On the Need for Intellectual Property Protection in Model-Driven Co-Engineering Processes

Salvador Martínez[1], Sebastien Gerard[1] and Jordi Cabot[2]

[1] CEA-LIST Paris, France
{salvador.martinez,sebastien.gerard}@cea.fr
[2] ICREA-UOC Barcelona, Spain
jordi.cabot@icrea.cat

Abstract. We live in an increasingly complex world where all systems tend to include heterogeneous and interconnected components. To cope with these systems, industry is shifting towards co-engineering development processes where partners with very different roles and access needs must collaborate together. Therefore, protecting the intellectual property (IP) of the shared assets is a must. Model-Driven Engineering (MDE) may play a key role in the successful enactment of industrial co-engineering processes but only if it succeeds at integrating at its core the concern for IP protection, that has been up to the date largely ignored. In order to advance in this direction, we provide in this paper an initial roadmap towards the holistic protection of IP in collaborative modeling scenarios and we discuss how existing technologies such as Cryptography, Access-control (AC) or Digital Rights Management (DRM) are adapted and integrated in a framework for IP protection in the MDE.

1 Introduction

Systems are becoming more and more complex every day and often integrate IoT components, AI, Big data, and other heterogeneous subsystems. As a result, outsourcing parts of the system design process becomes a necessity. In a classical supply chain, an original equipment manufacturer (OEM) shares parts of its design artefacts with its direct suppliers (tier 1) so that they can perform their tasks. In turn, tier 1 suppliers may also share their artefacts with their own suppliers (tier 2) and so on. This is also becoming true for *digital* assets in what constitutes a paradigm shift towards the co-engineering of systems design.

Model-Driven Engineering (MDE), due to its capacity to specify and reason on digital assets at a high abstraction level, can become a key enabler of this industrial co-engineering shift by helping all participants to contribute to a common project using different (modeling) languages and tools depending on their technical profile and level of expertise. However, for this to happen, MDE should integrate at its core the concerns for Intellectual Property (IP) protection that so far have been largely ignored.

A modeling project is typically composed of models and metamodels [3] but also of transformations, queries and constraints on those (meta)models. All of them must be protected in such a way that only what is strictly necessary for a given task is shared with the relevant partners.

Nevertheless, and no matter the strength of the security mechanisms in place, the event of an IP leakage from authorized parties may not be discarded. Thus, an effective IP protection system for a collaborative modelling framework must also deal with such events by providing mechanisms to deal with stolen IP.

These aspects are a major hurdle for any collaborative modeling project. The French Alternative Energies and Atomic Energy Commission (CEA) has struggled with this challenge when trying to set up collaborations with partners in projects involving MDE and where, due to industrial confidentiality agreements, it has to ensure the disclosure of the full details of the models or had to ensure the IP preservation. This was the initial motivation of this work.

To advance in this direction, we provide in this paper an initial roadmap towards the holistic protection of IP in collaborative modelling scenarios. Partial approaches based on access-control [21,13], cryptography [5] and/or encapsulation (such as Functional Mock-up Units (FMUs)) [7] have been explored in the past. We go beyond these approaches by pushing for the first unified solution including advanced access-control, cryptography and digital rights management (DRM) techniques for collaborative modeling.

The rest of the paper is organized as follows. Section 2 describes existing mechanism successfully used in other domains for the protection of digital assets. Then, section 3 details how we propose to adapt them to the modeling technical space and enable their integration in an IP protection framework for model-based collaborative development. Finally, Section 4 outlines a working plan to achieve this goal and Section 5 presents our conclusions and further work.

2 IP Protection Mechanisms

A number of different mechanisms have been successfully employed to protect digital assets. They are clear candidates to be part of an IP protection framework for MDE. Here we summarize the main “families” of (complementary) approaches we will reuse and discuss the, very few, previous attempts on porting them to the MDE realm.

– Authorization policies are a mechanism to assign permissions, e.g. the capability of performing actions on resources within a system, to users depending on certain conditions like role and time. Many different access-control paradigms exist such as Role-Based Access-Control (RBAC) [8] or Attribute-Based Access-Control (ABAC) [22]. Privacy policies are conceptually similar but add conditions on the intent and purpose for the data to be shared to protect the user privacy.

[3] Metamodels define the abstract syntax of a modeling language, i.e. they specify the elements that can appear in a model and how they can be related to each other. Metamodels are usually represented as models themselves.

- Cryptography is a mechanism to assure the secrecy of information by converting ordinary information into unintelligible information. Typically, cryptography relies on the use of keys and on two encryption and decryption functions. Obfuscation is a similar concept.
- Digital Rights Management (DRM) refers to the mechanisms used to restrict the usage of proprietary resources with the purpose of preventing IP to be shared freely. It involves access-control mechanisms such as *pay-per-view* or *membership-based* content access **but also mechanisms intended to provide prosecution evidence for legal purposes.** The former can be regarded as part of the "Authorization policies" family mentioned above. The latter are typically implemented via digital watermarking and fingerprinting technologies. *Watermarking* consists in embedding in an object imperceptible marks that can be subsequently used to determine IP ownership. *Fingerprinting*, unlike watermarking, does not involve the modification of a resource. It relies on the identification of *unique* features of a resource so that its *fingerprint* can be stored by a trusted arbiter to resolve potential IP conflicts.

MDE has been largely employed to model (and generate) security aspects of the software artefacts to be developed [17] but very few works focus on protecting MDE artefacts themselves and no other approach tries to combine several in one MDE IP framework as we propose. First steps towards providing access-control for model artefacts are proposed by Debreceni et al. [6], whereas obfuscation has been explored by Fill [9]. Industry-wise, popular tools such as such as CDO [19] and Viatra[20] provide, respectively, model access-control and obfuscation. Some of our own works will also be adapted in the building of this framework as we will see next.

3 IP Protection Framework for MDE

Our framework requires two main components (Figure 1): 1) a *design time* component to enable the specification of the **IP protection policy** of the collaborative project and 2) a *runtime* component in charge of **evaluating and enforcing that policy** upon model access and manipulation requests from users during the collaborative modeling process.

We use the term *megamodel* [2] to refer to the whole set of interrelated modeling artefacts involved in the project. This includes all kinds of model manipulation operations (like model transformations) that from a MDE perspective can also be regarded as models themselves. This allows for a uniform treatment of all artefacts.

3.1 IP Protection policy setup

The IP protection policy is expressed via an IP policy language that integrates the definition of **classical authorization** constraints plus advanced **conditions** to refine those authorizations and additional **obligations** the IP protection mechanism must satisfy before delivering the requested modeling artefact.

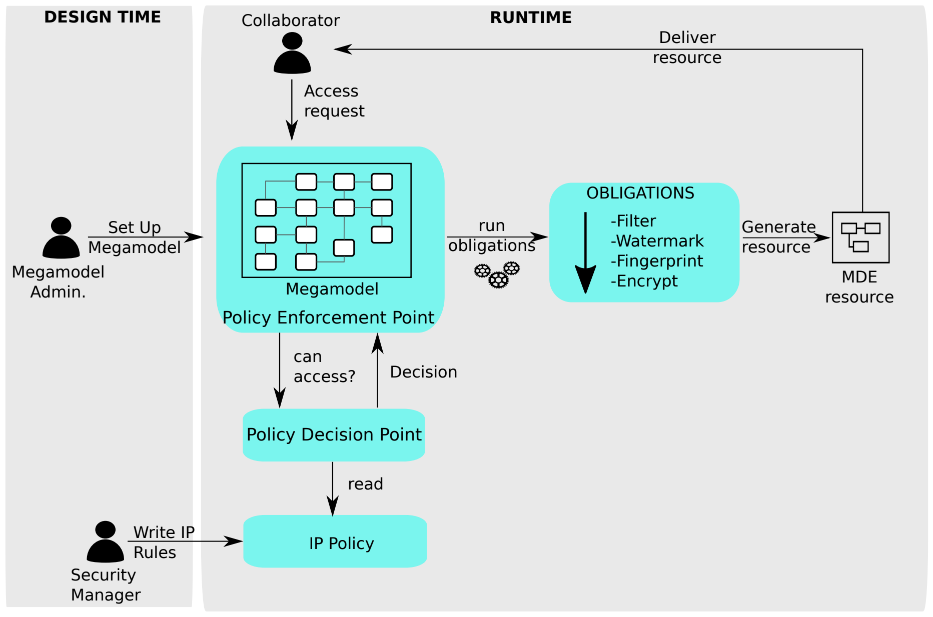

Fig. 1. IP Protection Approach

The same language can be used to define policies to restrict both internal and external access to the model elements, depending on the required security level.

Authorization The IP policy language must support the definition of authorization rules where each *rule* specifies whether a *Subject* can perform a given *Action* (typically *read* or *write*) on a given *Object* by attaching to this triple a *decision* element (e.g. *allow* or *deny*)[4]. Effectively, this corresponds to a classical discretionary access-control policy as discussed in [18].

These concepts are generic enough to cover authorization needs on all kinds of modeling artefacts but we also foresee their refinement when more domain-specific actions are required (e.g. the need to indicate whether a transformation can be executed and not only read or written).

For this global policy, the granularity is set at the model level (i.e. users can request access to complete modeling artefacts). More specific permissions, for instance to grant access on specific parts of a model, are managed with the *Filter* obligation described below.

Conditions To increase the expressiveness (w.r.t. protection needs) of the IP protection rules, the language must include context conditions such as time (i.e.

[4] You can also add a *default* authorization policy to indicate whether allow or deny access to elements when an explicit permission is not provided

the model can only be accessed within a given time interval) and geographical location as well as privacy concepts, such as the purpose of the request.

Obligations Even when all conditions are met we may still want to process the model before we deliver it to the user. As such, obligations are tasks to be conducted by the security engine before, after or together with the enforcement of the authorization decisions.

The IP policy language must support, at least, the following *Obligations*: *Watermark*, *Fingerprint*, *Encrypt* and *Filter*. The *Filter* obligation is linked to a function that calculates a view on the model based on the user requesting the access. This view is "the" model from the perspective of that user. If existing, this is the first obligation to be executed. Then we can proceed to watermark/fingerprint the model (for IP protection) or encrypt it (e.g. for a secure transmission).

Listing 1.1 shows an example of an IP policy where companies try to access an important resource, the *ArchitectureModel*. As we trust those partners holding the *TrustedPartner* role, we state in the first rule that they have the right to read it but only between 9:00 and 17:00. Even when access is granted, the model resource is sent encrypted and watermarked in order to protect our IP. Instead, partners holding the *Contractor* role are in charge of developing specific parts of the system and do not need access to the *ArchitectureModel*. Thus, in the second rule we forbid them to *read* the model.

Listing 1.1. Policy Example

```
Rule r1 (
  Subject S1 {attributes <'role' = 'TrustedPartner'>},
  Object ArchitectureModel,
  Action Read,
  Time 9:00-17:00
) -> Accept [encrypt, watermark]

Rule r2 (
  Subject S2 {attributes <'role' = 'Contractor'>},
  Object ArchitectureModel,
  Action Read
) -> Deny
```

3.2 Enforcement at Runtime

Given an IP policy, our framework needs to combine a number of runtime components to enforce it. We first discuss the mechanism for evaluating the rules and then those in charge of fulfilling the obligations.

Authorization Evaluation and Enforcement The recommendation in the implementation of modern policy frameworks is separating the infrastructure logic from the application logic by using a *reference monitor* architecture [1].

This architecture consists in two basic components: a Policy Enforcement Point (PEP) and a Policy Decision Point (PDP). Every model access action requested by a subject is intercepted by the PEP that, in turn, forwards it to the PDP to yield an access decision. Note that values for attributes such as location or time (or any other contextual attribute referenced in the access conditions) must be attached to the access-request (or directly taken from the runtime environment) in order for the PDP to evaluate the match.

We follow this architecture. Concretely, we intercept access requests to any of the artefacts in the megamodel by hooking our framework to the API of the modeling technology stack. The request data is then forwarded to our PDP to resolve the request. The PDP is implemented as a model transformation itself [15] to facilitate its execution within the context of modeling tools.

The access decision yielded by the PDP includes the obligations to be fulfilled which are passed on to the next components in the framework.

Filtering Mechanisms for MDE The filtering obligation aims to provide a finer degree of access-control (at the model element level) for the users. Its implementation requires an efficient fragmentation mechanism for models.

Given the nature of collaborative projects, where many distributed and remote users may require quick access to the models elements, we settled down for a *materialized view* mechanism as the most adequate solution. That is, once a user requests access to a model, she will get back a view of the model containing all the elements she has read access to. This view will be materialized (instead of calculated on the fly every time the user needs to access a different element within the model) and, from a user point of view, it will be indistinguishable from a "normal" model while protecting the parts of the original model users should not even know they exist.

These kind of *materialized views* may be derived by using techniques such as bidirectional transformations [11] or virtual models [4], being the latter our choice. In this sense, the filter obligation must be accompanied by a view definition. This view definition can be automatically inferred from the model-element level access-control policy as we show in [14]. Note that, to ensure that the generated view is always consistent, the view inference process follows a number of permission propagation and consistency rules. In short, these rules propagate permissions across the containment and hierarchy relationships to assign permissions to any element not directly mentioned in the access-control policy. This propagation takes into account that no elements whose access depends on a forbidden element can be part of the view, no matter their individual permission.

If the user is allowed to perform modifications, she can send back the updated view and the changes will be propagated to the original model under some strict and well-known (especially in the database community) limitations.

Cryptography for MDE The adaptation of encryption techniques for models depend on the storage format used to exchange them. In the simplest scenario (models exchanged as XMI files), encryption can be directly performed with off-the-shelf crypto tools for XML. Similarly, for models stored in a relational or NoSQL databases, available encryption mechanisms in those platforms could be directly used with some adaptations.

Watermarking and Fingerprinting for MDE Contrary to the previous obligation, standard watermarking techniques cannot be applied to protect the IP of modeling artefacts. To begin with, watermarking for non-media data is challenging as the toleration level to modifications is much lower than for media data. And the few approaches for non-media data (e.g. for XML or graphs) rely on a number of assumptions (basically related to the existence of non mutable identifiers and abundant numerical data) that are not true for models. Therefore, we have developed a new robust (i.e. resistant to data modification to a certain degree) labelling mechanism that uniquely identifies model elements considering both their contents and position (i.e. relationships with other elements) in the model [16].

This labelling mechanism enables us to then reuse in MDE state-of-the-art watermarking algorithms for non-media data. The process of watermarking uses our labelling mechanism to select a small percentage of the total number of elements to be part of the watermark. Then, we could either include the watermark directly in the model (by making some minor, and mostly imperceptible changes, on the element data like modifying the less relevant digits in numbers or other types of data) or calculate and store the watermark corresponding to the selected elements outside the model (e.g. encrypted on a trusted third party). By using our method with different degrees of robustness in the tolerance to modifications, it can be used to both, identify ownership and tampering, this is, watermarking and fingerprinting.

4 Roadmap

We believe the framework we have proposed provides the basis for the holistic IP protection of MDE artifacts in collaborative modeling scenarios. It reuses, integrates and adapts a number of model-driven technologies and security mechanisms to protect IP by concealing access and tracking leaks. A preliminary implementation of the main components of this framework has been conducted as a proof-of-concept.

Nevertheless, plenty of interesting challenges to consolidate and expand the framework must be addressed before it can actually represent a valid solution for IP protection in industrial projects. In what follows we discuss some of these challenges. Challenges are divided in conceptual, technical and domain specific challenges.

On a conceptual level, we consider interesting to explore the integration of: (1) Peer-to-peer collaboration scenarios where there is no central partner that defines the global protection policy, (2) Multi-level security policies, and (3) functional encryption [3] results, where a decryption key enables a user to learn a specific function of the encrypted data (this could be assimilated to the concept of model query) but nothing else about the data itself.

On a technical level, the framework may be enhanced to support: (1) content-based requests where users define the type of data they need without even knowing in what model/s that data is located, (2) advanced integration of user updates on model fragments to deal with the "view updating" problem.

On a domain-specific level, when collaboration occurs in the development of critical systems such as those integrated in the automobile and avionics domains, traceability and accountability become essential security properties. The history of a model must be thus immutable and verifiable. In that sense our framework may be enhanced by the use of blockchain technologies. Initial steps towards the use of blockchain for model-based knowledge management provided in [10] and [12] may be integrated in our approach so that access and manipulation requests may be stored in a blockchain together with the model modification for further use in verification and certification processes.

5 Conclusions and Future Work

We have presented an approach aimed at providing the basis for the holistic protection of MDE artefacts in collaborative model-based co-engineering scenarios. Our approach is based on the coordination and adaptation to the modeling technical space of several different IP protection mechanisms such as Cryptography, Access-control and Digital Rights Management. This is achieved through a framework composed of two main components: 1) a policy language, that enables the specification of the IP protection policy of the collaborative project; and 2) a megamodel-based runtime infrastructure, in charge of evaluating and enforcing that policy upon model access and manipulation requests from users during the collaborative modeling process.

As future work, we envisage several extensions to our approach. From the challenges outlined in Section 4, we intend to preferentially focus on the domain-specific challenges. Concretely, we intend to study the adaptation of our framework to critical domains such as the aerospace and automotive domains. We are specially interested in dealing with traceability and accountability issues together with an on-the-fly validation & verification of security properties. This requires the adaptation and integration of additional mechanisms, such as blockchain and solvers, into our framework.

Acknowledgement.

This work is partially funded by the H2020 ECSEL Joint Undertaking Project "MegaM@Rt2: MegaModelling at Runtime" (737494) and the Spanish Ministry of Economy and Competitivity through the project "Open Data for All: an API-based infrastructure for exploiting online data sources" (TIN2016-75944-R).

References

1. Information technology - Open Systems Interconnection - Security frameworks for open systems: Access control framework (International Standard ISO-10181-3/X.812), 1996.
2. J. Bézivin, F. Jouault, and P. Valduriez. On the need for megamodels. In *OOPSLA/GPCE workshops.*, 2004.
3. D. Boneh, A. Sahai, and B. Waters. Functional encryption: a new vision for public-key cryptography. *Communications of the ACM*, 55(11):56–64, 2012.
4. H. Bruneliere, J. Garcia, M. Wimmer, and J. Cabot. Emf views: A view mechanism for integrating heterogeneous models. In *ER*, pages 317–325, 2015.
5. X. Cai, F. He, W. Li, X. Li, and Y. Wu. Encryption based partial sharing of cad models. *ICAE*, 22(3):243–260, 2015.
6. C. Debreceni, G. Bergmann, I. Ráth, and D. Varró. Enforcing fine-grained access control for secure collaborative modelling using bidirectional transformations. *SOSYM*, pages 1–33, 2017.
7. E. Durling, E. Palmkvist, and M. Henningsson. FMI and IP protection of models: A survey of use cases and support in the standard. In *Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, number 132, pages 329–335, 2017.
8. D. Ferraiolo, J. Cugini, and D. R. Kuhn. Role-based access control (rbac): Features and motivations. In *ACSAC*, pages 241–48, 1995.
9. H.-G. Fill. Using obfuscating transformations for supporting the sharing and analysis of conceptual models. In *Multikonferenz Wirtschaftsinformatik 2012 - Teilkonferenz Modellierung betrieblicher Informationssysteme*, Braunschweig, 2012.
10. H.-G. Fill and F. Härer. Knowledge blockchains: Applying blockchain technologies to enterprise modeling. In *HICSS*, 2018.
11. J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, and A. Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *TOPLAS*, 29(3):17, 2007.
12. F. Härer. Decentralized business process modeling and instance tracking secured by a blockchain. In *ECIS*, 2018.
13. T. Kim, C. D. Cera, W. C. Regli, H. Choo, and J. Han. Multi-level modeling and access control for data sharing in collaborative design. *AEI*, 20(1):47–57, 2006.
14. S. Martínez, A. Fouche, S. Gérard, and J. Cabot. Automatic generation of security compliant (virtual) model views. In *ER*, pages 109–117, 2018.
15. S. Martínez, J. García, and J. Cabot. Runtime support for rule-based access-control evaluation through model-transformation. In *SLE*, pages 57–69, 2016.
16. S. Martínez, S. Gérard, and J. Cabot. On watermarking for collaborative model-driven engineering. *IEEE Access*, 6:1–1, 2018.
17. P. H. Nguyen, M. Kramer, J. Klein, and Y. Le Traon. An extensive systematic review on the model-driven development of secure systems. *IST*, 68:62–81, 2015.
18. R. S. Sandhu and P. Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.
19. E. Stepper. CDO model repository, 2010.
20. D. Varró and A. Balogh. The model transformation language of the viatra2 framework. *Science of Computer Programming*, 68(3):214–234, 2007.
21. Y. Wang, P. N. Ajoku, J. C. Brustoloni, and B. O. Nnaji. Intellectual property protection in collaborative design through lean information modeling and sharing. *JCISE*, 6(2):149–159, 2006.
22. E. Yuan and J. Tong. Attributed based access control (abac) for web services. In *ICWS*, 2005.