Noname manuscript No.
(will be inserted by the editor)

Scalable Modeling Technologies in the Wild

An Experience Report on Wind Turbines Control Applications Development

Abel Gómez · Xabier Mendialdua · Konstantinos Barmpis · Gábor Bergmann · Jordi Cabot · Xabier de Carlos · Csaba Debreceni · Antonio Garmendia · Dimitrios S. Kolovos · Juan de Lara

Received: date / Accepted: date

Abstract Scalability in modeling has many facets, including the ability to build larger models and domain specific languages (DSLs) efficiently. With the aim of tackling some of the most prominent scalability challenges in Model-Based Engineering (MBE), the MONDO EU project developed the theoretical foundations and open-source implementation of a platform for scalable modeling and model management. The platform includes facilities for building large graphical DSLs, for splitting large models into sets of smaller interrelated fragments, to index large collections of models to speed-up their querying, and to enable the collaborative construction and refinement of complex models, among other features.

This paper reports on the tools provided by MONDO that Ikerlan, a medium-sized technology center which in the last decade has embraced the MBE paradigm, adopted in order to improve their processes. This experience produced as a result a set of model editors and related technologies that fostered collaboration and scalability in the development of wind turbine control applications. In order to evaluate the benefits obtained, an on-site evaluation of the tools was performed. This evaluation shows that scalable MBE technologies give new growth opportunities to small and medium-sized organizations.

Keywords Model-Based Engineering (MBE), Scalability, Domain Specific Graphical Modeling Languages, Collaborative Modeling, Model Indexing, Experience Report

A. Gómez
Internet Interdisciplinary Intitute (IN3), Universitat Oberta de Catalunya (UOC), Barcelona, Spain
E-mail: agomezlla@uoc.edu

X. Mendialdua · X. de Carlos
Ikerlan Research Center, Arrasate, Spain
E-mail: {xmendialdua|xdecarlos}@ikerlan.es

Konstantinos Barmpis · D. S. Kolovos
Dept. of Computer Science, University of York, York, UK
E-mail: {dimitris.kolovos|konstantinos.barmpis}@york.ac.uk

G. Bergmann · C. Debreceni
MTA-BME Lendület Research Group on Cyber-Physical Systems, Budapest University of Technology and Economics, Budapest, Hungary
E-mail: {bergman|debreceni}@mit.bme.hu

J. Cabot
ICREA – Internet Interdisciplinary Intitute (IN3), Universitat Oberta de Catalunya (UOC), Barcelona, Spain
E-mail: jordi.cabot@icrea.cat

A. Garmendia · J. de Lara
Universidad Autónoma de Madrid, Madrid, Spain
E-mail: {Antonio.Garmendia|Juan.deLara}@uam.es

1 Introduction

Ikerlan is a Spanish private non-profit technology center created in 1974, and a technological R&D actor within the Mondragon Corporation [50]. Ikerlan is a point of reference for innovation, dedicated to advanced technology transfer to industry and comprehensive product development (from concept to implementation) for a wide variety of domains: energy (wind and solar power, and storage systems), transportation (railway and vertical), automation, industrial, health, home appliances, etc. Ikerlan works closely with companies to improve their competitiveness through the application of technological knowledge to develop innovative products as well as by providing new tools and methodologies for implementation in design and production processes. It has a staff of more than 200 qualified researchers and engineers, with experience in interdisciplinary work and capable of tackling complex problems. As a center of excellence in the transfer of technology, more than 800 R&D projects have been completed until 2019 in cooperation with companies developing new products, implementing customized systems in design,

and manufacturing processes. The actions of Ikerlan are devoted to providing product, process and service innovation for companies. To achieve these ambitious goals, Ikerlan offers comprehensive solutions that combine the three domains in which it has a high degree of specialization and expertise: Electronics, Information and Communication Technologies (EICT), Energy and Power Electronics and Advanced Manufacturing.

Focusing on energy, and specifically in wind power, Ikerlan has been working for the last 11 years in the development of supervisory and control platforms for wind turbines for one of the world's leading companies in the field of renewable energy. Wind turbines are complex systems where hardware and software components need to interact in intricate ways. To tackle this complexity, Model-based Engineering (MBE) [85] technologies were introduced in 2008 in Ikerlan for the engineering of the supervisory and control systems. The goal for adopting and investing in MBE was to improve the productivity and competitiveness of its industrial customers by enhancing their software development processes using, as Section 2 sketches, Domain Specific Languages (DSL) [53] and code generators [21]. The experiences reported by customers showed significant productivity increases, indicating that MBE has been critical in the development of new software products faster, cheaper and with fewer errors than in previous projects.

However, too often, MBE tools and methodologies have targeted the construction and processing of small models in non-distributed environments. This focus neglects common scalability challenges [59], considering that a more typical scenario involves different engineers working in collaboration at distributed locations. Handling these issues is a challenging task that requires specific solutions that foster scalability, as we discuss in Section 3.

In 2013, the MONDO project[1] was launched with the aim of tackling some of the most prominent challenges of scalability in MBE by developing the theoretical foundations and open-source implementation of a platform for scalable modeling and model management. Achieving scalability in MBE involves Achieving scalability in modeling and MDE involves being able to construct large models and domain specific languages in a systematic manner, enabling teams of modelers to construct and refine large models in a collaborative manner, advancing the state-of-the-art in model querying and transformations tools so that they can cope with large models (of the scale of millions of model elements), and providing an infrastructure for efficient storage, indexing and retrieval of large models. Among the technologies developed[2], Section 4 focuses on the ones that can provide Ikerlan the opportunity to offer its customers software development methodologies in geographically distributed scenarios where **multiple users can work collaboratively with large models and DSLs**; and Section 5 describes the different solutions developed in Ikerlan using the MONDO platform.

Section 6 describes how the scalable MONDO technologies have been evaluated in Ikerlan and presents the results obtained. Section 7 analyses related work, especially focusing on the technologies developed and adopted in this experience. Finally, Section 8 draws conclusions and discusses this experience on the application of scalable modeling technologies in a company like Ikerlan.

This article is an extension of our previous paper presented at the ECMFA'17 conference [44]. In this version, we provide an extended description of the MONDO framework in Section 4, expanding upon the capabilities of the different solutions of the MONDO framework that are of interest for Ikerlan. We also provide further details on how the MONDO framework was adopted in Ikerlan, presenting in Section 5 a description of the different solutions implemented. In Section 6, we report on the evaluations performed, including a description of the different experimental setups. And finally, we provide a comparison with related work in Section 7.

2 Background: Towards an MBE Development Process

In 2008 Ikerlan adopted the MBE paradigm to develop software applications in a wide range of domains to increase the productivity in their software development processes. In 2008, the application of this more modern MBE approach started in its wind turbines division [100]. This section introduces the changes such modernization implied and the improvements achieved.

The result of this initial modernization effort forms the baseline for the experience reported in this paper, and provides the rationale for the decisions taken and technologies used for the evolution of the framework.

2.1 Wind Turbines

A wind turbine is a complex system composed of a set of physical subsystems whose aim is to convert wind energy into electrical energy. The *Wind Turbine Control System* (WTCS) [1] is the system which monitors and controls all of the subsystems that make up the wind turbine. Its aim is to maximize the generation of electrical energy, always ensuring the correct operation of the turbine and avoiding any problem which can cause any damage to it. It monitors the status of the wind turbine and the environmental conditions, making decisions to obtain the highest energy production. The WTCS is a HW/SW system that runs on a dedicated hardware platform. This is connected to the wind turbine through assorted communications to receive infor-

[1] http://www.mondo-project.org/

[2] http://www.mondo-project.org/technologies

mation from inputs (sensors, device state signals, etc.) and to actuate on outputs (device actuators).

2.2 In the Beginning, there was only Code

Ikerlan's work is primarily focused on the development of the software part of the WTCS, which mainly handles control algorithms, communications and user interfaces. The initial WTCS – i.e., prior to 2008 – was designed to manage the system operation, but its design did not originally consider extensibility or customizability for particular wind turbines.

The size of the entire control system was roughly half a million lines of C++ code. There were several software components such as control algorithms, communications, interfacing, simulation, and so on. The development process was already based on modeling artifacts, mostly state-charts for the behavior of the system, and structural diagrams for the main architecture of the system; but models were not driving the process nor used for generating code.

The control system in operation was one already deployed in hundreds of wind turbines. As the system was considered proven and reliable, starting from scratch was not a realistic choice, instead, a more evolutionary approach was needed.

Considering the limitations of the initial system, the chief architect was requesting and internally leading the need for a long-term shift in the design of the WTCS; and indeed, the entire development team was actually encouraging such a shift, and so was well prepared for change.

This is the environment where the initial modernization took place, which indeed was appropriate for the implantation of MBE techniques according to – at the moment – current [17] and later [48,49] experiences: a **highly motivated workforce**, committed to perform changes in their organization and processes, but following a **progressive and iterative approach** which allows a direct return of investment.

2.3 Towards using Model-based Engineering for Offshore Wind Turbine Control System Development

The system specification and the control runtime were tangled in the software implementation, and software elements were actually mapped to control physical subsystems. This situation introduced some degree of complexity when modifications and customizations were needed at the system level. This situation also prevented the specialization of different engineers, something the growing development team was calling for. It was necessary to separate the wind turbine system specification (i.e., which elements are controlled by an individual system, which specific inputs/outputs are used, which specific params are set, etc.) from the runtime control software (i.e., the real-time components of the system).

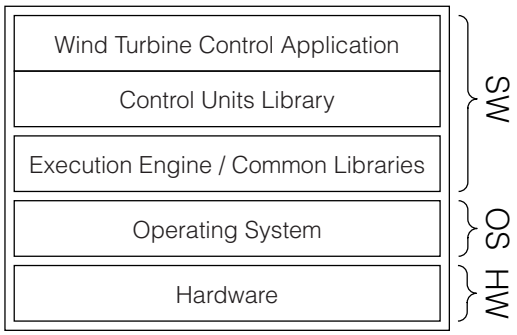

Fig. 1: HW/SW architecture of a Wind Turbine Control System

Figure 1 shows the improved HW/SW architecture of the WTCS that enabled such separation of concerns. The two lower layers refer to the HW and the operating system. The software layer is composed of the following components:

The *Execution Engine* is a component that cyclically executes the algorithms to monitor and control the wind turbine.

The *Control Units Library* contains a set of reusable control algorithms. These are basic blocks, with well defined interfaces, which can be instantiated and interconnected to implement the *Wind Turbine Control Application*.

A *Wind Turbine Control Application (WTCA)* comprises the set of algorithms that must be executed in order to ensure the correct operation of the wind turbine the WTCS is monitoring and controlling. The control algorithms of the wind turbine are specified by instantiating control units available in the *Control Units Library* and by combining those instances.

In this software architecture, the *Execution Engine* is a stable software component which does not vary from one wind turbine to another. The *Control Units Library* is also a stable component which, generally, does not vary either, unless some new device is used in a wind turbine and a custom control unit has to be implemented to control it. Finally, the WTCA is the part of the software in a WTCS that is customized for each wind turbine, depending on the specific requirements that WTCS must met.

The development of a WTCS is a process whereby a multidisciplinary team of hardware, software and telecommunications engineers, as well as electrical, mechanical and other engineers work in collaboration. However, since the top layer (WTCA) is the only part that is specific for each different wind turbine, this paper will only focus on the development of the WTCA.

In the late 2000's, Ikerlan had already implemented different solutions based on DSLs built on top of open-source tools, and more specifically, on Eclipse [94] and its Eclipse Modeling Framework (EMF) [89]. Based on this experience,

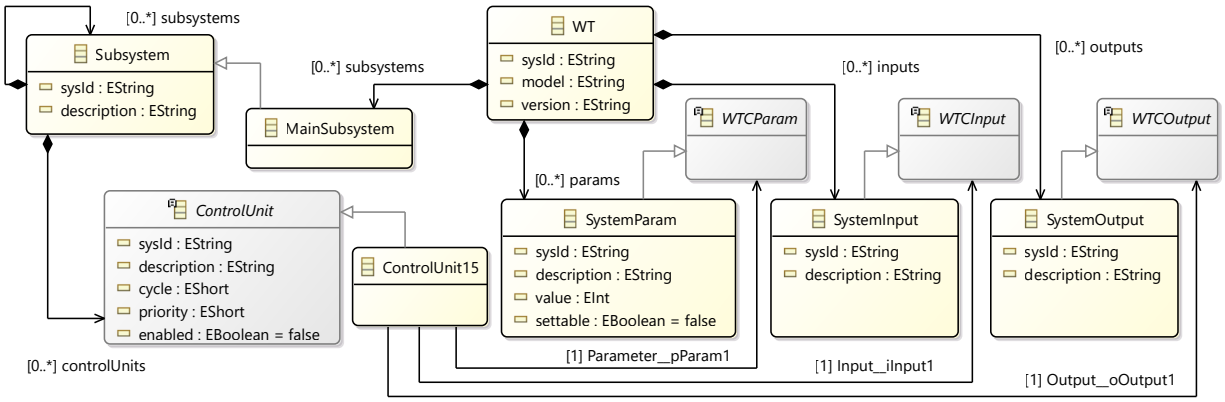

Fig. 2: Metamodel of the Wind Turbine DSL (excerpt)

the development of a new set of tools exploiting a domain-specific modeling tool for the development of control applications for wind turbines – the so-called *Wind Turbine Control Modeler* (WTCM) – started, using Eclipse and related open-source technologies. The WTCM provides the catalogue of available *control units* that engineers can use to develop the algorithms to monitor and control the subsystems of the wind turbine.

The control system of a wind turbine is typically composed of almost 2000 basic control units, involving about 2000 inputs and up to 1000 outputs, depending on the specific model configuration. A control unit is a basic and reusable control algorithm that may be combined with other control units to build more complex algorithms. The control system is structured into logical subsystems, each controlling different physical subsystems or parts of them. The control of a wind turbine is built through the aggregation of basic control units in order to specify those complex algorithms.

Figure 2 shows an excerpt of the metamodel, which describes the abstract syntax of the DSL provided by the WTCM. As the figure shows, the control system of a wind turbine (*WT*) is a parameterizable element (via *SystemParam*) that processes a set of *SystemInputs* to obtain a set of *SystemOutputs*. This control system contains a set of *MainSubsystems* – a specific kind of *Subsystems*[3] – and that in turn, contain *ControlUnits*. *Subsystems* may also contain other *Subsystems* following the composite pattern. *ControlUnits* – which may be also parameterized using *WTCParams* – implement the algorithms which process a set of inputs (*WTCInput*) to provide a set of outputs (*WTCOutput*). For the sake of simplicity, Figure 2 only shows an example *ControlUnit15*, which processes a single input according to a single param, and produces a single output. As aforementioned, the *Control Units Library* is a stable element which, in general, does not vary. As such, our Wind Turbine DSL contains a subclass extending *ControlUnit* for each one of the reusable control algorithms we have in the *Control Units Library*.

[3] We differentiate between *MainSubsystems* and *Subsystems* because, in our case study, the code generated for the former is inherently different from the code generated for the latter.

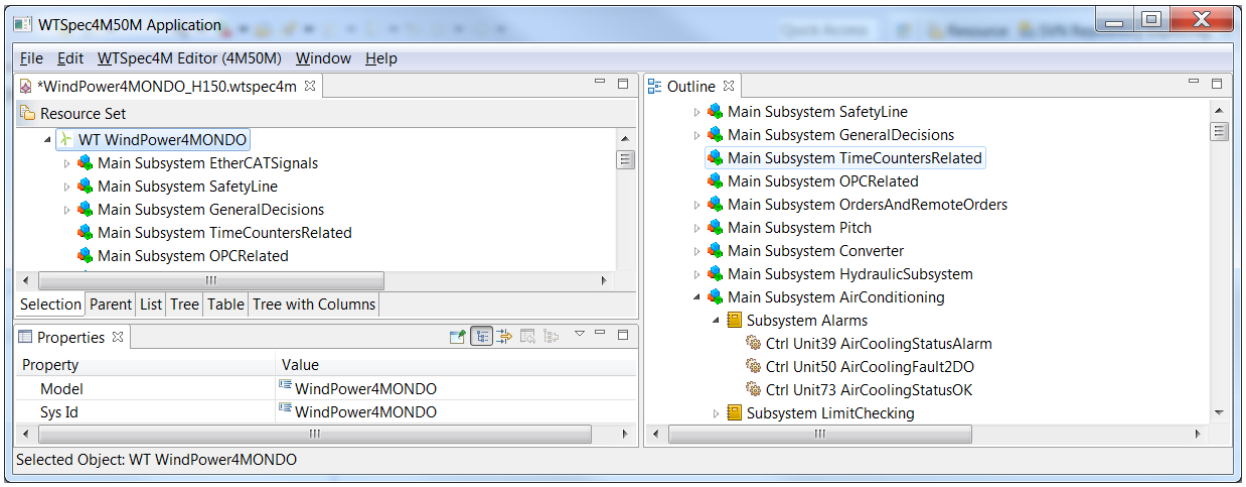

Fig. 3: Screenshot of Wind Turbine Control Modeler

Figure 3 shows what the initial implementation of the WTCM – which implements the Wind Turbine DSL – looks like. As it can be seen, the initial WTCM is an Eclipse application, which enables engineers to edit WTCS models using a regular EMF tree editor. This regular tree editor is a single-user editor designed to be executed in a desktop computer. Models created using this tree editor are stored as XMI [74] files.

Once a model has been created using the WTCM, the actual C++ code for monitoring and controlling the physical subsystems can be generated using model-to-text transformations expressed in the Epsilon Generation Language (EGL) [57].

2.4 Measurable Benefits

The adoption of MBE practices facilitated coping with the inherent system complexity. One of the reasons was that the systems were specified in terms closer to the problem domain, which gave engineers the ability to detect and resolve issues at that level, while separating them from the software implementation details. Thanks to the efficient separation of concerns in the improved platform, the knowledge about the system greatly improved among the engineers in charge of its development. As a result, there was a substantial gain in terms of efficiency and productivity: Ikerlan adopted the newly MBE-based solutions in a small group of developers (around 5 persons), which from that moment on applied them to the development of 20 new systems with nearly 100 subsystems each. Other developers continued using the old manual approach. After measuring the time spent to specify the different models and generate the code for a WT, and comparing such time with the the time spent to manually code a similar WTCA in terms of complexity, the results showed that the time to develop a new WTCA targeted for a new product using MBE techniques was reduced from 240 weeks to only 15 for a junior engineer; while for a senior engineer was reduced from 60 to 15 weeks. These improvements where obtained in an incremental way, by refining the platform that had already been developed, and by taking advantage of the experience of the in-house developers.

3 Challenges

Today, MBE is used by a group of around 10 developers working in the R&D area of Ikerlan developing control systems for new families of wind turbines. The aim of Ikerlan is to extend MBE technologies to other activities such as wind turbine control customization for specific customer requirements. Considering that there are more than 30 different variants of control applications that are still being developed using non-MBE methodologies, it is expected that the number of different models can grow significantly within the next years, increasing the number of developers using modeling techniques up to 20 or more in the mid-term.

This requirement poses a major challenge to the initial WTCM approach presented before, as it lacks the features that would enable a team of engineers to work collaboratively: each engineer has to work with their own copy of the model, and model merging operations – e.g., to include changes performed by others – need to be carried out manually. This manual process is a complex, tedious and error prone activity that can take more than half an hour depending on the amount and type of changes made.

Another important limitation is that engineers do not have mechanisms to work with a subset of the model. That means that all engineers must always work with the whole model (which may add up to thousands of elements as described in the previous section), although a small subset of elements of the model can be sometimes enough to perform a specific modeling or validation activity.

Based on these limitations, the following challenges have been identified to improve the development of WTCAs:

1. The **first challenge** is to move from a single-user modeling tool built for an engineer to work in an isolated way, to a modeling tool enabling several engineers to work collaboratively and securely by sharing – possibly big – models located in a central repository.
2. The **second challenge** is the ability to edit partial models or model fragments. It should be noted that a typical wind turbine model contains thousands of control units. Thus, the ability to edit partial models or model fragments allows each engineer to work with a specific part of the model (as opposed to the whole model), thereby easing modeling activities. Additionally, the use of model fragments allows minimizing the volume of data transferred over the network and limits the number of merge conflicts.
3. The **third challenge** is to graphically display and edit WTCS hierarchical models. Graphical models are more expressive for this domain as they ease the identification of relationships between model elements. This is an important enhancement with respect to the initial tree-based editor, where relationships have to be found using auxiliary views of the editor. Such graphical models, may contain, again, thousands of elements. Thus, graphical editors handling such models must be capable of displaying big models while being usable. Such scalable editors should include additional features like filtering facilities, hierarchies of diagrams, etc. Figure 4 shows a mockup of what a graphical WTCM would look like.
4. Finally, the **fourth challenge** is to enable model editing using a lightweight mobile device – instead of a lap-

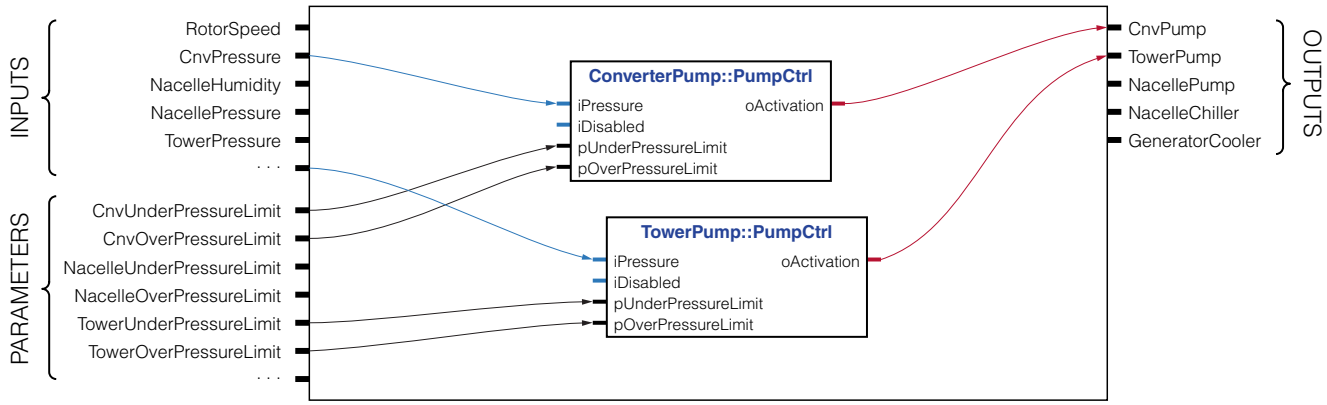

Fig. 4: Wind Turbine Control graphical modeling conceptual mock-up

top – to perform the modeling activities on site in the wind farm.

4 The MONDO platform

The goal of Ikerlan joining the MONDO project was twofold: *(i)* to provide a real-world scenario where the scalable modeling technologies can be tested and evaluated; and *(ii)* to improve their development processes by introducing such techniques. It is worth noting that the fact that MONDO was being developed as an open-source platform was important, as one of our requirements was to continue using open-source MDE tools. The MONDO platform is the open-source solution[4] for scalable modeling and model management developed within the MONDO project. Its main purpose is to address the shortage of scalable and collaborative support in state-of-the art technologies within the MBE landscape. It is composed by several components for the development of scalable Eclipse-based editors and DSLs, collaborative modeling, model indexing and scalable model transformations and queries.

In what follows we describe these main MONDO components, focusing on the aspects that were relevant to Ikerlan's implementation of its Wind Power infrastructure, and more specifically, on the components that play an important role in the solutions presented in Section 5 and the evaluation presented in Section 6.

4.1 Collaboration

The **MONDO Collaboration Framework**[5] [23] is a set of server-side and client-side tools and services aimed at improving collaboration among multiple organizations as well as multiple professionals within a single organization.

[4] Sources available at `https://github.com/mondo-project/`
[5] `https://github.com/FTSRG/mondo-collab-framework`

The framework offers two main avenues of improvement over the state-of-the-art. First, *better security* through model-aware fine-grained access control of modeling artifacts. Second, *better conflict management* either *(i)* through simultaneous multi-user model editing in online collaboration; or in case of traditional offline collaboration, *(ii)* through rule-based model element-level locking to avoid conflicts, and *(iii)* through intelligent automated merging to resolve conflicts when they do occur.

In the following, we present a brief overview of challenges as well as design decisions to address them, with the primary focus on access control features.

4.1.1 Motivation and challenges

Access control — The development of complex systems necessitates intensified collaboration between distributed teams of different stakeholders (system integrators, engineers of component providers/suppliers, certification authorities, etc.), potentially employed by different companies. However, such collaboration introduces significant challenges for access control management to protect the respective Intellectual Property (IP) of different parties. For instance, the detailed internal design of a component needs to be revealed to certification authorities, but it needs to be hidden from competitors who might supply a different component in the system. Even within a single organization, export control regulations may prevent certain artifacts from being divulged to external company offices. Furthermore, certain critical aspects of the system model may only be modified by domain experts with appropriate qualifications. For all these reasons, engineering artifacts must undergo *access control*.

Model fragmentation and granularity — At the time the development of *MONDO Collaboration Framework* was started, access control management in most existing collaborative modeling repositories was still in a preliminary phase, supporting permission assignments only

globally, or using fragments (i.e. on the level of model files or projects). In order to share various parts of system models between collaborators, models needed to be split into a large number of static fragments (e.g. over 1000 for automotive models). Consequently, the re-fragmentation of the model (which may be necessary when collaboration roles, model contents or policy decisions evolve) is hard or infeasible. Therefore, static fragmentation becomes both a scalability and usability bottleneck. Static fragmentation can be mitigated by *fine-grained access control* where each model element may have its own set of permissions. Unfortunately, large industrial models may have millions of model elements, thus explicitly assigning permissions for each element would be labor-intensive and error-prone. Moreover, understanding and maintaining permissions after model changes can also be problematic.

Online and offline collaboration — System models are traditionally developed either in an offline or online manner. In *offline collaboration*, which is very common in current MBE practice, engineers check out an artifact from a repository into a local copy, work on their disconnected copy for an arbitrary duration, and then commit local changes to the repository in asynchronous (long) transactions. In *online collaboration*, engineers may simultaneously access and edit a model in short synchronous transactions which are immediately propagated to all other users. This latter strategy is similar to online collaborative office tools like Google Docs, and it is showcased in modeling tools like WebGME [64], AToMPM [20], GenMyModel / Web Modeling Framework [96] or MetaEdit+ [99].

In the offline case, all information available to a specific user (accounting for read access control) needs to be provided as a self-contained model that can be displayed and edited by existing off-the-shelf modeling tools. Simply hiding elements from the user interface of a desktop modeling tool is insufficient, as the IP is still accessible on the client side e.g. by file inspection. Hence, a filtered but modifiable model that excludes any confidential information needs to be sent to each client. Furthermore, the online scenario requires immediate change propagation, where model modifications need to be evaluated in an efficient, reactive way in response to the change; this change processing must account for write access control (is the user allowed to make this change?) as well as read access control (which other users are allowed to see the effects of this change?).

Conflicts and merging — Effective collaboration requires that engineers working simultaneously do not interfere with each other. This is traditionally achieved by partitioning the model into fragment files and providing file-level locks; while object-level locks are also available in some systems. The former is too coarse-grained and thus prone to overlocking (needlessly preventing unrelated edits), while the latter may be too fine-grained and may easily lead to accidental underlocking (where locks fail to prevent conflicting edits). Conflicts are especially prevalent in asynchronous offline collaboration, where individual collaborators execute longer checkout-commit cycles, with a larger chance of conflicting edits happening in the meantime.

When conflicts do occur, the alternative versions have to be correctly merged. Computing differences, conflicts and merged resolutions is significantly more complex over MBE models with graph-based knowledge representations (and complex well-formedness rules) than over textual source code. Inter-model dependencies make conflicts surprisingly easy to introduce and hard to resolve.

4.1.2 Architectural overview

The central component of the *MONDO Collaboration Framework* is the *MONDO Collaboration Server* [10], which enforces fine-grained access control during both offline and online collaborative modeling. The server provides secure views with precisely defined model access to each collaborator, synchronized with each other by *bidirectional model transformations* [25]. In particular, a transformation applies read access restrictions to present a filtered view to a user, while a separate transformation merges changes proposed by a user into the unfiltered model (if write access restrictions allow); both transformations are automatically derived from the declared access control policy.

In the *offline collaboration scenario*, the server hosts models in a *version control system* (VCS). However, traditional source code repositories cannot address the challenges associated with fine-grained access control of models. The *MONDO Collaboration Server* provides multiple user-specific version histories instead, which are all synchronized with each other; see the architecture depicted in Figure 5. The full and unabridged model is versioned in the so-called *gold repository* which is inaccessible to the actual users. There is also a separate *front repository* dedicated to each user, that contains a copy of the gold repository, with complete version history (black vertical arrows in Figure 5), where the contents of each version snapshot is filtered according to the read access privileges of that user (see solid green horizontal arrows in Figure 5). Users are given access to a dedicated front repository so that they can read the current or historical contents of the model files (up to their read privileges) and commit their changes (which may be denied based on write permissions). In case changes (see dashed arrows in Figure 5) are successfully committed, they are transparently propagated to the gold repository (blue dashed horizontal arrow), and then, by the usual filtering, to the front repositories

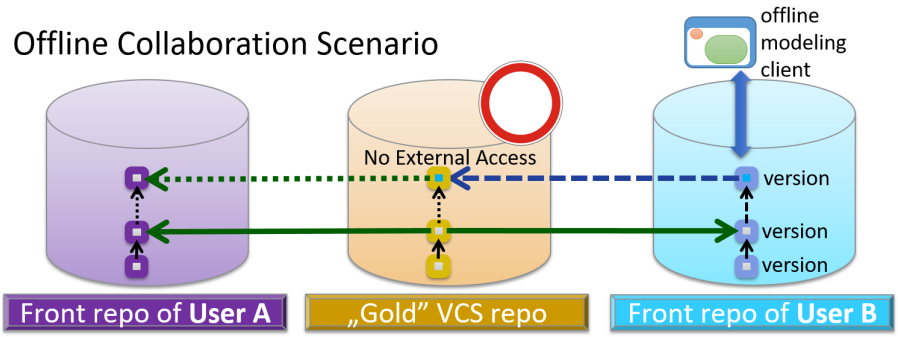

Fig. 5: Overview of access control (offline collaboration)

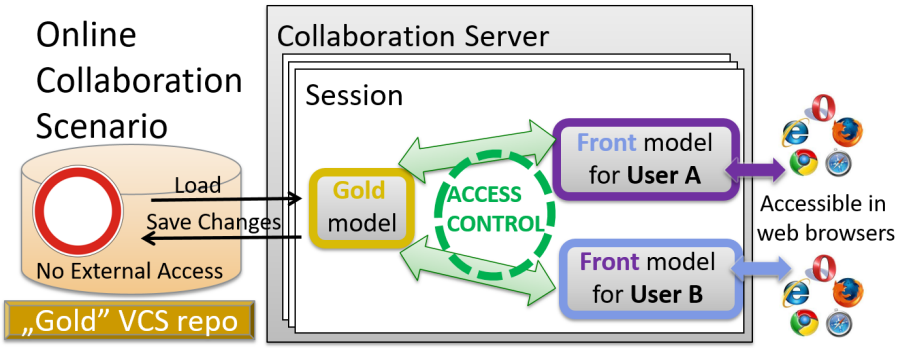

Fig. 6: Overview of access control (online collaboration)

of all other users (green dashed horizontal arrow). In their normal day-to-day workflow, users interact with their front repository using standard VCS protocols and off-the-shelf VCS client software.

For an *online collaboration scenario*, MONDO provides an online collaborative modeling tool, where users can open, view and edit models stored in the VCS backends using a web browser, thus no client software needs to be installed. As depicted in Figure 6, multiple users can collaborate on the same model simultaneously, enjoying the same access control mechanism that underlies the offline collaboration framework. The editor is provided as an Eclipse RAP-based web application [95]; note that the model editing Web UI is language-dependent, and thus has to be provided separately for each supported modeling domain.

4.1.3 Query-based access control

As argued before, security needs a fine-grained access control mechanism that must determine the permissions of each model element in the model. It would be possible to adopt a low-level specification method, i.e. require engineers to manually assign security permissions to each model element, each attribute slot value and each cross-reference. However, industrial wisdom (see e.g. the survey [98], itself citing several older surveys in the field) advises against such a solution, and advocates for "high-level specification of access rights" instead. Indeed, the low-level approach suffers from the following difficulties:

- the necessary per-element permission annotations would form a large amount of additional user input;
- manually specifying and correctly maintaining them according to organizational security policies would put an undue burden on each collaborating specialist that edits the model;
- as such a large amount of security-critical user input, they would be prone to contain errors (with business-critical consequences) that are difficult to test for in an automated fashion;
- overseeing them and auditing them for conformance with organizational security policies would be labor-intensive for security officers.

We have therefore chosen a high-level alternative for specifying access rights. A *rule-based* approach for concisely defining fine-grained model access control policies has been proposed in [10], where a single rule may grant or deny permissions for many elements in a model selected according to a *model query*. A *MONDO Access Control Policy* combines individual rules with various priority classes, in addition to applying sensible defaults, according to semantics defined in [24]. An incremental evaluation mechanism was provided in [11] so that the access control policies can

be reactively applied after each single editing operation in the online collaboration scenario.

The key distinguishing feature of *MONDO Access Control* from (non-model-based) standard rule-based policy languages (such as XACML [43]) is that it applies to models. Therefore, it uses expressive model queries (*graph patterns*) to identify the model elements to which the rules of the policy are applicable. Furthermore, it ensures *referential integrity* of the model throughout the interactions of rules, so that the filtered local copies provided in the offline scenario are consistent, if incomplete, models.

4.1.4 Locking and merging

The *MONDO Collaboration Framework* implements a novel *property-based locking* technique introduced in [26], which is a common generalization of several widely used locking approaches. Property-based locks can be used to protect model editing operations (including complex refactorings) by capturing their pre-conditions as a declarative query. The lock forbids other users from concurrently changing the model in a way that affects the result set of the query, thereby violating a precondition of the lock owner. The *MONDO Collaboration Server* applies locks analogously to access restrictions.

Nevertheless, conflicts occasionally do occur and have to be merged. The framework includes a client-side tool called *DSEMerge* [27] – and seamlessly integrated into Eclipse-based client modeling environments – that performs automated search-based model merge. The tool computes and displays possible candidate resolutions (conflict-free merged models) for a conflict, from which the user can select the most suitable one. In the fortunate case where there are no conflicts between element-level changes (even if there was a file-level conflict), the tool comes up with a single "perfect" solution candidate that includes all changes, so that the user does not even have to choose, but merely approve. Otherwise, choosing a solution is assisted by several summary metrics such as the number of elements deleted, number of changes included in the solution, etc.

More precisely, the tool identifies the set of individual model editing operations on the two branches to be merged (since the last common ancestor), and automatically produces the set of "maximally merged" conflict-free states that cannot be extended further by including more of these editing operations. The developer of the modeling language may provide additional domain-specific insights to further improve this mechanism. By default, the solution candidates are constructed by breaking down the overall model differences into atomic edit operations (e.g. modify attribute, delete object), but this may be overridden if complex domain-specific *semantic edit operations* and resolution rules are provided. Furthermore, beside obvious hard contradictions (e.g. assigning different values to the same attribute), the notion of "conflict" may include domain-specific *well-formedness rules* as "soft goals". The automated generation of candidate solutions may also take into account a distinction between *essential and incidental modifications*, e.g. a core design decision vs. simply rearranging a diagram; obviously, the former kind is prioritized over the latter.

4.2 Model indexing

Hawk[6] is a *model indexer*: it keeps an up-to-date read-only view of model collections (aka a *model index*) and facilitates performing efficient queries on them. Hawk monitors local or remote data locations (such as local file directories, version control systems like Git or SVN or remote URLs) and periodically synchronises itself with model files of interest in these monitored locations.

4.2.1 Using Hawk as part of the MONDO Platform

Hawk enables scalable queries on models in the MONDO platform. By querying model indexes of such models (instead of the originating model fragment files), the MONDO platform is able to perform various read-only operations in a scalable and incremental manner. For example, a graphical editor needing to display parts of a very large EMF model is able to query Hawk for the subset of model elements it currently needs to display, hence visualizing it in a more efficient way (in terms of time and memory use) than if it needed to load the entire model from its originating fragments. This can be extended to queries performing arbitrary computations on parts of large fragmented models (such as the ones created by Ikerlan), as such queries can execute without having to load, resolve or navigate through the parts of the model they are not interested in. Further information on Hawk, including its updating and querying policies, can be found in [5,6,7,38].

4.2.2 Architectural overview

Figure 7 shows an overview of Hawk, focusing on how its components interact to maintain an up-to-date model index:

Version Control Managers — Specific for each version control system (VCS), these components need to compute the set of changed files (added, removed or updated) relative to the revision last indexed by Hawk (and the current latest version in the VCS).

Model Resource Factories — These components offer parsers for reading from specific model persistence formats, like EMF models persisted in XMI or Modelio [66]

[6] `https://github.com/mondo-project/mondo-hawk`

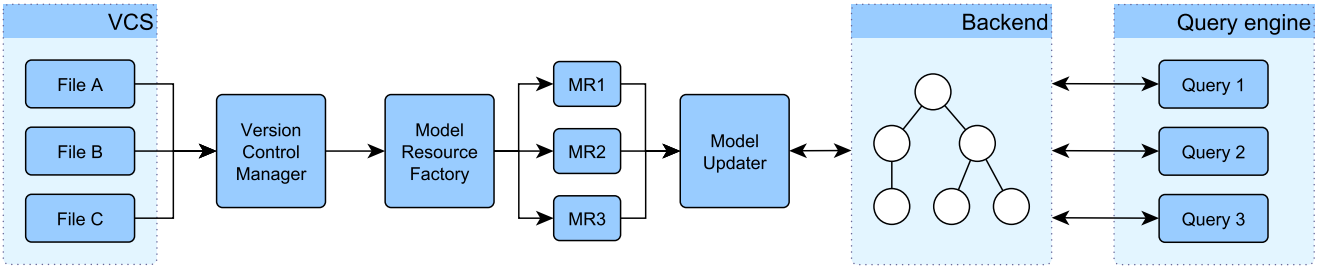

Fig. 7: Component architecture of Hawk

models in XML. Such parsers take as input the contents of files (provided by version control managers) and produce as output a uniform in-memory representation ("resource") of such models.

Model Updater — This component receives resources created by the appropriate model parser(s), and inserts/updates them into Hawk's persistence (database), through back-end specific drivers. The structure of these stores assumes that such back-ends provide a mechanism for rapidly accessing specific elements using a key (aka database indexing). This is the case for many popular stores such as MySQL, MongoDB or Neo4J, which include their own embedded database indexes.

Query Engine — This component provides a bridge between Hawk and any model management tools querying it. Queries can be performed on both local and remote Hawk model indexers and can return textual or numeric results, as well as collections of model elements.

4.3 Pattern-based scalable DSLs

DSL-tao[7] [80] is the component within the MONDO platform that enables the systematic development of scalable graphical DSLs based on EMF. Its working scheme is depicted in Figure 8.

DSL-tao supports the systematic creation of DSLs by reusing patterns. It supports 5 types of patterns, covering the different aspects of DSL design: domain, design, infrastructure, semantics and concrete syntax. The patterns either help in designing the metamodel (e.g., design and domain patterns), or contribute with services for the final environment (e.g., infrastructure, semantics and concrete syntax). This way, DSL-tao profits from an extensible library of metamodeling patterns, which are instantiated and combined when building a DSL. Some of the currently supported domain patterns (e.g., for variants of state machine, workflows, ex-

[7] https://github.com/jdelara/DSL-tao

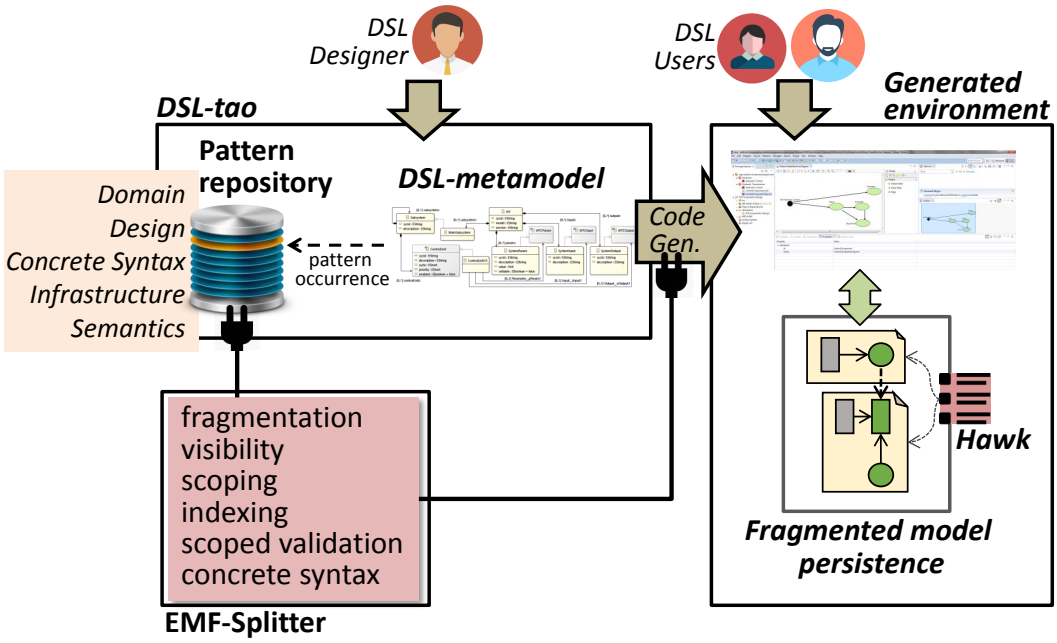

Fig. 8: Working scheme of DSL-tao

pressions, etc) were extracted from the analysis of existing metamodels in public repositories [80].

DSL-tao provides extensibility mechanisms, so that new patterns can be added to its repository. A new pattern can contribute its own wizard to facilitate its instantiation, and with services (realized by code generation) to contribute functionality to the final modeling environment. This way, a number of infrastructure patterns have been created, to define modularization strategies for DSLs, including patterns to define fragmentation strategies (so that models are not persisted in a monolithic way), visibility (to define access rules for model elements), scoping (to specify reference scope), filtering (to select interesting objects) and graphical concrete syntax support, among others. The application of these patterns, and their associated services are contributed by EMF-Splitter[8] [41], a tool that can also be used standalone. The generated environments rely on Sirius for graphical model editing, while Hawk [4] – the MONDO model indexer – is used for efficient look-up of models element across fragments.

In the following, we explain the different parts of the approach: the pattern structure and application process (Section 4.3.1), the modularization patterns (Section 4.3.2), and the concrete syntax pattern (Section 4.3.3).

4.3.1 DSL Patterns

Our DSL patterns have the form of a metamodel, which is conceptually located one meta-level higher than the domain metamodel being built. The elements of a pattern are called *roles* [60].

To apply a pattern, its metamodel is instantiated and then integrated into the existing domain metamodel. This integration requires first a mapping from the roles in the pattern instance to the elements in the domain metamodel. Once this identification is made, an automatic merge operation adds to the domain metamodel the roles in the pattern that are not mapped, and annotates the metamodel elements with the pattern roles.

Figure 9 illustrates the scheme of pattern application. Label 1 depicts a simple metamodeling design pattern describing tree structures [13]. The pattern has two class roles (*Tree* and *Node*), two reference roles (*root* and *children*) and two attribute roles (*ordered* and *ident*). The *ordered* attribute role is a *configuration* attribute (indicated using the "@1" potency annotation [2,61]), meaning that it takes a value when the pattern is instantiated, rather than being mapped to an attribute in the domain metamodel.

In Figure 9, label 2 shows the pattern instantiation the metamodel designer has chosen, where two instances of *children* and *ident* have been created. Label 3 shows an excerpt of a domain metamodel the designer is working on. To apply

[8] https://github.com/antoniogarmendia/EMF-Splitter

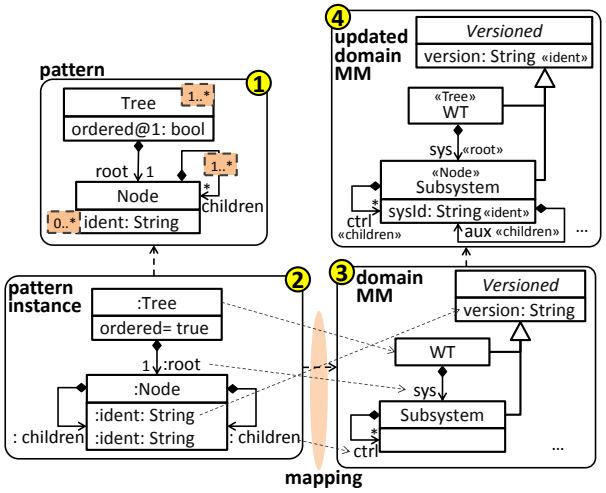

Fig. 9: Scheme of pattern application.

the pattern, mappings between the pattern instance and the domain metamodel need to be established. Our mechanism supports structural matching, so that, e.g., *ident* in *Node* is mapped to *version* in class *Versioned*. As the Figure shows, the merge operation creates the elements not mapped in the pattern instance in the domain metamodel, and annotates the domain metamodel elements with the pattern roles. The created elements take by default the name of the role element, but this name can be changed when the mapping is being defined (as in case of the example). When the pattern is applied, if an element with same name already exists in the metamodel, a new name is generated (concatenating a numerical index). Other conflicts (e.g., inheritance cycles) are automatically detected either when the mapping is created, or when the pattern itself is applied. In both cases, an error is reported, and the engineer is asked to change the mappings.

It must be stressed that the pattern may bring services to the generated domain-specific environment, for example, constraints ensuring a proper order in case of ordered trees, and operations ordering the tree according to the selected identifiers. By defining these services on the pattern, they become reusable for any metamodel where the pattern is applied. Moreover, this approach also brings benefits when the DSL evolves. In this case, the designer needs to change the mappings to accommodate the change, and regenerate the environment again with the updated service provided by the pattern. This way, the service code does not need to be changed by hand, but only the mappings need to be modified. If the services would have been defined directly on the domain metamodel, the designer would have needed to manually modify the code implementing the service, which typically would require more effort.

While role cardinalities support the customization of a pattern for a given context, our patterns also support more coarse-grained variation. For example, in case of trees, in

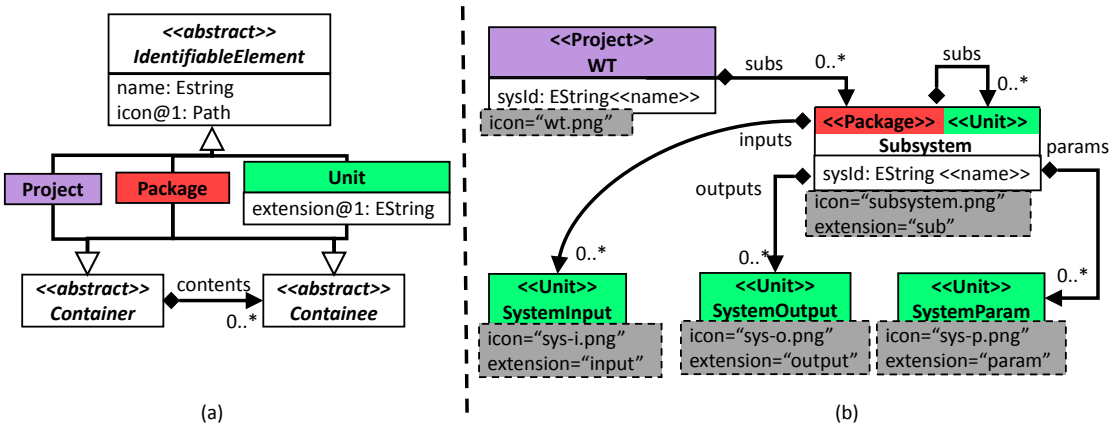

Fig. 10: (a) Fragmentation modularity pattern (b) Applying the fragmentation pattern to the WT metamodel.

addition to the simple structure shown in Figure 9, we may have structured trees (enabling the differentiation of leaf and non-leaf classes, leading to a structure similar to the composite pattern [37]), trees changing over time, or overlapping trees (whose nodes may share a parent node), among others [13]. This way, we support pattern variants, which are selected through a feature model [51].

4.3.2 DSL modularization patterns

In MDE, models are the main assets used to create software. However models frequently lack native modularization mechanisms, unless they are explicitly encoded in the modeling language and implemented in the supporting modeling environment. Thus, we have devised a number of patterns to provide modularity services to the DSL environments. In this sense, we follow a similar philosophy to the Java Development Tools (JDT). In this way, each model corresponds to an Eclipse project, and the model content will then be fragmented into units, and organized in folders, with a direct mapping to the file system.

The main pattern to achieve this modularity is the fragmentation pattern, shown in Figure 10 (a). This pattern defines *Project*, *Package* and *Unit* as classes roles. Language engineers can configure which classes in the domain metamodel will play those roles. For example, typically the class that is mapped to *Project* will be the root class in the metamodel (the one that directly or indirectly contains all the other classes). As a result, each time this class is instantiated in the generated environment, a new modeling project (i.e., a folder that will hold all fragments of the model) is produced. Similarly, when a class with role *Package* is instantiated, the environment creates a folder in the file system, together with a hidden file storing the value of the class attributes and non-containment references. Finally, instantiating a class with *Unit* role results in the creation of a file that holds instances of the classes that can be directly or indirectly reached by means of containment relations.

The application of the fragmentation pattern to the WT metamodel is shown in Figure 10 (b). The *WT* class has been assigned role *Project*, while *Subsystem* has been assigned roles *Package* and *Unit*. This means that within *WT* projects we can find both, folders and files to represent subsystems, and the model developer chooses which representation is desired. Finally, classes *SystemInput*, *SystemOutput* and *SystemParam* are assigned role *Unit*.

Hence, altogether, by using this pattern the DSL designer can devise a suitable fragmentation strategy for the DSL, so that models are no longer monolithic, but fragmented and structured according to the chosen strategy. Technically, we rely on EMF crossreferences to realize references across fragments. These are based on the creation of proxy objects, which are only resolved when the reference is navigated. Moreover, the environment uses Hawk to optimize the handling of the different model fragments.

4.3.3 DSL concrete syntax patterns

In our approach, the concrete syntax of the DSL is defined by applying patterns as well. Two patterns are currently supported, to visualize the models in the form of graphs (graphical syntax), or tables (tabular syntax). These concrete syntax patterns enable the creation of language families with similar representations. In the case of graphical syntax, we support the definition of nodes, edges and spatial relationships between elements for visualization in a graphical editor. The concrete syntax patterns can be used to automate the generation of editors supporting the defined syntax (which otherwise should be implemented by hand), and can be attached to domain (e.g., the state machine pattern [80]) or design patterns (e.g., the tree pattern) in order to define different default visualization options for them.

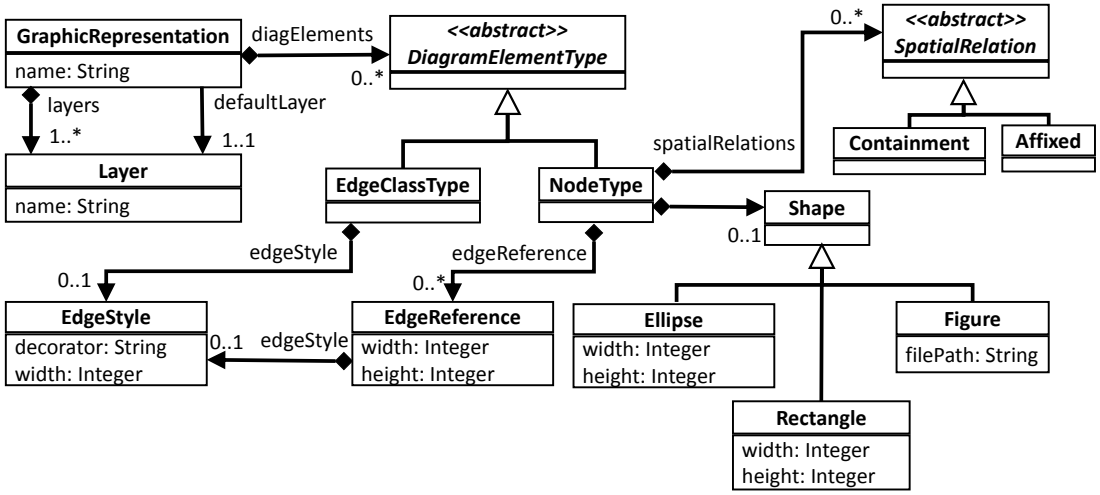

Fig. 11: Excerpt of the *GraphicRepresentation* metamodel

The application of the concrete syntax must map the classes in the domain metamodel to a graphical element. This process has many specificities (like selecting figures for nodes and decorators for edges), that is why we implemented a dedicated wizard to facilitate this task (shown later in Figures 13 and 14). For instance, the customized wizard implements heuristics to decide which classes will be represented as nodes, which ones as edges, the attributes to display, and the nodes that are containers of other nodes. Then, the designer can refine the inferred concrete syntax and fine-tune the visual representation for nodes and edges, using the pattern wizard.

The wizard allows customizing the heuristics that best suit the metamodel for which the environment is generated. The heuristics analyze the classes and references defined in the metamodel and infer a visual representation. For example, to choose the root node there are two strategies: choosing the class that contains more children (classes) or the class that does not have parents. The first strategy counts how many classes are contained in each class and selects the one that contains more. The second strategy suggests classes that are not contained in other classes. Both strategies are based on the tree of containment references defined in the metamodel.

With respect to edges, our heuristics support selecting edge-like classes and references as edges. In the first case, we select the classes that define two non-containment references with lower bound 0 or 1, and upper bound 1. These two references will be mapped to the source and target of the edge representation for the class. In second case, the heuristics identify the references that will be displayed graphically as edges, compartments or affixes.

We consider abstract syntax metamodels defined in Ecore, for which the concrete syntax can be established according to Figure 11. In particular, classes in the domain metamodel can be represented either as nodes (class *NodeType*) or as edges (class *EdgeClassType*) and are referred to through the class *DiagramElementType*. In case the class is represented as an edge, it is possible to configure the references of the class acting as source and target of the edge. References in the domain metamodel can be mapped into *EdgeReferences*, and their concrete syntax annotations are mapped into an *EdgeStyle*. All created graphical elements are included in the default layer.

Technically, the concrete syntax pattern relies on Sirius. This way, once the pattern is applied, the generated modeling environment will automatically feature Sirius-based editors to edit the different model fragments.

5 MONDO Solutions for Offshore Wind Power

Using the previous core MONDO infrastructure, three modeling solutions have been implemented in Ikerlan. These three modeling solutions aim to cover the different scenarios showing up in Ikerlan (see Section 6.1), where the challenges listed in Section 3 emerge. Next we describe these solutions, whose sources and demonstrators are also available online[9].

5.1 The Online Concurrent WTCS Modeling Solution

The *Online Concurrent WTCS Modeling Solution* is a web modeling application that allows multiple modelers to share a modeling session. All modelers can work concurrently with the same WTCS model versioned in a model repository. All changes performed by a modeler are automatically propagated to all other modelers, constantly providing them with an up-to-date version of the model.

[9] `https://github.com/mondo-project/mondo-demo-wt`

Besides allowing concurrent modeling activities, this solution also supports working with partial models. It means that each modeler can work with a different view of the same model, thus editing a different fragment of it. This feature is possible thanks to the model access rules, which can be defined by using the model access control policy language provided by the *MONDO Collaboration Framework* (see subsection 4.1). This is achieved by creating a custom model view containing only the model elements a user is allowed to see and edit, hiding all other elements in the model. Unauthorized editing of model elements is also prevented as a consequence of using this process. The *Online Concurrent WTCS Modeling solution* allows modelers to commit the changes performed in the model to the model repository.

The *Online Concurrent WTCS Modeling Solution* is a web application that can be accessed both from desktop computers and mobile devices. Each in-memory model session is managed by the *MONDO Collaboration Server*, which transparently enforces the model access rules by *(i)* filtering the content to be presented in the web-based model editor to each of several simultaneously connected users and *(ii)* intercepting their model edit operations for write access control and propagation to the views of other users[10].

5.2 The Offline Collaborative WTCS Modeling Solution

The *Offline Collaborative WTCS Modeling Solution* is an Eclipse-based modeling application that runs locally. It enables several engineers to work with a shared model, but unlike the solution presented above, collaborative modeling is done asynchronously, i.e., each modeler working with the shared model edits a local copy of the shared WTCS model, which is checked out (or updated) from a model repository. When model editing has finished – or whenever the user decides – a commit operation is requested and the *MONDO Collaboration Server* carries out the operation.

From the perspective of engineers using the modeling platform, the main difference between the offline and online modeling solutions is that the offline solution allows the user to check out a disconnected, persisted copy of the model. This local version can then be processed by legacy or off-the-shelf MBE tooling. The user has to explicitly commit any local changes to the server, and merge with changes that may have been performed in the mean time. This is aided by a full version history maintained on the server, providing a standard *version control repository* interface for client-side or server-side tooling to interact with.

Access control restrictions, imposed by the same policy introduced above, are of course enforced in the offline solution as well. This is achieved by introducing access-filtered replicas of the version control repository hosting the models. There will be a separate *front repository* for each different user role and it will contain the (version history of the) filtered partial model for that user role. The *MONDO Collaboration Server* will be in charge of keeping synchronized all the front repositories with the so-called *gold repository* (hidden from users), which will always contain the whole WTCS model. This synchronization will be done when any commit operation is carried out. For synchronizing the gold repository and all the front repositories, model access rules described above will be used by the *MONDO Collaboration Framework*, to take into account the permissions each user will have to access and edit model elements. This way forbidden model editing operations will be detected by the *MONDO Collaboration Framework* before any commit operation is performed and the commit requested by the modeler will be rejected.

As aforementioned, in MONDO, the management of partial models in an offline manner is handled by the *MONDO Collaboration Server* (see subsection 4.1) which performs the synchronization between all the front repositories and the gold repository; there exists a different front repository for each different user type which contains the partial model for that user type. The modeling client, therefore, only needs to include an off-the-shelf VCS client in order to be able to participate (by communicating with their dedicated front repository using standard VCS protocols).

However, for more efficient handling of conflicts, the *Offline Collaborative WTCS Modeling Solution* is additionally bundled with *DSEMerge* [27] (see subsection 4.1), which is the MONDO solution for automated model merging. *DSEMerge* may be customized for each modeling language by domain-specific operations and well-formedness constraints in order to reduce the search space and obtain accurate merged models (that are semantically consistent as well). Such operations and constraints should be defined only once for a given domain, and not once per each merge activity. In case of *Offline Collaborative WTCS Modeling Solution*, we specified no domain-specific operations, and only applied the following custom well-formedness constraints: it is mandatory that all SystemInput and SystemParam instances should be referenced by at least one control unit and each SystemOutput has to be referenced by a unique control unit.

Apart from collaboration related operations, the way a model user will work with the *Offline Collaborative WTCS Modeling solution* is quite similar to the way the modeler was working with the tree-based single user modeling tool introduced in Section 2.3: in summary, the engineers will now have all the features that will allow them to work in collaboration with other engineers but without having to change the way they were used to.

[10] For more information and screenshots about the *Online Concurrent WTCS Modeling Solution* (Section 5.1) and the *Offline Collaborative WTCS Modeling Solution* (Section 5.2), refer to the public MONDO deliverable [97].

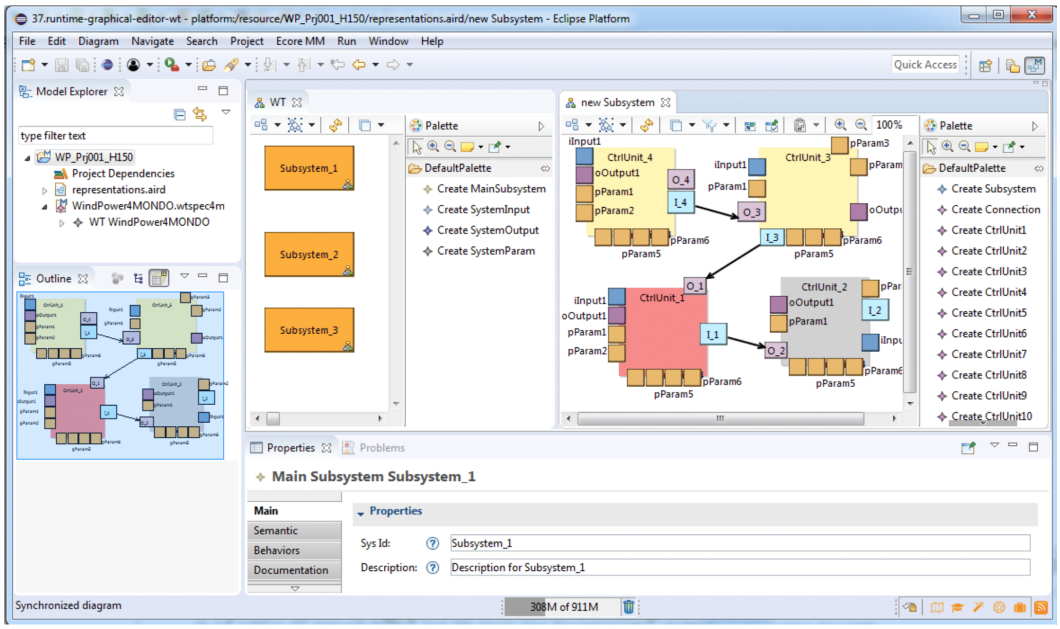

Fig. 12: Graphical editor for Offline Collaborative WTCS Modeling solution

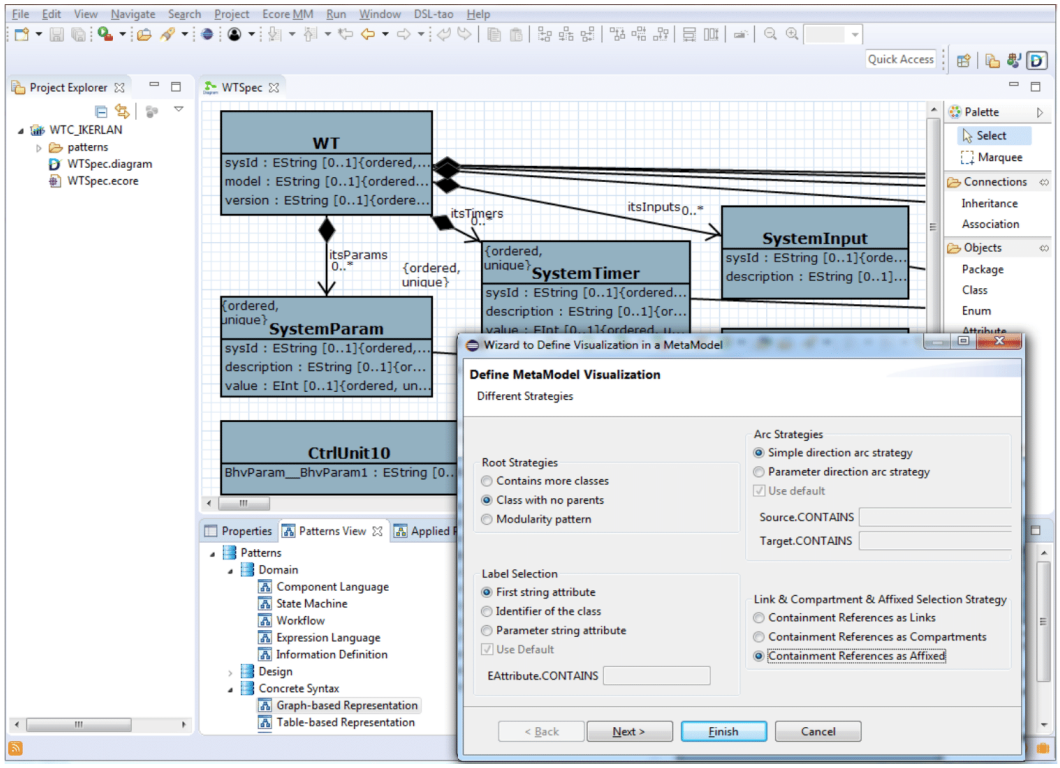

Fig. 13: Applying the Graph-based representation pattern

5.3 The Offline Graphical Collaborative WTCS Modeling Solution

The *Offline Graphical Collaborative WTCS Modeling Solution* is a Sirius-based editor, which allows editing WTCS models graphically, as depicted in Figure 12. As mentioned above, WTCS models edited by the offline modeling solution are built with the Wind Power domain specific tree editor, but, as mentioned in Section 3, an important challenge is the capability of editing WTCS models graphically, with an ed-

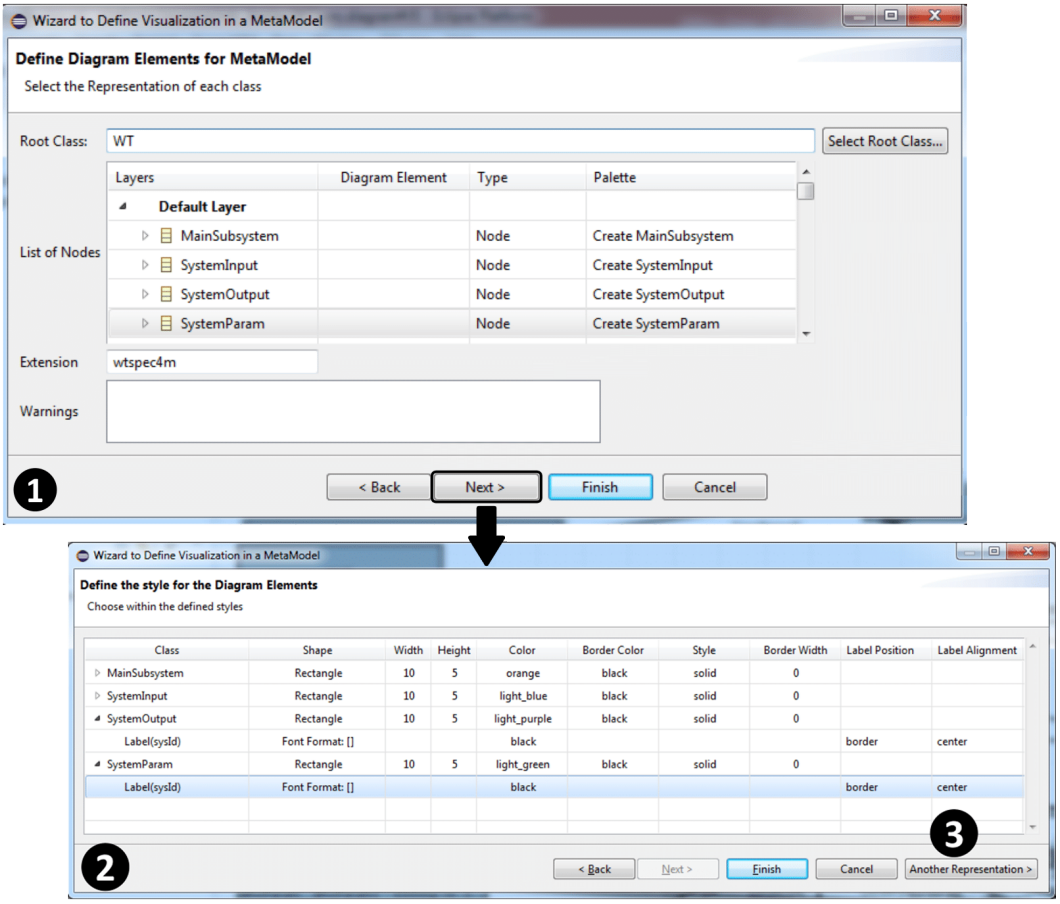

Fig. 14: Shape selection and visualization details

itor that provides advanced features like drill-down, element filtering, layers, custom views, different diagrams, etc.

This challenge has been addressed by integrating a Sirius based graphical editor in the *Offline Collaboration WTCS Modeling solution*. This way, the same model (and model fragments) that can be edited by the EMF based tree editor and the graphical editor based on Sirius, taking advantage of all the functionality Sirius provides. Both client modeling perspectives enjoy the same collaborative features.

It must be noted that the ability to edit models using graphical diagrams based on Sirius is not a contribution of MONDO project, but a contribution of the developer team of Sirius. As the MONDO project was progressing, Sirius has become one of the most widely used Eclipse components to build graphical EMF-based model editors. For this reason it was decided to use Sirius for addressing the challenge of editing graphical models in this Wind Power use case. Hence, the MONDO project has been focused on facilitating creation of and integration with Sirius based modeling editors.

Taking this into account, we integrated the *graph-based representation* concrete-syntax pattern explained in Section 4.3.3 into DSL-tao in order to simplify the process of construction of this kind of editors based on Sirius. Although Sirius provides its own tools to design graphical editors, this design activity carried out using Sirius design tools can be time consuming, and the *graph-based representation* pattern aims to make this process much more simple to the DSL designers, hiding low level details of Sirius graphical specifications, which must be taken into account if design is carried out using Sirius design tools. Therefore, we designed the graphical concrete syntax for the wind power DSL using the *graph-based representation* pattern provided by DSL-tao. This design pattern provides a wizard-based design process. The steps followed to build the graphical syntax are shown in Figures 13 and 14.

The first step consists in selecting the pattern through DSL-tao, as shown in Figure 13. This Figure shows the main DSL-tao canvas in the background, with the DSL metamodel, and a pattern view that permits browsing and selecting the patterns to be applied to the metamodel.

It shows the first page dedicated pattern wizard, which allows the combination of heuristics to automatically generate a graphical representation for the metamodel. The wizard offers different heuristics for the automatic selection of a *root* class representing the diagram (box *Root strategies*). In this case, we use the *Class with no parents* to select the class that

is not contained in any other class, which in this example is the *WT* class. The heuristics also permit the automatic proposal of a suitable label for each class representation (*Label Selection*). For this example, we select the first string attribute of the class as its label. In addition, the heuristics can detect the Edge-like nodes (*Arc Strategies*), which are the classes holding two non-containment references. Finally, it is possible to decide how containment references will be represented (section *Links & Compartment & Affixed Selection Strategy*). This way, it is possible to visualize them as edges, or via spatial relationships (containment or adjacency). In our case, we select that containment references will be mapped as affixed elements since, at it can be seen in Figure 12, the parameters, input and outputs are positioned at the border of the control units.

Once these initial settings are configured, the representation for nodes and edges can be refined, and organized in layers, as seen in Figure 14 (label 1). Finally, visualization details (shapes, link decorators, colors, labels, etc.) can be set, as shown in Figure 14 (label 2).

The wizard permits defining various visualizations for different parts of the meta-model, to create a multi-view environment (see button "Another representation" with label 3 in Figure 14). This way, we generated a view that permits exploring the hierarchical representation of the WTCS models. As a result, the diagram on the left in Figure 12 shows a global view of model subsystems, and the diagram on the right, shows the details of the *Subsystem_1* element.

The result of this design process is the graphical model editor we showed in Figure 12. We have used this editor to carry out the evaluation related to the graphical visualization and editing of WTCS models.

6 Evaluation

In order to assess the success of the MONDO technologies in the Ikerlan MBE processes, an evaluation has been performed. This section reports on the methodology used and results obtained from this evaluation.

6.1 Evaluation Framework

Three realistic scenarios, which express the four challenges presented in Section 3, are used for the validation of the MONDO technologies:

Scenario S1: Wind turbine control design — Different system engineers work concurrently on a single model, modeling different subsystems. Each system engineer works on a partial model or submodel and MONDO technologies shall merge all the partial models into a unified one.

Scenario S2: Wind turbine commissioning — A system engineer works on a particular submodel during the commissioning of a subsystem. Transformations for code generation will only take into account the artefacts contained in (and referenced from) the submodel the engineer is working on.

Scenario S3: Maintenance activities in the wind farm using mobile devices — A maintenance operator of a wind farm detects a malfunction of a non-critical element in a wind turbine. This causes the wind turbine to be out of operation. The engineer makes a minor change in the control model and obtains new code, including automatically generated tests, to put the wind turbine into operation in a degraded mode. These changes – typically minor changes in values and conditions – are made using a tablet or a mobile device.

The solutions presented in Section 5 support the three aforementioned scenarios:

Scenario S1 can be supported by two MONDO solutions: the *Online Concurrent WTCS Modeling Solution* and the *Offline Collaborative WTCS Modeling Solution*. Although in a different way, both solutions allow several engineers to work in parallel on the algorithms for the different subsystems of a wind turbine. Likewise, both solutions manage all changes performed by each engineer merging them in a single WTCS model.

Scenario S2 is supported by the *Offline Collaborative WTCS Modeling Solution*. A specific subsystem manager is allowed to load only a fragment of the entire WTCS model. Thus they can only edit the part of the model for the subsystem under their responsibility. The subsystem manager also has the ability to generate code for the subsystem.

Scenario S3 is supported by the *Online Concurrent WTCS Modeling Solution*. As this is a web based solution deployed on a web server, it can be accessed using a tablet. Thus, on-site modeling operations related to maintenance activities can be performed by the maintenance operator.

In order to better measure the impact of MONDO technology in the development of a WTCA, several indicators have been defined for purposes of this assessment. Such indicators express industrial needs of Ikerlan that were difficult to jointly satisfy with the state of the art before the MONDO project.

Their suitability for assessing the modeling solutions is explained below:

Time for committing model changes — Collaboration was not supported in the modeling solution for the Wind Power domain before the MONDO project. Thus, a merging operation when several engineers modified their models had to be carried out by hand. Model merging was tedious and error prone task, which could take from two or three minutes for slight model modifications, up to

Table 1: Quantitative Measures and Evaluation Criteria

ID	DESCRIPTION	SUFFICIENT	GOOD	EXCELLENT
QN1	Increase in time for loading a model on a tablet instead of on a PC	25%	15%	10%
QN2	Number of concurrent users working with a model	2	3	5+
QN3	Time for change propagation and notification among concurrent users	<5 s	<3 s	<1 s
QN4	Maximum number of elements that can be displayed in a diagram	25	50	>50
QN5	Time for loading a diagram having 25 elements to be displayed	2 s	1 s	<1 s
QN6	Time for committing model changes	<5 s	<3 s	<1 s
QN7	Performance impact caused by the MONDO Collaboration Framework	<5%	<2%	<1%
QN8	Time reduction for building graphical domain specific modeling editors	25%	50%	75%

Table 2: Qualitative Measures

ID	DESCRIPTION
QL1	Is there a methodology which specifies how a large DSL should be constructed?
QL2	Is there a tool support for the methodology, which guides the user on the construction of a large DSL?
QL3	Does this tool provide a way to create a basic but fully functional collaborative domain specific modeling tool?
QL4	Is MONDO technology mature enough to be used in industrial solutions?
QL5	Does MONDO technology allow concurrent editing of a model?
QL6	Does MONDO technology allow partial loading of models?
QL7	Does MONDO technology allow progressive loading of a model?
QL8	Does MONDO technology allow working with several modeling languages in a single tool?
QL9	Can a model be edited using a tablet?

30 minutes when a large set of changes were made. This merging operation was made using file comparison tools, and the engineer who was performing the merge operation had to decide how files should be merged. This aims at capturing the benefits of using MONDO-based collaborative tooling.

Impact on performance derived from using MONDO Collaboration technology — When an innovative technical solution is used to add new features, the improvement obtained is often penalized with a loss of performance on the previously available features. Therefore, one potential risk of using the *MONDO Collaboration Framework*, is that, although it enables new modeling features that were not available before – like the collaboration among several modelers who work with a domain specific modeling tool – it can also increase the time required to carry out some operations that were already available in the solutions prior to MONDO (e.g., model visualization on editors, code generation using model to text transformations, etc.). In this sense, this indicator aims at measuring the possible loss of performance caused by the *MONDO Collaboration Framework*.

Time reduction for building graphical domain specific modeling editors — This indicator aims at measuring the reduction on the time required to construct a graphical editor to model a WTCS using the design tools provided by MONDO (i.e., DSL-tao [80]) as opposed to the time required for constructing it with other standard tools.

Based on the expertise of Ikerlan in this domain, as well as their industrial needs, these generic indicators have been materialized into a set of quantitative and qualitative measures, which are summarized in Tables 1 and 2.

The quantitative evaluation is carried out based on the measures shown in Table 1. The table also summarizes the corresponding evaluation criteria used to determine the level of success of the proposed solution. Section 6.3.1 gives further details on the experiments executed to perform the evaluation and on the rationale behind the evaluation criteria. The qualitative evaluation is carried out amongst the engineers participating in the evaluations using the questions in Table 2. In their answers, a four point scale (i.e., *fully*, *largely*, *partially*, *none*) is used together with the opportunity for respondents to provide comments and clarifications regarding their assessment. Section 6.3.2 reports on the comments and feedback received.

Table 3: WTCS model user types

DOMAIN ENGINEER	MODEL VIEW	ELEMENTS IN WTCS ALLOWED TO ACCESS/EDIT
Principal Engineer (PE)	Full WTCS model	The *Principal Engineer* will have permissions to modify (add, remove and update) any element in a WTCS model.
IOManager (IOM)	*SystemInput* and *SystemOutput* elements	Only *SystemInput* and *SystemOutput* elements can be edited by an *IOManager*. Access to any other element types in the model will be denied for the *IOManager*.
Subsystem Manager (SM)	*Subsystems*	A *Subsystem Manager* will be allowed to edit the content of any *MainSubsystem* element in the WTCS model. It means they can edit (add, remove and modify) subsystems contained in any main subsystem as well as *CtrlUnit* elements in the main subsystem or in a contained subsystems, at any level in the hierarchy of subsystems. They will be allowed to read *SystemInput*, *SystemOutput*, *SystemParam*, *SystemFault* and *SystemTimer* type elements to reference them from the *CtrlUnit* elements being edited but they will not be allowed to add, remove nor modify any of these referenced elements.
Generator Manager (GM)	*Generator* subsystem	A *Generator Manager* will be allowed to edit the content of the *Generator* subsystem (i.e., the *MainSubsystem* element in the model named as *Generator*). This user will be allowed to do the same operations as *SubsystemManager* is allowed to do, but only for a predefined *MainSubsystem* (*Generator*).
Converter Manager (CM)	*Converter* subsystem	A *Converter Manager* will be allowed to edit the content of the *Converter* subsystem (i.e., the *MainSubsystem* element in the model named as *Converter*). This user will be allowed to do the same operations as *SubsystemManager* is allowed to do, but only for a predefined *MainSubsystem* (*Converter*).
Pitch Manager (PM)	*Pitch* subsystem	A *Pitch Manager* will be allowed to edit the content of the *Pitch* subsystem (i.e., the *MainSubsystem* element in the model named as *Pitch*). This user will be allowed to do the same operations as *SubsystemManager* is allowed to do, but only for a predefined *MainSubsystem* (*Pitch*).

6.2 Evaluation Setup

Next, we describe the elements that characterize the evaluation setup, such as persons involved, roles and access rules, model characteristics, and hardware and equipment used.

6.2.1 Persons involved and roles

To carry out the evaluation a set of different WTCS model user types (Wind Power domain engineers) have been defined and the model access rules for all of them have been implemented. These user types and the model elements they will be allowed to edit are described in Table 3.

Permissions for each user type are set by implementing model access rules using the access control facilities provided by the *MONDO Collaboration Framework*. Listing 1 shows an example of model access rules defined for a WTCS model user (*Converter Manager*). Permission to access the subsystem named *Converter* is set on the first rule, while the other restricts access to all other subsystems (i.e., those not named *Converter*).

Listing 1: Model access rules for Converter Manager

```
/* Grant access to Converter MainSubsystem for
     converter manager */
rule enableConverter permit RW to convertermanager {
  query "macl.project.objectSubsystemNithName"
  bind name value "Converter"
}
/* Deny access to any other MainSubsystem for
     converter manager */
rule denyOtherSubsystemsForConverter deny RW to
     convertermanager {
  query "macl.project.objectSubsystemNotHavingName"
  bind name value "Converter"
}
```

6.2.2 Model size and characteristics

The evaluation has been carried out with an offshore WTCS model having 5441 objects and 5429 references, distributed as shown in Table 4. The maximum depth for subsystems in the hierarchy is 4.

6.2.3 Equipment

Evaluations requiring a single computer have been carried out using a *Dell Latitude E5540* laptop with an Intel Core i7

Table 4: Model characteristics

ELEMENT TYPE	NUMBER OF INSTANCES
WT	1
MainSubsystem	20
Subsystem	110
CtrlUnit	1590
SystemInput	1698
SystemOutput	900
SystemParameter	285
SystemAlarm	798
SystemTimer	39

processor at 2.7GHz, 16 GB of RAM, and using Windows 7 Professional SP1.

In the experiments that required several engineers working concurrently, the evaluation has been carried out using *(i)* 3 × *Dell Latitude E5540*, 15.6" laptop, Core i7 @ 2.7GHz, 16 GB of RAM, Windows 7 Professional SP1, and *(ii)* 2 × *Dell Latitude E6530*, 15.6" laptop, Core i7 @ 2.9GHz, 8 GB of RAM, Windows 7 Professional SP1, .

Evaluations performed on tablet have been done on a 9" device: an *HTC Nexus 9*, with 32 GB of storage, 2 GB of RAM, and running *Android* 5.0.

6.3 Evaluation Results

Next we present how we evaluated the different quantitative and qualitative measures, and the results obtained.

6.3.1 Quantitative Measures

QN1 *– Increase in time for loading a model on a tablet instead of on a PC —* The *Online Concurrent WTCS Modeling Solution* has been used for the evaluation of this measure. The aim of this measure was to compare the time required to load a model on a modeling tool running on a tablet, and time required to load the same model on a modeling tool running on a PC. Thus, the *Online Concurrent Modeling Solution* has been separately executed first on a PC and then on a tablet, and the time required to load the same model in both devices has been measured. Since there are several different engineers that will work on the modeling of the WTCS, and different model fragments are loaded for each type of engineer, this operation has been carried out for the following engineers: *PrincipalEngineer*, *IOManager*, *GeneratorManager*, *ConverterManager*, *PitchManager* and *Yaw Manager*. Table 5 collects all the measured times, along with the model objects and references contained in the loaded model (in the case of the *Principal Engineer*, who accesses the full model) or model fragment (in the case of all the other engineers).

Table 5: Time required for model loading (QN1)

ENGINEER TYPE	MODEL OBJECTS / REFERENCES	TIME FOR LOADING ON PC	TIME FOR LOADING ON TABLET
Principal Engineer	5441 / 5429	9.1 s	11.0 s
IO Manager	2599 / 0	3.8 s	4.3 s
Generator Manager	3924 / 676	7.1 s	7.9 s
Converter Manager	4305 / 2102	8.2 s	9.2 s
Pitch Manager	3943 / 545	7.0 s	7.9 s
Yaw Manager	3921 / 637	7.1 s	7.9 s

Analyzing the measures collected in the table, we see that the increase of the time required to load the model on a tablet instead of on a PC, ranges between the 11% and the 13% for model fragments, and goes up to the 20% when the full model is loaded.

The target measures and evaluation criteria for this indicator were:

- **Sufficient** — No more than 25% increase.
- **Good** — No more than 15% increase.
- **Excellent** — No more than 10% increase.

Thus, the results achieved for this measure are **sufficient**.

QN2 *– Number of concurrent users working with a model —* The number of concurrent users working with a WTCS model has been evaluated using the *Online Concurrent WTCS Modeling Solution*. Note that in this context, "number of concurrent users" specifically refers to an online collaboration scenario, whereby this number only includes those simultaneously displaying and editing the same model (or rather access-controlled views of it) via thin clients, i.e. without having local copies. Beyond this group of users who share a common collaboration session, there can of course be many more users overall, each with their own offline copies of the model, not having to be constantly connected to (and taking up resources in) the collaboration server.

To see whether the solution meets the most ambitious target measure set by industrial needs, 5 different users participated in the scenario: *PrincipalEngineer*, *IOManager*, *PitchManager*, *ConverterManager* and *GeneratorManager*. The five users have been working in the same shared modeling session. Thus every change made by one of the users has been automatically propagated to all other users, and their editors have been automatically refreshed to show the updated model.

The target measures and evaluation criteria for this indicator were:

- **Sufficient** — 2 concurrent users.
- **Good** — 3 concurrent users.
- **Excellent** — 5 or more concurrent users.

Since five users have been able to concurrently edit the model, the result achieved for this measure is **excellent**.

QN3 – *Time for change propagation and notification among concurrent users* — The time needed for change propagations and notification has been evaluated using the *Online Concurrent WTCS Modeling Solution*. This solution has been executed by five different users: *PrincipalEngineer*, *PitchManager*, *ConverterManager*, *GeneratorManager* and *IOManager*. The first four were working with their desktop PCs, while the fifth one executed it on a tablet. Each time a single modification was made by one user, the time required for notifying and updating other users' models has been measured, obtaining the result that changes were propagated in less than one second.

The target measures and evaluation criteria for this indicator were:

- **Sufficient** — Propagate and notify in less than 5 seconds.
- **Good** — Propagate and notify in less than 3 seconds.
- **Excellent** — Propagate and notify in less than 1 second.

The obtained result is that less than one second is required to propagate the changes. Thus, we consider that the result achieved for this measure is **excellent**.

QN4 – *Maximum number of elements that can be displayed in a diagram* — The evaluation of the maximum number of elements displayed in a diagram has been carried out on a PC using the *Offline Graphical WTCS Modeling Solution*.

To carry out this evaluation, several diagrams having different number of elements have been tested with the *Offline Graphical WTCS Modeling Solution*. The different diagram types used for this evaluation are collected in Table 6.

The target measures and evaluation criteria for this indicator were:

- **Sufficient** — Display a diagram with 25 elements.

Table 6: Number of elements in diagram (QN4)

DIAGRAM TYPE	NUMBER OF ELEMENTS IN DIAGRAM
Small-size diagrams	5 to 15
Mid-size diagrams	25
Pitch diagram	196 (194 *CtrlUnits* + 2 *Subsystems*)
Converter diagram	452 (447 *CtrlUnits* + 5 *Subsystems*)

Table 7: Time required for diagram loading (QN5)

DIAGRAM SIZE	TIME
Diagram having 5 to 15 elements	< 2 s
Diagram having 25 elements	< 3 s
Pitch diagram (194 *CtrlUnits* + 2 *Subsystems*)	< 4 s
Converter Manager (447 *CtrlUnits* + 5 *Subsystems*)	< 7 s

- **Good** — Display a diagram with 50 elements.
- **Excellent** — Display a diagram with more than 50 elements.

Considering that the *Offline Graphical WTCS Modeling Solution* can successfully load diagrams having up to 450 elements, and that the edition capabilities remain functional with this model size, the results achieved for this measure are **excellent**.

QN5 – *Time for loading a diagram having 25 elements to be displayed* — The evaluation of the time required to load a diagram has been carried out on a PC using the *Offline Graphical WTCS Modeling Solution*. The diagrams used for this evaluation are those collected in Table 6, since they are the same diagrams we used for the evaluation of QN4.

The results of the evaluation are presented in Table 7, where the size of the diagrams together with the time required to load them are summarized.

The target measures and evaluation criteria for this indicator were:

- **Sufficient** — Load the diagram in 2 seconds.
- **Good** — Load the diagram in 1 second.
- **Excellent** — Load the diagram in less than 1 second.

Thus, considering the results of the evaluation carried out – which shows that more than two seconds are required to load and display a diagram having 25 elements – the first conclusion is that target measures set in evaluation plan are not met. Nevertheless, we must remark that results achieved are very close to the *sufficient* target results, as this diagram can be loaded and displayed in less than 3 seconds. Thus, we can consider the result good enough. It must also be taken into account that the largest diagram for the WTCS model used in this evaluation, having 452 elements to be displayed, is loaded in less than 7 seconds, which is also considered a good result. So, although the target result has not been met, the overall results achieved in this evaluation are considered **sufficient**.

QN6 – *Time for committing model changes* — The time needed for committing model changes to the repository has been evaluated using the *Offline Collaborative WTCS Modeling Solution*.

Table 8: Time required for committing model changes (QN6)

CHANGES	TIME
Add 1 *Subsystem*	22 s
Add 10 *Inputs*	24 s
Add 2 *MainSubsystems*, 8 *CtrlUnits*, 16 references to *Inputs*, 8 references to *Outputs*	26 s
Add 1000 *Inputs*, 1000 *Outputs*, 10 *MainSubsystems*, 40 *CtrlUnits*, 800 references to *Inputs*, 400 references to *Outputs*	29 s

When a commit operation is performed in an offline collaborative scenario, model changes are committed to the gold repository and are propagated then to the front repositories for the different engineer types who are working in collaboration. So, the overall time required for the full commit and model merging operation will depend on the number of front repositories. To make this indicator independent of the number of front repositories, the target measures were set considering a single front repository.

To carry out this evaluation, different sets of changes have been made on models with the aim of measuring whether the amount of changes significantly impacts on the time required to commit all the changes to the repository or not.

Results of the evaluation are shown in Table 8, where changes performed in the model are presented together with the time required to commit them.

The evaluation scenario comprises one gold repository and 10 front repositories. The times collected in Table 8 correspond to the overall times required to commit changes to the gold repository and to update all the front repositories with those changes.

The first conclusion after the evaluation is that the amount of changes to be committed to the repository has no significant impact on the time needed to commit all the changes.

The target measures and evaluation criteria for this indicator were:

- **Sufficient** — Commit changes in less than 5 seconds.
- **Good** — Commit changes in less than 3 seconds.
- **Excellent** — Commit changes in less than 1 second.

Taking into account that times collected in Table 8 are the overall times for committing changes and updating all the front repositories, we see that between two and three seconds are needed for each repository. Therefore we can conclude that results for this measure are **good**.

Nevertheless, keeping in mind – as explained earlier – that model merging was a hand-made and error prone activity which could take up to half an hour, the improvement obtained thanks to MONDO technology which automates the merging process is in any case excellent.

QN7 – *Performance impact caused by the MONDO Collaboration Framework* — The impact on performance has been evaluated using the *Offline Collaborative WTCS Modeling Solution*. No requirement about performance degradation was defined in the initial project requirements, nor was any indicator defined in the evaluation plan in this regard. However, we consider that it is worth mentioning that the solution designed and developed for offline collaboration does not penalize the performance of features that were already available in the solution built without MONDO technology, which are also used in this *Offline Collaborative WTCS Modeling Solution*.

Therefore, an engineer will get the same performance for the modeling activities they were used to, but they will now have collaborative modeling capabilities not available prior to the MONDO project.

The target measures and evaluation criteria for this indicator were:

- **Sufficient** — Less than 5% of performance loss.
- **Good** — Less than 2% of performance loss.
- **Excellent** — Less than 1% of performance loss.

In this sense the conclusion is that results for this measure are **excellent**.

QN8 – *Time reduction for building graphical domain specific modeling editors* — The process of construction of the *Offline Graphical WTCS Modeling Solution* has been considered for the evaluation of time reduction for building graphical domain specific modeling editors.

The aim of this measure was to compare the time required to create a graphical WTCS modeling editor using the MONDO technology (specifically, DSL-tao and its design patterns) and the time required to create the graphical modeling editor using the design tool provided by Sirius.

A graphical WTCS modeling editor has been constructed by an expert using the Sirius graphical specification design tool in two hours.

An equivalent graphical modeling editor has been constructed using DSL-tao – as explained in Section 5.3 in no more than half an hour.

The target measures and evaluation criteria for this indicator were:

- **Sufficient** — 25% of reduction in time.
- **Good** — 50% of reduction in time.
- **Excellent** — 75% of reduction.

The time required to create the editor using DSL-tao is one fourth of time required to create it using the Sirius design tool, so the result achieved is **excellent**.

Quantitative Measures Summary — For the sake of understandability, Table 9 summarizes the rates achieved in

Table 9: Quantitative measures Results

ID	RATING
QN1	Sufficient
QN2	Excellent
QN3	Excellent
QN4	Excellent
QN5	Sufficient
QN6	Good
QN7	Excellent
QN8	Excellent

the different quantitative measures according to the criteria specified in the different experiments and also summarized in Table 1.

6.3.2 Qualitative Measures

Besides the quantitative measures presented above, a set of qualitative measures were also planned. The qualitative evaluation was carried out by three engineers participating in the evaluation playing six different roles in different moments of the experience (see Table 3 for such roles). The data for the quantitative analysis was gathered based on their experiences while performing the experiments. In their answers, respondents had the opportunity to provide additional comments and clarifications regarding their assessment.

The fulfillment of this set of measures is explained below, in Table 10, where the level of compliance for each measure is presented along with their comments. The level of compliance is set using a four point scale with the following values: *(i) fully*, the expected target measure has been achieved for the Wind Power domain; *(ii) largely*, although the target measure has not been fully been reached, the achieved result is very close to the expected result; *(iii) partially*, some interesting results have been achieved for the Wind Power domain thanks to the MONDO technology, although the target measure has not been reached; and *(iv) none*, no result related to the target measure has been obtained with the MONDO technology.

6.3.3 Scenario Coverage

Table 11 summarizes the relationship among the evaluation scenarios described in Section 6.1, the MONDO solutions used for each scenario, and the measures evaluated.

6.3.4 Threats to validity

In this section, we discuss the main threats to the validity of the approach and methodology used, as well as those of the results of the experiment itself. Our work is subjected to a number of threats to validity, namely: *(i)* internal validity, which is related to the inferences we made; and *(ii)* external validity, which discusses the generalization of our approach/findings.

Regarding the internal validity, our threats are mainly associated with the empirical validation and the measurements that were taken as part of it. In particular, experiments and metrics were defined based on the experience and estimated needs of Ikerlan which introduces a bias to the validation process. Nevertheless, to mitigate this risk, the rest of the partners reviewed the metrics to check they were reasonable and aligned with their own perceptions. A second threat to internal validity is that the engineers involved may have been positively biased towards the MONDO project. This risk was mitigated by the need to write and report objectively in the project deliverables, which were reviewed (both off-line and on-line via a demo) by the project officers.

Regarding the external validity, our threats are related to our focus on a single industrial case study and the constraints it imposed in the initial selection of frameworks to choose from. For example, the time reduction for building graphical domain specific modeling editors was validated using one DSL, but results may depend on the editor complexity, and the expertise of the engineers involved. Therefore, the results of this experiment should not be generalized to the global population of modeling tools and technologies even if our personal experience suggests that indeed at least some of the problems and solutions part of the MONDO framework are useful in a large variety of modeling scenarios. Another threat was the number of participants in the experiment (tens of users) and the models used in it (thousands of elements). An experiment orders of magnitude larger than these could have reflected a different outcome. Nevertheless, the size of the experiment was realistic for Ikerlan's needs in terms of Wind Turbine modeling.

Furthermore, since this use case uses brownfield development, it focuses on leveraging existing domain-knowledge and avoiding recreation of the solution, as much as possible, by extending the current tool-set instead of replacing it. It also limits itself to using open-source tools, and hence can only be generalizable to cases where similar requirements are present.

7 Related Work

In this section, we analyze related work concerning the systematic development of scalable graphical DSLs (Section 7.1), scalable modeling (Section 7.2), and access control (Section 7.3). We also note the relevant features that are unavailable in the state-of-the-art, which resulted to the need for a different solution, in this case using the MONDO platform.

Table 10: Qualitative Measures Results

ID	FULFILLMENT	COMMENTS
QL1	Fully	DSL-* tools provide a step by step process for designing large DSLs.
QL2	Fully	The tool supporting large DSL construction is DSL-tao. It provides a set of design patterns to design the DSL and to build its modeling tool.
QL3	Largely	A functional domain specific modeling tool can be created using DSL-tao, but collaboration features are not fully supported.
QL4	Largely	Components like the MONDO Collaboration Framework are ready to use. Setting up of the collaboration environments, however, should be automated.
QL5	Fully	This feature is provided by MONDO Online Collaboration Framework, which has been used to build the *Online Concurrent WTCS Modeling solution*.
QL6	Fully	This feature is provided by the MONDO Collaboration Framework. EMF-Splitter provides also this feature, enabling to split a model into different physical files that can be loaded separately on demand.
QL7	Fully	Progressive loading can be achieved by EMF-Splitter where each model fragment can be loaded on demand, when modularity pattern is applied.
QL8	Fully	Although this requirement has not been validated in the previous use cases, the MONDO technology has been tested to confirm that there is no constraint to combine two different modeling languages in a single modeling tool.
QL9	Fully	The MONDO Collaboration Framework allows users to edit a model concurrently using a web modeling application run on a tablet.

Table 11: Scenarios, solutions and evaluated measures

SCENARIO	MONDO SOLUTION	MEASURE ID
S1	Online Concurrent	QN2, QN3, QL5, QL6, QL4
S1	Offline Collaborative	QN4, QN5, QN6, QN7, QN8, QL1, QL2, QL3, QL4, QL6, QL7, QL8
S2	Offline Collaborative	QN6, QN7, QL1, QL2, QL3, QL4, QL6
S3	Online Concurrent	QN1, QL4, QL9

7.1 Systematic development of graphical DSLs

Today's development of external DSLs normally relies on language workbenches, which facilitate the DSL design task. Frequently, these tools promote a model-driven approach to DSL development, allowing for automation in the development process. On top of these tools, several authors have proposed techniques, processes and guidelines to make the development of DSLs more systematic and repeatable [53, 90, 55, 103, 65].

For example, in [52] the authors identify worst practices for DSL development, like the lack of involvement of domain experts, or lack of domain understanding. Conversely, Volter [103] identifies best practices for model-based development, including limiting the expressiveness of the DSL, defining notations that fit the domain, and support for model partitioning, among many others.

In [90] the authors suggest explicit processes for developing DSLs, including building a mock-up language when the domain is complex, and proposing a special process to extract the DSL from an existing system. In [65] the authors identify recommendations to help in the decision, analysis, design, implementation and deployment phases of DSLs. At the design level, for languages based on existing notations, they recommend either to piggyback DSL features on the existing language; specialize or extend the language.

At a more technical level, some works have proposed patterns for building DSLs [18, 83, 87]. However, while DSL-tao proposes domain, design, concrete syntax, dynamic semantics and infrastructure patterns, these works typically focus only on one or two kinds of patterns.

Regarding design patterns, Cho et al. [18] propose a set of metamodeling design patterns, but their lack of support for variability limits their practical use. Schäfer et al. [83] also acknowledge the lack of proper engineering processes for building DSLs, and thus propose documenting DSL requirements using design patterns and use cases. Mernik et al. [65] review common patterns and phases for DSL development. While their proposals are for textual DSLs, our work is oriented to graphical DSLs. Finally, Spinellis [87] defines architectural patterns for DSL design. Altogether, design patterns in these works improve the inner quality of the DSL, by providing design guidelines akin to traditional object-oriented design patterns [37], but they are normally low-level

and provide less gain in productivity compared to domain and infrastructure patterns. Please note that in MONDO, in addition to tooling, we proposed methodologies for systematic engineering of DSLs, including specific notations for gathering requirements for DSLs (called DSL-maps), and automated transition into a metamodel design [81].

Domain patterns capture domain knowledge and are useful to automate the construction of metamodels. Pedro et al. [79] proposed building DSLs by composing domain concepts. The latter are metamodels and their semantics are given by model transformations. The composition of concepts entails the composition of the respective transformations. Our domain patterns are located at a higher metalevel, which allows a more flexible instantiation. In addition, we support combined pattern application, variants, and the component-based construction of the DSL environment.

Some authors have proposed the use of methods from software product lines to facilitate the configuration of components and the composition of DSLs out of smaller parts. For example, White et al. [104], propose equipping DSLs with a feature model, where a configuration produces a metamodel variant. In the *grammarware* technical space, Neverlang [101] supports building textual languages from so called slices, which are made of fragments of the concrete syntax (a textual grammar) and semantics (an evaluator). Slices can be combined using a feature model. In our approach, we provide further flexibility by the instantiation of roles, and support the synthesis of modeling environments by attaching services to patterns.

Many graphical DSL workbenches have been proposed for different applications, like meta-CASE tools [53], diagram sketching [15], or multi-formalism modeling and simulation [62]. The popularity of Eclipse has promoted frameworks to create graphical editors as plugins, like Tiger [12], GMF [42], Eugenia [58], Spray [88], Graphiti [46], or Sirius [86]. Some other approaches, like WebGME [64] or AToMPM [20] are based on the web browser.

All these tools use a model-based approach to specify the concrete syntax, except Graphiti, which requires manual programming using a Java API. Some of them are based on code generation for other lower-level approaches. For example, Eugenia relies on GMF, and Spray generates code for Graphiti. In our case, the editors produced by DSL-tao rely on Sirius. All these frameworks rely on code generation except Sirius, which is interpreted. The way of specifying the concrete syntax varies: Eugenia requires annotating the metamodel elements, Spray uses a textual DSL, GMF and Sirius require building models that describe the graphical syntax, and Graphiti requires Java programming. Conceptually, our approach is closer to Eugenia, as pattern applications result in metamodel annotations. However, DSL-tao is based on wizards and heuristics, and supports attaching concrete syntax styles to domain patterns, which improves the productivity in the creation of graphical environments. This feature, and the ability to specify tool services via infrastructure patterns, are unique among the mentioned tools. In particular, none of these tools is able to produce graphical environments with model fragmentation capabilities.

Graphical DSLs are often evaluated according to some established criteria in the visual languages community, like Moody's "physics" of notations [69], or the cognitive dimensions of notations framework [47]. The goal is to evaluate and ensure that the graphical concrete syntax designed for a DSL possesses adequate perceptual features, so that models can be easily understood and built. Some tools, like CEVINEDIT [45], provide assistance for some of Moody's criteria. In our case, providing support for engineering cognitive effective graphical syntaxes is left for future work.

Overall, compared to related works in this area, the contributions of the MONDO framework include an approach to facilitate the systematic creation of metamodels, and their associated graphical environments. This is realized by the use of patterns (of five different types) and heuristics.

7.2 Scalable persistence

Different approaches have been proposed to process large models, fragmentation of models being a popular solution solving many of the challenges faced. For instance, Scheidgen and Zubow [84] propose the EMF-Fragments persistence framework, which supports automatic and transparent fragmentation to add, edit and update EMF models. To guide the fragmentation, the composition references in the metamodel that are aimed at producing fragments need to be annotated. EMF-Fragments stores fragmented models in memory, and for persistence primarily relies on distributed file-systems and key-value stores like MongoDB [67] and HBase [93].

The use of fragmentation techniques have also been used to improve the comprehensibility of models. For example, Kelsen et al. [54] propose an algorithm to fragment a model into submodels (actually a lattice of submodels), where each submodel is conformant to the original metamodel. The algorithm considers cardinality constraints but not general OCL constraints, and there is no tool support. Other works use Information Retrieval (IR) algorithms to split a model based on the relevance of its elements [91]. This research resulted in the creation of *Splittr* as a tool to split models. Therefore, splitting models that belong to the same metamodel can produce different structures. Customizable graph clustering techniques, with the purpose of metamodel modularization, have also been proposed [92]. The techniques are based on several clustering algorithms operation on a distance matrix. Such matrix is obtained by weighting different metamodel relations (generalization, composition, association) according to their relevance. While *EMF-Splitter* is applicable to existing large models, a distinctive feature of this solution

is that we also generate a modeling environment that enforces the defined modularization strategy when creating a new model.

Different (open-source, proprietary, and/or academic) model repositories have been proposed in the last few years to improve scalability and offering different services [82]. Proprietary model repositories such as No Magic's Teamwork Server [72] have been developed, providing versioned collaborative development of models; they offer model-element-level versioning and querying, supporting concurrent users accessing the same model. Nevertheless, these systems are highly-coupled with their modeling tools, having limited flexibility, due to binding the user to a specific modeling technology.

Similarly, open-source model repositories such as EMFStore [56] and CDO [30] have arguably gained little traction in industry, even though they commonly support a wide variety of back-end technologies and offer lower coupling with specific modeling tools. In our view, there are various valid reasons for this: from a user's point of view, adopting a model-specific version control system supported by a small open-source community for storing business-critical models is hard to support, when contrasted with robust and widely-used and supported file-based version control systems. Also, using two version control systems in parallel – such as having code in a Git repository and models in CDO – can introduce additional overhead and inconsistency risks, as code and models altered in the context of the same conceptual *commit*, are manually split and distributed amongst two unconnected systems.

Several model persistence technologies have been developed in the past years, as a scalable alternative to XMI – which is used in popular modeling technologies like EMF. Many of these, such as NeoEMF [9], Morsa [77], MongoEMF [16] and also EMF-Fragments [84] use NoSQL databases like Neo4J [70] or MongoDB as their back-ends, and often deliver promising results for model traversal and querying. On the other hand these systems do not handle version control of models stored in them, focusing on model management operations over a single conceptual version.

With respect to Hawk, further advancements have been made since then, including the ability to index the entire history of models in a single Hawk index. This work, a preliminary version of which has been already presented [39], allows for temporal queries spanning over arbitrary numbers of commits, and can enable cross-referencing or model comparison queries without the need to retrieve, load and query multiple different models.

Since all of these technologies were relevant to Ikerlan, a decision had to be made on which one to adopt. Using Hawk allowed Ikerlan to continue using their current models, in their original format, whilst gaining the advantages of scalable model querying. This meant that they did not have to change any of their current practices or tools in order to add Hawk to their technology stack, but instead only had to learn to use it as a technology orthogonal to their own. In the context of the MONDO stack as a whole, this meant that Hawk did not interfere with or add any further constraints to any of the other tools that aimed at supporting collaborative file-based modeling.

7.3 Fine-grained Access Control in Various Domains

In this section, we analyse different techniques for fine-grained access control, and their application in different technological spaces and domains.

As the following paragraphs will show in detail, none of the related approaches in the state of the art can readily fulfill the conjunction of all design goals of the *MONDO Collaboration Framework*, as stated in subsection 4.1.

File-based Access Control — Traditional version control systems (like CVS, SVN) and file sharing technologies adopt file-level access policies (or even more coarse-grained, such as branch/repository-level access control in Git), which are clearly insufficient for fine-grained access control specifications.

Off-the-shelf file systems typically require resources (files and folders) to be explicitly labeled with permissions that take the form of an *Access Control List* (ACL), or the simplified form user/group/other flags. An ACL consists of entries regarding which user/subject is granted or denied permission for a given operation.

File-based solutions can be directly applied to MBE, but cannot provide fine-grained access control, where different parts of a model file have different permissions. MONDO Access Control policies are fine-grained, use implicit rules – so that model elements do not have to be explicitly annotated with permission flags, which is difficult to manually maintain as the model evolves – and respect internal referential consistency; all the while being more flexible [24] in the conflict resolution method.

Access Control in RDF Triple/Quad stores — Graph-based access control is a popular strategy for many triple and quad stores (4store [40], Virtuoso [75], IBM DB2) developed for storing large RDF data. User privileges can be granted to for each named graph while access control is actually checked when issuing a SPARQL query. Denial of access for a graph filters the query results obtained from this specific graph. Data access in AllegroGraph [34] can be controlled on the database or catalog level (coarse-grained) as well as on the graph and triple level (fine-grained) while Stardog only allows database-level access control.

Similarly to our approach, fine-grained access control is discussed by Dietzold and Auer [28], who propose using

graph queries as preconditions of rules to select certain assets on which the permissions need to be enforced. The major difference is that we apply queries in an MBE environment (which has very important implications with respect to internal model consistency [24]), and we also provide offline collaboration.

In the Oracle Database Semantic Technologies [76], access control is carried out by default on the model (graph) level. Furthermore, it can be configured on the triple (row) level, which is implemented by query rewriting. In this case, the definition of access control policies is based on so-called match and apply (graph) patterns, where the former identifies the type of access restriction while the latter injects access-control specific constraints to the query. Again, due to the different requirements in MBE, our approach additionally supports internal model consistency [24]) and offline collaboration.

Another access control technique is called *label based security*, which offers *(i)* triple-level control using (a hierarchy of) sensitivity labels attached to each triple, and *(ii)* RDF resource-level access control for subject/predicate/object. Explicit data access labels are implemented by Oracle [76] and are generalized into abstract tokens and operators by Papakonstantinou et al. [78]. Again, due to the different requirements in MBE, our approach additionally supports internal model consistency [24]); it also operates with graph query-based polcies that do not require manual security label assignments.

Access Control for XML Documents — A number of standards such as XACML [43] (OASIS standard) provide fine-grained access control for XML documents. These type of documents are similar to models in a way, that they consists of nodes with attributes that may contain other nodes. XACML provides several combining algorithms to select from contradicting policies. Fundulaki and Marx [35] formalize fine-grained access control using XPath for XML documents, their work claims that the visibility of a node depends on its ancestors, thus when a node is granted access, then access is also granted to its descendants. However, other dependencies are not discussed related to XML Documents.

Similarly to our approach, a dedicated policy language is used by Montrieux and Hu [68], from which a transformation (lens) is automatically generated to enforce access control for XML documents. In addition to the attributes and context of the assets (XML nodes), the XQuery-based policy can take into account external (subject or context) attributes as well.

As the above techniques are not MBE approaches, there is no treatment of cross-references. There is no discussion of internal referential consistency either (except for the containment hierarchy, which is relevant for XML as well). Finally, there is no discussion of the challenges of online and offline collaboration. These are the major differentiating characteristics of our approach ahead of the state of the art.

Access control in Collaborative Modeling Environments — Currently, fine-grained access control is not considered in the state of the art tools of MDE such as MetaEdit+ [99], VirtualEMF [19], WebGME [64], EMFStore, GenMyModel / Web Modeling Framework [96], Obeo Designer Team [73], MDEForge [8] or the tools developed according to [36]. They either offer no low-level fine-grained access control, or do so without support for high-level specification of the access policy. See also the broader survey in [82].

The generic framework CDO (used for example in Obeo Designer [73], Eclipse Papyrus [31], PolarSys CHESS [33] or OpenCert [32]) provides both online collaboration and role-based access control with type-specific (class, package and resource-level) permissions, but no facility for instance level access control policy specifications. However, there is a pluggable access control mechanism that can specify access on the object level; it should be possible to integrate fine-grained solutions such as the currently proposed system.

The collaborative hardware design platform called VehicleFORGE stores their model in graph-based databases and has an access control scheme TrustForge [22] that uses an implementation of KeyNote [14] trust management system. This system is responsible for evaluating the request addressed to the database, which can be configured in various ways. It supports unlimited permission levels and it is also able to handle consistency constraints by adding them as assertions. Conflict resolution strategies are not discussed. AToMPM [20] provides fine-grained role-based access control for online collaboration; no offline scenario or query-based security is supported, though. Access control is provided at elementary manipulation level (RESTful services) in the online collaboration solution of [29].

As a substitute for developing custom DSL tools, UML-based tools could plausibly be adopted with the usage of UML profiles. While UML profiles offer a way to add domain knowledge to UML, it is difficult to create and use a concise DSL aimed at solving the problem at hand. Hiding irrelevant parts of the UML language and the user interface of modeling tools may be supported by certain vendors, but in our case, we have found it to likely require more work than creating a DSL from scratch. Furthermore, briefly surveying major collaborative UML tooling vendors, we have found no solutions that would satisfy our security needs. Neither GenMyModel [3] nor No Magic's Teamwork Server [72] and Teamwork Cloud [71] support fine-grained access control. Sparx EA [63] natively supports fine-grained (element-level) access control for write access to shared collaborative repositories, but there is no high-level specification method, and also no read access control. EA's access control features,

however, can be programmatically extended on the level of the storage backend.

8 Conclusions

Scalable modeling technologies can provide new opportunities for small and medium-sized organizations to grow their software development teams. In this paper we have reported on the experience at Ikerlan after implementing the technologies developed in the MONDO project. The experience has been extremely positive, and the evaluation shows that five out of eight quantitative measures scored excellent – one of them scored good and two others scored sufficient – while seven out of nine qualitative measures were fully fulfilled – the two remaining were largely fulfilled.

From this experience, we can also learn that **continuous compliance** with existing development processes is a key factor for success. The MONDO scalable technologies do not impose a big change on the processes and tools that were already implemented in the company. In this sense, the new solutions enable teamwork in the offline scenario in such a way that can be integrated without changing the pre-existing single-user modeling tools. This way developers continue working in the same way they used to work, and collaboration features only come into play to automate operations that were manual before (e.g., model merging).

Part of this success has been due to, not only the technology itself, but to the **methodological guidance** provided by MONDO. Specifically, the methodology supported by DSLtao can be easily followed to construct large scale DSLs. In this sense, it is important that this methodology provides a wide set of predefined design patterns, which DSL designers can take advantage of to build their custom modeling solutions.

Another important contribution of the scalable technology is the capability for **concurrent model editing using web technology**, enabling real-time collaboration with secure access control, even using mobile devices. While there are several emerging modeling frameworks to support web-based collaborative modeling (e.g., AToMPM, WebGME, Web Modeling Framework, etc.) security and scalability remains a major challenge for them. As demonstrated by *Online Graphical Collaborative WTCS Modeling Solution* the Eclipse RAP platform [95] is not mature enough.

Finally, this experience also evidences that web-based solutions are not best suited to carry out modeling activities in handheld mobile devices, since they present usability issues. In this sense, another possible avenue for research is the development of **dedicated domain-specific modeling environments for mobile devices** [102].

Acknowledgements This work has been supported by the MONDO (EU FP7-ICT-611125) project.

The work of Gábor Bergmann was also partially supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, the ÚNKP-18-4 New National Excellence Program of The Ministry of Human Capacities, and the ÚNKP-19-4 New National Excellence Program of the Ministry For Innovation and Technology.

Antonio's and Juan's work was also partially supported by the Spanish Ministry of Science (RTI2018-095255-B-I00) and the Madrid Region (S2018/TCS-4314).

Finally, would like to thank Ana Pescador, István Ráth, Dániel Varró, and all the MONDO researchers for their contributions to the project.

References

1. Ackermann, T., Söder, L.: Wind energy technology and current status: a review. Renewable and Sustainable Energy Reviews **4**(4), 315 – 374 (2000). DOI 10.1016/S1364-0321(00)00004-6
2. Atkinson, C., Kühne, T.: Rearchitecting the UML infrastructure. ACM Trans. Model. Comput. Simul. **12**(4), 290–321 (2002)
3. Axellience: GenMyModel.com. URL: `https://www.genmymodel.com/`, last accessed Aug. 2019
4. Barmpis, K., Kolovos, D.: Hawk: Towards a scalable model indexing architecture. In: Proceedings of the Workshop on Scalability in Model Driven Engineering, BigMDE '13, pp. 6:1–6:9. ACM, New York, NY, USA (2013). DOI 10.1145/2487766.2487771
5. Barmpis, K., Kolovos, D.S.: Hawk: towards a scalable model indexing architecture. In: Proceedings of the Workshop on Scalability in Model Driven Engineering, BigMDE '13, pp. 6:1–6:9. ACM, New York, NY, USA (2013)
6. Barmpis, K., Kolovos, D.S.: Towards scalable querying of large-scale models. In: Proceedings of the 10th European Conference on Modelling Foundations and Applications, pp. 35–50 (2014). DOI 10.1007/978-3-319-09195-2_3
7. Barmpis, K., Shah, S., Kolovos, D.S.: Towards incremental updates in large-scale model indexes. In: Proceedings of the 11th European Conference on Modelling Foundations and Applications (2015). DOI 10.1007/978-3-319-09195-2_3
8. Basciani, F., Rocco, J.D., Ruscio, D.D., Salle, A.D., Iovino, L., Pierantonio, A.: MDEForge: an extensible web-based modeling platform. In: CloudMDE@MoDELS (2014)
9. Benelallam, A., Gómez, A., Sunyé, G., Tisi, M., Launay, D.: Neo4emf, a scalable persistence layer for emf models. In: Modelling Foundations and Applications, pp. 230–241. Springer (2014)
10. Bergmann, G., Debreceni, C., Ráth, I., Varró, D.: Query-based access control for secure collaborative modeling using bidirectional transformations. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, October 2-7, 2016, pp. 351–361 (2016). DOI 10.1145/2976767.2976793
11. Bergmann, G., Debreceni, C., Ráth, I., Varró, D.: Towards efficient evaluation of rule-based permissions for fine-grained access control in collaborative modeling. In: 2nd International Workshop on Collaborative Modelling in MDE. Austin, Texas, USA (2017)
12. Biermann, E., Ehrig, K., Ermel, C., Hurrelmann, J.: Generation of simulation views for domain specific modeling languages based on the eclipse modeling framework. In: ASE, pp. 625–629. IEEE CS (2009)
13. Blaha, M.: Patterns of data modeling. CRC Press (2010)
14. Blaze, M., Keromytis, A.D.: The keynote trust-management system version 2 (1999)

15. Brieler, F., Minas, M.: A model-based recognition engine for sketched diagrams. J. Vis. Lang. Comput. **21**(2), 81–97 (2010). DOI 10.1016/j.jvlc.2009.12.002. URL http://dx.doi.org/10.1016/j.jvlc.2009.12.002
16. Bryan Hunt: MongoEMF. URL: https://github.com/BryanHunt/mongo-emf/, last accessed Nov. 2019
17. Cezo, J., Krueger, C.: Use product line engineering to reduce the total costs required to create, deploy & maintain systems & software. URL https://www.embedded.com/design/prototyping-and-development/4008186/Use-product-line-engineering-to-reduce-the-total-costs-required-to-create-deploy--maintain-systems--software
18. Cho, H., Gray, J.: Design patterns for metamodels. In: DSM (2011)
19. Clasen, C., Jouault, F., Cabot, J.: VirtualEMF: A model virtualization tool. In: Advances in Conceptual Modeling. Recent Developments and New Directions, pp. 332–335 (2011)
20. Corley, J., Syriani, E., Ergin, H.: Evaluating the cloud architecture of atompm. In: MODELSWARD 2016 - Proceedings of the 4rd International Conference on Model-Driven Engineering and Software Development, pp. 339–346. SciTePress (2016)
21. Czarnecki, K., Eisenecker, U.W.: Generative Programming: Methods, Tools, and Applications. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000). ISBN: 0-201-30977-7
22. DARPA VehicleFORGE, P.U.: TrustForge: Flexible Access Control for VehicleForge.mil Collaborative Environment (2012). URL http://cps-vo.org/node/6851
23. Debreceni, C., Bergmann, G., Búr, M., Ráth, I., Varró, D.: The mondo collaboration framework: Secure collaborative modeling over existing version control systems. In: 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. ACM, ACM, Paderborn, Germany (2017). DOI https://doi.org/10.1145/3106237.3122829
24. Debreceni, C., Bergmann, G., Ráth, I., Varró, D.: Deriving effective permissions for modeling artifacts from fine-grained access control rules. In: 1st International Workshop on Collaborative Modelling in MDE. ACM, ACM, Saint Malo, France (2016)
25. Debreceni, C., Bergmann, G., Ráth, I., Varró, D.: Enforcing fine-grained access control for secure collaborative modelling using bidirectional transformations. Software & Systems Modeling (2017). DOI 10.1007/s10270-017-0631-8. URL https://doi.org/10.1007/s10270-017-0631-8
26. Debreceni, C., Bergmann, G., Ráth, I., Varró, D.: Property-based locking in collaborative modeling. In: 20th International Conference on Model Driven Engineering Languages and Systems. Austin, Texas, USA (2017)
27. Debreceni, C., Ráth, I., Varró, D., Carlos, X.D., Mendialdua, X., Trujillo, S.: Automated Model Merge by Design Space Exploration. In: P. Stevens, A. Wasowski (eds.) Fundamental Approaches to Software Engineering - 19th International Conference, FASE 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings, *Lecture Notes in Computer Science*, vol. 9633, pp. 104–121. Springer (2016). DOI 10.1007/978-3-662-49665-7_7
28. Dietzold, S., Auer, S.: Access Control on RDF Triple Stores from a Semantic Wiki Perspective. In: C. Bizer, S. Auer, L. Miller (eds.) Proc. of 2nd Workshop on Scripting for the Semantic Web at ESWC, Budva, Montenegro, June 12, 2006, *CEUR Workshop Proceedings ISSN 1613-0073*, vol. 183 (2006). URL http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-181/paper3.pdf
29. Farwick, M., Agreiter, B., White, J., Forster, S., Lanzanasto, N., Breu, R.: A web-based collaborative metamodeling environment with secure remote model access. In: Web Engineering, 10th International Conference, ICWE 2010, Vienna, Austria, July 5-9, 2010. Proceedings, *LNCS*, vol. 6189, pp. 278–291. Springer (2010)
30. Foundation, T.E.: CDO Model Repository. URL: http://www.eclipse.org/cdo/, last accessed Nov. 2019
31. Foundation, T.E.: Eclipse Papyrus. URL: https://www.eclipse.org/papyrus/, last accessed Nov. 2019
32. Foundation, T.E.: OpenCert. URL: https://www.polarsys.org/projects/polarsys.opencert, last accessed Nov. 2019
33. Foundation, T.E.: PolarSys CHESS. URL: https://www.polarsys.org/projects/polarsys.chess, last accessed Nov. 2019
34. Franz, I.: AllegroGraph. http://franz.com/agraph/allegrograph/doc/security.html
35. Fundulaki, I., Marx, M.: Specifying access control policies for XML documents with XPath. In: 9th ACM Symposium on Access Control Models and Technologies, pp. 61–69 (2004)
36. Gallardo, J., Bravo, C., Redondo, M.A.: A model-driven development method for collaborative modeling tools. J. Network and Computer Applications **35**(3), 1086–1105 (2012)
37. Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley (1994)
38. Garcia-Dominguez, A., Barmpis, K., Kolovos, D.S., Wei, R., Paige, R.F.: Stress-testing remote model querying apis for relational and graph-based stores. Software & Systems Modeling (2017). DOI 10.1007/s10270-017-0606-9
39. García-Domínguez, A., Bencomo, N., Paucar, L.H.G.: Reflecting on the past and the present with temporal graph-based models. In: Proceedings of the 13th International Workshop on Models@run.time, at MODELS'18 (2018)
40. Garlik: 4store Access Control. http://4store.org/trac/wiki/GraphAccessControl
41. Garmendia, A., Guerra, E., Kolovos, D.S., de Lara, J.: EMF splitter: A structured approach to EMF modularity. In: Proc. XM@MODELS, *CEUR Workshop Proceedings*, vol. 1239, pp. 22–31. CEUR-WS.org (2014). URL: http://ceur-ws.org/Vol-1239/xm14_submission_3.pdf
42. GMF: https://wiki.eclipse.org/Graphical_Modeling_Framework
43. Godik, S., (eds), T.M.: eXtensible access control markup language (XACML) version 1.0. (2003)
44. Gómez, A., Mendialdua, X., Bergmann, G., Cabot, J., Debreceni, C., Garmendia, A., Kolovos, D.S., de Lara, J., Trujillo, S.: On the opportunities of scalable modeling technologies: An experience report on wind turbines control applications development. In: Modelling Foundations and Applications - 13th European Conference, ECMFA, *Lecture Notes in Computer Science*, vol. 10376, pp. 300–315. Springer (2017)
45. Granada, D., Vara, J.M., Bollati, V.A., Marcos, E.: Enabling the development of cognitive effective visual dsls. In: MoDELS, vol. 8767, pp. 535–551. Springer (2014)
46. Graphiti: http://eclipse.org/graphiti/
47. Green, T.R.G., Petre, M.: Usability analysis of visual programming environments: A 'cognitive dimensions' framework. J. Vis. Lang. Comput. **7**(2), 131–174 (1996)
48. Hutchinson, J., Rouncefield, M., Whittle, J.: Model-driven engineering practices in industry. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, pp. 633–642. ACM, New York, NY, USA (2011). DOI 10.1145/1985793.1985882. URL http://doi.acm.org/10.1145/1985793.1985882
49. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of mde in industry. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, pp. 471–480. ACM, New York, NY, USA (2011). DOI 10.1145/1985793.1985858. URL http://doi.acm.org/10.1145/1985793.1985858

50. IKERLAN: IKERLAN, Where technology is an attitude. URL: http://www.ikerlan.es/en/ikerlan/, last accessed Nov. 2019
51. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis feasibility study. Tech. rep., CMU-SEI (90)
52. Kelly, S., Pohjonen, R.: Worst practices for domain-specific modeling. IEEE Software **26**(4), 22–29 (2009)
53. Kelly, S., Tolvanen, J.: Domain-specific modeling - enabling full code generation. Wiley (2008). URL http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470036664.html
54. Kelsen, P., Ma, Q., Glodt, C.: Models within models: Taming model complexity using the sub-model lattice. pp. 171–185. Springer (2011)
55. Kleppe, A.: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels, 1 edn. Addison-Wesley Professional (2008). ISBN: 0321553454, 9780321553454
56. Koegel, M., Helming, J.: EMFStore: a model repository for EMF models. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 2, pp. 307–308. ACM (2010). DOI 10.1145/1810295.1810364
57. Kolovos, D.S., Paige, R.F., Polack, F.A.: The Epsilon Transformation Language. In: Proceedings of the 1st International Conference on Theory and Practice of Model Transformations, ICMT '08, pp. 46–60. Springer-Verlag, Berlin, Heidelberg (2008). DOI 10.1007/978-3-540-69927-9_4
58. Kolovos, D.S., Rose, L.M., bin Abid, S., Paige, R.F., Polack, F.A.C., Botterweck, G.: Taming EMF and GMF using model transformation. In: MODELS, *LNCS*, vol. 6394, pp. 211–225. Springer (2010)
59. Kolovos, D.S., Rose, L.M., Matragkas, N., Paige, R.F., Guerra, E., Cuadrado, J.S., De Lara, J., Ráth, I., Varró, D., Tisi, M., Cabot, J.: A research roadmap towards achieving scalability in model driven engineering. In: Proceedings of the Workshop on Scalability in Model Driven Engineering, BigMDE '13, pp. 2:1–2:10. ACM, New York, NY, USA (2013). DOI 10.1145/2487766.2487768
60. Kühn, T., Böhme, S., Götz, S., Aßmann, U.: A combined formal model for relational context-dependent roles. In: SLE, pp. 113–124. ACM (2015)
61. de Lara, J., Guerra, E., Cuadrado, J.S.: When and how to use multilevel modelling. ACM Trans. Softw. Eng. Methodol. **24**(2), 12:1–12:46 (2014)
62. de Lara, J., Vangheluwe, H.: Atom^3: A tool for multi-formalism and meta-modelling. In: FASE, *LNCS*, vol. 2306, pp. 174–188. Springer (2002)
63. Ltd., S.S.: Enterprise Architect. URL: https://www.sparxsystems.eu/enterprise-architect, last accessed Aug. 2019
64. Maróti, M., Kecskés, T., Kereskényi, R., Broll, B., Völgyesi, P., Jurácz, L., Levendovszky, T., Lédeczi, Á.: Next generation (meta)modeling: Web- and cloud-based collaborative tool infrastructure. In: Proceedings of the 8th Workshop on Multi-Paradigm Modeling (MPM@MODELS), *CEUR Workshop Proceedings*, vol. 1237, pp. 41–60. CEUR-WS.org (2014)
65. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37**(4), 316–344 (2005)
66. Modeliosoft: Modelio Open Source - UML and BPMN free modeling tool. URL: https://www.modelio.org/, last accessed Nov. 2019
67. MongoDB Inc.: MongoDB. URL: https://www.mongodb.com, last accessed Nov. 2019
68. Montrieux, L., Hu, Z.: Towards attribute-based authorisation for bidirectional programming. In: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, SACMAT '15, pp. 185–196. ACM, New York, NY, USA (2015). DOI 10.1145/2752952.2752963. URL http://doi.acm.org/10.1145/2752952.2752963
69. Moody, D.L.: The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Software Eng. **35**(6), 756–779 (2009)
70. Neo4j, Inc.: Neo4J Graph Platform. URL: https://neo4j.com/, last accessed Nov. 2019
71. No Magic, I.: Teamwork Cloud. URL: https://www.nomagic.com/products/teamwork-cloud, last accessed Aug. 2019
72. No Magic, I.: Teamwork Server. URL: https://www.nomagic.com/products/teamwork-server, last accessed Aug. 2019
73. Obeo: Obeo designer team. https://www.obeodesigner.com/en/collaborative-features
74. OMG: XML Metadata Interchange (XMI), Ver. 2.5.1. https://www.omg.org/spec/XMI/2.5.1/
75. OpenLink Software: OpenLink Virtuoso. URL: https://virtuoso.openlinksw.com, last accessed Nov. 2019
76. Oracle: Database Semantic Technologies. http://docs.oracle.com/cd/E11882_01/appdev.112/e11828/fine_grained_acc.htm
77. Pagán, J.E., Cuadrado, J.S., Molina, J.G.: A repository for scalable model management. Software & Systems Modeling **14**(1), 219–239 (2013). DOI 10.1007/s10270-013-0326-8
78. Papakonstantinou, V., Michou, M., Fundulaki, I., Flouris, G., Antoniou, G.: Access control for RDF graphs using abstract models. In: 17th ACM Symposium on Access Control Models and Technologies, SACMAT '12, Newark, NJ, USA - June 20 - 22, 2012, pp. 103–112. ACM (2012)
79. Pedro, L., Buchs, D., Amaral, V.: Foundations for a domain specific modeling language prototyping environment: A compositional approach. In: DSM (2008)
80. Pescador, A., Garmendia, A., Guerra, E., Cuadrado, J.S., de Lara, J.: Pattern-based development of Domain-Specific Modelling Languages. In: 18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MoDELS, pp. 166–175. IEEE Computer Society (2015). DOI 10.1109/MODELS.2015.7338247
81. Pescador, A., de Lara, J.: Dsl-maps: from requirements to design of domain-specific languages. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE, pp. 438–443 (2016)
82. Rocco, J.D., Ruscio, D.D., Iovino, L., Pierantonio, A.: Collaborative repositories in model-driven engineering [software technology]. IEEE Software **32**(3), 28–34 (2015). DOI 10.1109/MS.2015.61
83. Schäfer, C., Kuhn, T., Trapp, M.: A pattern-based approach to DSL development. In: DSM@SPLASH, pp. 39–46. ACM (2011)
84. Scheidgen, M., Zubow, A.: Map/reduce on emf models. In: Proceedings of the 1st International Workshop on Model-Driven Engineering for High Performance and Cloud Computing, MDHPCL '12, pp. 7:1–7:5. ACM, New York, NY, USA (2012). DOI 10.1145/2446224.2446231
85. Selic, B.: The pragmatics of model-driven development. IEEE Software **20**(5), 19–25 (2003). DOI 10.1109/MS.2003.1231146
86. Sirius: https://eclipse.org/sirius/
87. Spinellis, D.: Notable design patterns for domain-specific languages. Journal of Systems and Software **56**(1), 91–99 (2001)
88. Spray: https://code.google.com/a/eclipselabs.org/p/spray/
89. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework 2.0, 2nd edn. Addison-Wesley Professional (2009). ISBN: 0321331885
90. Strembeck, M., Zdun, U.: An approach for the systematic development of domain-specific languages. Softw., Pract. Exper. **39**(15), 1253–1292 (2009)
91. Strüber, D., Rubin, J., Taentzer, G., Chechik, M.: Splitting models using information retrieval and model crawling techniques. pp. 47–62. Springer (2014)

92. Strüber, D., Selter, M., Taentzer, G.: Tool support for clustering large meta-models. In: BigMDE 2013, p. 7. ACM (2013)
93. The Apache Software Foundation: Apache HBase. URL: http://hbase.apache.org/, last accessed Nov. 2019
94. The Eclipse Foundation: Eclipse - The Eclipse Foundation open source community website. URL: https://eclipse.org/, last accessed Nov. 2019
95. The Eclipse Foundation: Remote Application Platform (RAP). URL: http://eclipse.org/rap/, last accessed Nov. 2019
96. The Eclipse Foundation: Web Modeling Framework (previously genmymodel.com). URL: https://projects.eclipse.org/proposals/web-modeling-framework/, last accessed Nov. 2019
97. The MONDO Project: Work Package 4 – Scalable Collaborative Modelling. Deliverable 4.4: Prototype Tool for Collaborative Modeling (2016). URL: http://hdl.handle.net/20.500.12004/1/P/MONDO/D4.4
98. Tolone, W., Ahn, G.J., Pai, T., Hong, S.: Access control in collaborative systems. ACM Comput. Surv. **37**(1), 29–41 (2005). DOI 10.1145/1057977.1057979
99. Tolvanen, J.: MetaEdit+: Domain-specific modeling and product generation environment. In: Software Product Lines, 11th Int. Conf. SPLC 2007, Kyoto, Japan, pp. 145–146 (2007)
100. Trujillo, S., Garate, J.M., Lopez-Herrejon, R.E., Mendialdua, X., Rosado, A., Egyed, A., Krueger, C.W., de Sosa, J.: Coping with variability in model-based systems engineering: An experience in green energy. In: T. Kühne, B. Selic, M.P. Gervais, F. Terrier (eds.) Modelling Foundations and Applications, pp. 293–304. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
101. Vacchi, E., Cazzola, W., Pillay, S., Combemale, B.: Variability support in domain-specific language development. In: SLE, *LNCS*, vol. 8225, pp. 76–95. Springer (2013)
102. Vaquero-Melchor, D., Palomares, J., Guerra, E., de Lara, J.: Active domain-specific languages: Making every mobile user a modeller. In: 20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS, pp. 75–82. IEEE Computer Society (2017)
103. Völter, M.: Md* best practices. Journal of Object Technology **8**(6), 79–102 (2009)
104. White, J., Hill, J.H., Gray, J., Tambe, S., Gokhale, A.S., Schmidt, D.C.: Improving domain-specific language reuse with software product line techniques. IEEE Software **26**(4), 47–53 (2009)

Abel Gómez is a researcher of the *Internet Interdisciplinary Institute*, a research center of the *Universitat Oberta de Catalunya*, Spain. Previously, he has hold different positions at the *Universidad de Zaragoza*, the *École des Mines de Nantes & Inria*, and the *Universitat Politècnica de València*; being this latter institution where he obtained his PhD degree in Computer Science.

His research interests fall in the broad field of Model-Driven Engineering (MDE), and his research lines have evolved in two complementary directions: on the one hand, the development of core technologies to support MDE activities; and on the other hand, the application of MDE techniques to solve Software Engineering problems. More information is available at https://abel.gomez.llana.me.

Xabier Mendialdua is a researcher of the *Dependable Software Team* at *Ikerlan S. Coop.* Since 1994 he has been involved in multiple projects in several domains as automated warehouse management systems, monitoring systems, wind power and railway. His main activity is focused in the design and development of dependable software for embedded control systems and his research activities have been involved on applying model-driven and product-line approaches for the engineering of control systems.

Dr. Konstantinos Barmpis is a Research Associate in the Department of Computer Science at the University of York, in the field of MDE (Model-Driven Engineering), DSLs (Domain-Specific Languages), Distributed Systems and Repository Mining, as part of the CROSSMINER project funded by the European Commission for the H2020 initiative. In addition to 19 papers in peer-reviewed journals, conferences and workshops, he has been one of the creators and main contributors of the Hawk open-source project on scalable model indexing, as well as the CROSSFLOW open-source project on parallel and distributed heterogeneous analysis workflow creation and execution. He holds a doctorate of engineering (EnGD) from the University of York, in large-scale complex IT systems. More information is available at https://www-users.cs.york.ac.uk/~kb.

Gábor Bergmann is an assistant professor at the Budapest University of Technology and Economics, a research fellow at the MTA-BME Lendület Research Group on Cyber-Physical Systems, as well as co-founder and MDE Expert at IncQueryLabs Ltd. His main research interest is model-driven systems engineering with a special focus on model queries. He has received his PhD in software engineering from Budapest University of Technology and Economics. Contact him at bergmann@mit.bme.hu.

Jordi Cabot received the B.Sc. and Ph.D. degrees in computer science from the Technical University of Catalonia. He was a Leader of an INRIA and LINA Research Group at École des Mines de Nantes, France, a Post-Doctoral Fellow with the University of Toronto, a Senior Lecturer with the Open University of Catalonia, and a Visiting Scholar with the Politecnico di Milano. He is currently an ICREA Research Professor at Internet Interdisciplinary Institute.

His research interests include software and systems modeling, formal verification and the role AI can play in software development (and vice versa). He has published over 150 peer-reviewed conference and journal papers on these topics. Apart from his scientific publications, he writes and blogs about all these topics in several sites. He is a member of the IEEE and the ACM.

Xabier De Carlos is a researcher in *Data Analytics and Artificial Intelligence* at *Ikerlan S. Coop.*, with skills in model driven engineering, embedded systems development, databases and development of data visualization and moni-

toring applications for industrial domains. His PhD dissertation focused on Model Driven Engineering and Databases. Xabier has participated in public FP7 and H2020 projects funded by the European Commission, and also in industrial projects.

Csaba Debreceni recently received his Ph.D. in Computer Engineering from the Budapest University of Technology and Economics. He is currently a Senior Researcher at IncQuery Labs Ltd. His main research interest is model-driven development especially on collaborative modeling.

Antonio Garmendia has a Ph.D. in Computer and Telecommunication Engineering from the Universidad Autónoma in Madrid. As a Ph.D. student, he made a research visit to the Philipps-University Marburg (Germany). He is a member of the "Modelling and Software Engineering" research group (`http://www.miso.es`) at UAM. His research interests are in scalability in Model-Driven Engineering (MDE) and the construction of graphical modeling environments. He has participated in the MONDO EU project on scalability in MDE.

Dimitris Kolovos is a Professor of Software Engineering in the Department of Computer Science at the University of York, where he researches and teaches automated and model-based software engineering. He is also an Eclipse Foundation committer, leading the development of the open-source Epsilon model-based software engineering platform, and an associate editor of the Software and Systems Modelling journal. He has co-authored more than 150 peer-reviewed papers and his research has been supported by the European Commission, UK's Engineering and Physical Sciences Research Council (EPSRC), InnovateUK and by companies such as Rolls-Royce and IBM.

Juan de Lara is full professor at the Universidad Autónoma of Madrid, where he leads the modelling and software engineering research lab (`http://miso.es`) together with Esther Guerra. His main research interests are in Model-driven Engineering, including meta-modelling, model transformations, flexible modelling and domain-specific languages. He has published more than 200 papers in international journals and conferences and has been the PC co-Chair of ICMT12, FASE12, and ICGT17. He is associate editor of the Journal on Software and Systems Modeling, JOT and IET Software.