagnat, and Salvador Martínez. 2023. Towards evolving secured multi-model systems with model federation. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 939–943.

[12] Hugo Bruneliere, Jokin Garcia Perez, Manuel Wimmer, and Jordi Cabot. 2015. EMF Views: A View Mechanism for Integrating Heterogeneous Models. In *Conceptual Modeling: 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings 34*, Paul Johannesson, Mong Li Lee, Stephen W. Liddle, Andreas L. Opdahl, and Óscar Pastor López (Eds.). Springer International Publishing, Cham, 317–325.

[13] 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.

[14] Justin Cooper, Alfonso De la Vega, Richard Paige, Dimitris Kolovos, Michael Bennett, Caroline Brown, Beatriz Sanchez Piña, and Horacio Hoyos Rodriguez. 2021. Model-based development of engine control systems: Experiences and lessons learnt. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 308–319.

[15] Istvan David, Kousar Aslam, Ivano Malavolta, and Patricia Lago. 2023. Collaborative Model-Driven Software Engineering—A systematic survey of practices and needs in industry. *Journal of Systems and Software* 199 (2023), 111626.

[16] Istvan David, Malvina Latifaj, Jakob Pietron, Weixing Zhang, Federico Ciccozzi, Ivano Malavolta, Alexander Raschke, Jan-Philipp Steghöfer, and Regina Hebig. 2023. Blended modeling in commercial and open-source model-driven software engineering tools: A systematic study. *Software and Systems Modeling* 22, 1 (2023), 415–447.

[17] Juan de Lara, Esther Guerra, and Jesús Sánchez Cuadrado. 2014. When and how to use multilevel modelling. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 24, 2 (2014), 1–46.

[18] 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 *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. Association for Computing Machinery, New York, NY, USA, 1486–1491.

[19] Trip Denton, Edward Jones, Srini Srinivasan, Ken Owens, and Richard W Buskens. 2008. NAOMI–an experimental platform for multi–modeling. In *Model Driven Engineering Languages and Systems: 11th International Conference, MoDELS 2008, Toulouse, France, September 28-October 3, 2008. Proceedings 11*, Krzysztof Czarnecki, Ileana Ober, Jean-Michel Bruel, Axel Uhl, and Markus Völter (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 143–157.

[20] Mahmoud El Hamlaoui, Bernard Coulette, Sophie Ebersold, Saloua Bennani, Mahmoud Nassar, Adil Anwar, Antoine Beugnard, Jean-Christophe Bach, Yassine Jamoussi, and Hanh Nhi Tran. 2016. Alignment of viewpoint heterogeneous design models: Emergency Department Case Study. In *4th International Workshop On the Globalization of Modeling Languages (GEMOC 2016) co-located with ACM/IEEE MODELS 2016*. Saint-Malo, France, pp. 18–27. https://hal.archives-ouvertes.fr/hal-01436169

[21] Jacky Estublier, Anh-Tuyet Le, and Jorge Villalobos. 2001. Using federations for flexible SCM systems. In *International Workshop on Software Configuration Management*. Springer Berlin Heidelberg, Berlin, Heidelberg, 163–176.

[22] Martin Gogolla, Robert Clarisó, Bran Selic, and Jordi Cabot. 2021. Towards facilitating the exploration of informal concepts in formal modeling tools. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 244–248.

[23] Fahad Rafique Golra, Antoine Beugnard, Fabien Dagnat, Sylvain Guérin, and Christophe Guychard. 2016. Continuous Requirements Engineering Using Model Federation. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 347–352. https://doi.org/10.1109/RE.2016.42

[24] Fahad Rafique Golra, Antoine Beugnard, Fabien Dagnat, Sylvain Guérin, and Christophe Guychard. 2016. Using Free Modeling as an Agile Method for Developing Domain Specific Modeling Languages. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems* (Saint-malo, France) *(MODELS '16)*. Association for Computing Machinery, New York, NY, USA, 24–34. https://doi.org/10.1145/2976767.2976807

[25] Fahad Rafique Golra, Fabien Dagnat, Jeanine Souquières, Imen Sayar, and Sylvain Guérin. 2018. Bridging the Gap Between Informal Requirements and Formal Specifications Using Model Federation. In *Software Engineering and Formal Methods*, Einar Broch Johnsen and Ina Schaefer (Eds.). Springer International Publishing, Cham, 54–69.

[26] Esther Guerra and Juan de Lara. 2018. On the quest for flexible modelling. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems* (Copenhagen, Denmark) *(MODELS '18)*. Association for Computing Machinery, New York, NY, USA, 23–33. https://doi.org/10.1145/3239372.3239376

[27] Christophe Guychard, Sylvain Guérin, Ali Koudri, Antoine Beugnard, and Fabien Dagnat. 2013. Conceptual interoperability through models federation. In *Semantic Information Federation Community Workshop / 16th International Conference, MODELS 2013*. 23.

[28] Sylvain Guérin, Joël Champeau, Jean-Christophe Bach, Antoine Beugnard, Fabien Dagnat, and Salvador Martínez. 2022. Multi-Level Modeling with Openflexo/FML: A Contribution to the Multi-Level Process Challenge. *Enterprise Modelling and Information Systems Architectures (EMISAJ)* 17 (2022), 9–1.

[29] Ábel Hegedüs, Ákos Horváth, István Ráth, and Dániel Varró. 2012. Query-driven soft interconnection of EMF models. In *International Conference on Model Driven Engineering Languages and Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 134–150.

[30] John Hutchinson, Jon Whittle, and Mark Rouncefield. 2014. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming* 89 (2014), 144–161.

[31] ISO/TC 184/SC 5 Interoperability, integration, and architectures for enterprise systems and automation applications committee. 1998. *Industrial automation systems and integration – Concepts and rules for enterprise models*. Standard ISO 14258:1998. International Organization for Standardization, Geneva, CH. https://www.iso.org/standard/24020.html

[32] ISO/IEC/IEEE. 2011. Systems and software engineering–architecture description. *ISO/IEC/IEEE 42010: 2011 (E)(Revision of ISO/IEC 42010: 2007 and IEEE Std 1471-2000)* 2011 (2011), 1–46.

[33] Rodi Jolak, Truong Ho-Quang, Michel R.V. Chaudron, and Ramon R.H. Schiffelers. 2018. Model-Based Software Engineering: A Multiple-Case Study on Challenges and Development Efforts. In *Proceedings of the 21th ACM/IEEE international conference on model driven engineering languages and systems (MODELS'18)*. Association for Computing Machinery, New York, NY, USA, 213–223. https://doi.org/10.1145/3239372.3239404

[34] Dimitrios Kolovos, Richard Paige, and Fiona Polack. 2006. The Epsilon Object Language (EOL). In *European conference on model driven architecture-foundations and applications*. Springer, Berlin, Heidelberg, 128–142.

[35] Dimitrios Kolovos, Richard Paige, and Fiona Polack. 2006. Merging models with the epsilon merging language (eml). In *International Conference on Model Driven Engineering Languages and Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 215–229.

[36] Dimitrios Kolovos, Louis Rose, Richard Paige, and Antonio García-Domínguez. 2010. *The Epsilon Book*. Eclipse. https://eclipse.dev/epsilon/doc/book/

[37] Ivan Kurtev, Jean Bézivin, and Mehmet Aksit. 2002. Technological spaces: An initial appraisal. In *4th International Symposium on Distributed Objects and Applications, DOA 2002*. 1–6.

[38] Juan de Lara and Hans Vangheluwe. 2002. AToM 3: A Tool for Multi-formalism and Meta-modelling. In *Fundamental Approaches to Software Engineering: 5th International Conference, FASE 2002 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8–12, 2002 Proceedings 5*. Springer Berlin Heidelberg, Berlin, Heidelberg, 174–188.

[39] Tom Leclerc, Soumya Paul, Jussi Roberts, Fabien Dagnat, Florian Ledoux, Jean-Christophe Bach, Marcus Wallum, Nicky Mezzina, Daniel Fischer, Sylvain Guérin, Ihab Benamer, and Pierre Jeanjean. 2023. A flexible and robust framework for the secure systems engineering of space missions. In *17th International Conference on Space Operations 2023*. Dubai, United Arab Emirates, 19 pages. https://hal.science/hal-04045293

[40] Grischa Liebel, Nadja Marko, Matthias Tichy, Andrea Leitner, and Jörgen Hansson. 2018. Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. *Software & Systems Modeling* 17 (2018), 91–113.

[41] Keila Lima, Ludovico Iovino, Maria Teresa Rossi, Rogardt Heldal, Tosin Daniel Oyetoyan, and Martina De Sanctis. 2023. Marine Data Observability using KPIS: An MDSE Approach. In *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 24–35.

[42] Salvador Martínez, Sebastien Gerard, and Jordi Cabot. 2019. On the need for intellectual property protection in model-driven co-engineering processes. In *Enterprise, Business-Process and Information Systems Modeling: 20th International Conference, BPMDS 2019, 24th International Conference, EMMSAD 2019, Held at CAiSE 2019, Rome, Italy, June 3–4, 2019, Proceedings 20*. Springer International Publishing, Cham, 169–177.

[43] Constantin Masson, Jonathan Corley, and Eugene Syriani. 2017. Feature Model for Collaborative Modeling Environments. In *Models (satellite events)*. 164–173.

[44] Abderrahman Matoussi, Frédéric Gervais, and Régine Laleau. 2011. A Goal-Based Approach to Guide the Design of an Abstract Event-B Specification. In *16th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2011, Las Vegas, Nevada, USA, 27-29 April 2011*, Isabelle Perseil, Karin K. Breitman, and Roy Sterritt (Eds.). IEEE, 139–148.

[45] Aleandro Mifsud, Georgia Samaritaki, Ulyana Tikhonova, and Jouke Stoel. 2023. Transforming an Internal Textual DSL into a Blended Modelling Environment. In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Programming Abstractions and Interactive Notations, Tools, and Environments* (Cascais, Portugal) *(PAINT 2023)*. Association for Computing Machinery, New York, NY, USA, 51–61. https://doi.org/10.1145/3623504.3623572

[46] Parastoo Mohagheghi and Vegard Dehlen. 2008. Where is the proof?-a review of experiences from applying mde in industry. In *Model Driven Architecture–Foundations and Applications: 4th European Conference, ECMDA-FA 2008, Berlin, Germany, June 9-13, 2008. Proceedings 4*. Springer Berlin Heidelberg, Berlin, Heidelberg, 432–443.

[47] Arne Nordmann and Peter Munk. 2018. Lessons learned from model-based safety assessment with SysML and component fault trees. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. Association for Computing Machinery, New York, NY, USA, 134–143.

[48] Richard F Paige and Dániel Varró. 2012. Lessons learned from building model-driven development tools. *Software & Systems Modeling* 11 (2012), 527–539.

[49] Cosmina Cristina Rațiu, Wesley KG Assunção, Rainer Haas, and Alexander Egyed. 2022. Reactive links across multi-domain engineering models. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*. Association for Computing Machinery, New York, NY, USA, 76–86.

[50] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. 2008. *EMF: eclipse modeling framework*. Pearson Education.

[51] Eugene Syriani, Hans Vangheluwe, Raphael Mannadiar, Conner Hansen, Simon Van Mierlo, and Huseyin Ergin. 2013. AToMPM: A web-based modeling environment. In *Joint proceedings of MODELS'13 Invited Talks, Demonstration Session, Poster Session, and ACM Student Research Competition co-located with the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013): September 29-October 4, 2013, Miami, USA*. 21–25.

[52] Steve Tueno, Régine Laleau, Amel Mammar, and Marc Frappier. 2017. Towards Using Ontologies for Domain Modeling within the SysML/KAOS Approach. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)* (Lisbon, Portugal). IEEE, 1–5. https://doi.org/10.1109/REW.2017.22

[53] Dustin Wüest and Martin Glinz. 2011. Flexible sketch-based requirements modeling. In *Requirements Engineering: Foundation for Software Quality: 17th International Working Conference, REFSQ 2011, Essen, Germany, March 28-30, 2011. Proceedings 17 (Lecture Notes in Computer Science)*. Springer-Verlag, Berlin, Heidelberg, 100–105. https://doi.org/10.5167/uzh-55685