# Towards a unified model-based engineering framework: integrating UML profiles into EMF

Muhammad Asim Minhas[1,*], Vitus Lüntzel[1] and Erik Burger[1]

[1]Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany

**Abstract**

The development of robust modeling frameworks that support consistent and accurate representation of system models is key to coping with the increasing complexity of Systems Engineering. The Unified Modelling Language (UML) supports the creation of profiles as a means of extension, chiefly to languages supported by commercial tools. For example, profiles have been applied in such popular domain-specific modelling languages (DSMLs) as SysML. However, proper support for profile-based extension is not available for the widely used open-source Eclipse Modelling Framework (EMF). In this paper, we present a method to create EMF-based representations of UML-based profiles. By applying manual transformations from UML profile elements to EMF metamodel entities, we have developed a method through which the semantics of UML-based profiles can be captured within the target EMF metamodel. We have applied our approach to one of the SysML-based Risk Analysis and Assessment Modeling Languages (RAAML) profiles and are able to develop the corresponding metamodel. This validates that our proposed technique provides a standardized EMF representation of the widely used UML-based profiles, thus contributing to the advancement of Model-based Systems Engineering (MBSE).

**Keywords**

Model-driven Software Development, SysML, UML Profiles, EMF

## 1. Introduction

The profile mechanism of UML is a powerful concept to achieve extendability and configurability in modeling tools. However, since it violates the strict metamodelling principle [1], the UML profile mechanism requires a deep integration in the tools that support it. This requirement hinders integration into common modelling tools that are based on the popular Eclipse Modeling Framework (EMF), because EMF follows the paradigm of a clear separation of (fixed) meta-metamodel layer, metamodel(MM) layer, and model layer.

The complexity of UML poses a problem, too, for the integration of profile-based domain-specific modeling languages (DSMLs) with other tools and languages. The UML specification offers a wide range of concepts, but DSMLs often need only a limited span of that full range. In spite of this, every concept must be supported, because also the instances of the domain-specific profile must be valid UML instances. For this reason, integration developers need to understand the full complexity of UML, even when an integration requires only the domain-specific concepts.

We argue that if metamodelers are given complete freedom in choosing the technical space, they may decide *against* the usage of the UML profile mechanism and *instead for* creating a custom, Ecore-based MM. Nonetheless, the support of UML models may still be required with applied profiles (e.g., SysML), to keep consistency with other tools. However, those profiles are often modelled in commercial tools.

In this paper, we present an approach to transform UML models with applied profiles into a compact Ecore-based representation that can be flexibly used with all EMF-based modelling tools. The resulting Ecore-based metamodels contain only the concepts that are relevant in the respective domain, and are thus independent of the full UML MM. The compactness of the resulting MM enables, likewise,

*ICMM 2025: (In-)Consistency Management in Modeling, co-located with STAF 2025, 10–13 June 2025, Koblenz, Germany*

*Corresponding author.

muhammad.minhas@kit.edu (M. A. Minhas); luentzel@kit.edu (V. Lüntzel); burger@kit.edu (E. Burger)

https://dsis.kastel.kit.edu/staff_muhammad_minhas.php (M. A. Minhas); https://www.itiv.kit.edu/english/21_8169.php (V. Lüntzel); https://dsis.kastel.kit.edu/staff_erik_burger.php (E. Burger)

0000-0002-1733-4539 (M. A. Minhas); 0009-0006-5739-0486 (V. Lüntzel); 0000-0003-2832-3349 (E. Burger)



published 2025-12-10

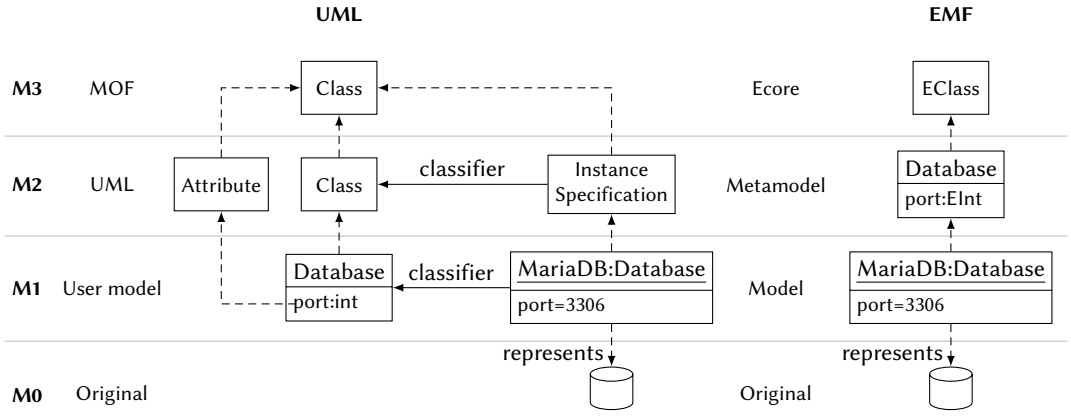


**Figure 1:** Metalevels in UML without profiling (left) vs. EMF (right), shown at the example of modelling a database (⇢ denotes instantiation unless labelled otherwise)

the definition of more compact consistency specifications, which are more understandable and better maintainable.

# 2. Foundations

## 2.1. Metamodeling vs Profiles

Metamodelling and profiling are the two approaches that are commonly used in Model-based Systems Engineering (MBSE). Both of these approaches support domain-specific modeling, but there are differences in their underlying concepts and applications. This section provides an overview of both approaches, relative advantages of their usage and their integration options into MBSE.

### 2.1.1. Metamodeling with EMF

Although the MOF standard [10] does not mandate a specific number of meta-levels, most modelling tools follow the four-level hierarchy labelled $M0$–$M3$, where a higher number means a higher level of abstraction. As shown in Figure 1, elements at level $n+1$ *represent* elements at level $n$, while we call the opposite direction *instantiation*, as usual in object-oriented approaches. The modelled *originals* (following the terminology of Stachowiak [15, p. 131–133]) reside at level $M0$. We illustrate this in the figure at a simple example of modelling a database. The main difference between modelling in UML (without profiling) and doing the same thing with EMF can be seen in the treatment of levels $M1$ and $M2$: While the elements that represent the domain originals are located at level $M1$ in both approaches, the classes that describe these elements are also located at level $M1$ when using UML, but at level $M2$ in EMF.[1] The $M2$ level in the UML case contains the UML MM, which is an instance of MOF ($M3$), while in the EMF case, the $M2$ contains a (possibly user-defined) domain MM that instantiates the EMF-specific meta-metamodel *Ecore* of level $M3$ (which is aligned with essential MOF (EMOF)).

The primary characteristics of EMF-based metamodels are:

- **Explicit Semantics**: In Ecore-based metamodels, domain concepts are precisely defined as first-class elements, which facilitates rigorous semantic interpretation.[16]
- **Tooling Support**: Robust tooling is provided by EMF for creating models, validation, transformation and code generation.[3]
- **Modularity & Extensibility**: As inheritance and composition are supported in EMF, this allows seamless extension and integration with other metamodels.

[1] This leads to the confusing situation that there are two types of instantiation in UML: The "true", level-crossing intance-of relation ⇢, and the UML relation classifier inside level $M1$.

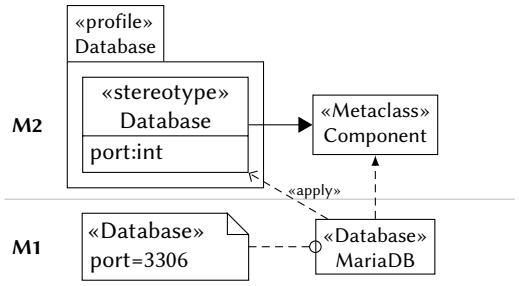


**Figure 2:** UML profiling mechanism applied to the database example (- - -▸ denotes instantiation)

- **Independence from UML**: In contrast to profiling, EMF metamodels are not dependent on UML semantics, which makes them more suitable for diverse domains beyond software engineering.
- **Alignment with OMG Levels**: EMF's foundation on OMG levels ensures a structured model development.

### 2.1.2. UML Profiling

UML profiling is an approach that allows customization of UML models by defining and applying stereotypes with tag definitions as shown in Figure 2. For the sake of simplicity, we show the profile package as a part of metalevel $M2$, although it is part of the user model and should thus be at level $M1$. Either way, profiles violate the strict metamodelling principle: If put at level $M2$, they would have an instantiation relation inside this level since they are instances of UML, not of MOF[2]; put at level $M1$, they would have a non-instantiation relation between Database ($M1$) and Component ($M2$). Because of the dual function of profile packages, it is sometimes said that they live at "level $M1.5$".

The key features of the profiling mechanism are:

- **Non-intrusive Extension**: Domain-specific customizations can be performed using profiling, without altering the UML metamodel [13].
- **Preservation of UML Semantics**: Profile models are always UML-conformant, enabling interoperability with UML-based tools [11].
- **Limited Expressiveness**: Due to its dependence on the UML metamodel, it only allows extension of existing UML elements. This limitation restricts the freedom to define new and domain-specific semantics which are often necessary in complex modeling scenarios. [2]

## 2.2. Shortcomings and Issues in UML Profile Mechanism

UML profiles provide a standardized approach for specializing and extending the UML MM for specific domains without changing its structure. Additionally, these profiles offer valuable customizations but have limitations and constraints that affect their utility and applicability. As Kirill Fakhroutdinov puts it, "a profile is a restricted form of a reference MM that must always be related to some reference MM created from MOF such as UML or CWM. It is not possible to define a standalone profile without its reference metamodel"[3]. The main objective of profiles is to offer a balance between flexibility and standardization, i.e., while maintaining compatibility with core UML framework, they also facilitate domain-specific extensions. This balance comes at the cost of substantial constraints on how profiles can be modified and extend the underlying MM.

### 2.2.1. Fundamental Structural Limitations

The expressiveness of the profiles is limited due to the constraints imposed on the customization of the MM. For instance, "new meta classes must not be inserted in the class hierarchy of the MM and it

[2]CMOF imports parts of UML, but not the profile concept.
[3]https://www.uml-diagrams.org/profile.html. Last Accessed on 03 March 2025

is forbidden to modify the class definitions in the MM"[14]. Due to this restriction, it becomes very difficult to introduce new concepts that do not align with existing UML meta-classes.

#### 2.2.2. Technical and Implementation Constraints

There are certain technical issues that impact the implementation and usage of profiles. For example, "it is not allowed to have associations between stereotypes or stereotypes and metaclasses in profiles, because this is a modification of metaclasses"[14]. Consequently, modelers are forced to use less direct approaches to represent relationships between domain-specific concepts.

#### 2.2.3. Expressiveness and Semantic Issues

Profiles often may not provide precise semantics for domain-specific concepts. For example, one criticism on the UML-RT profile is that "its modeling entities and syntactic constructions lack precise semantics and clearly defined syntax" [6]. This can lead to ambiguities and inconsistent interpretations among various tools and their users. Similarly, profiles like UML-RT "do not support modelling timing issues nor allow concurrent states". Furthermore, there are other tool support and practical application issues, such as visualization limitations, due to which profile-specific information is not visible in diagrams.

## 3. Transforming Profiles to EMF-based Metamodels

In this section, we present a compositional transformation method for Domain Specific Modeling Languages (DSMLs). The core objective of this approach is to provide EMF representation of the UML profiles by using metamodeling to support the development of DSMLs. We will describe the process of manually creating an Ecore-based metamodel and transforming it from UML with profiles to this metamodel. We intend to address the following research questions:

- **RQ-I**: To what extent can we preserve the semantics of the customized profiles in the unified EMF metamodel?
- **RQ-II**: What is the effect of the unified EMF metamodel on creating and executing consistency specifications between the DSML and other metamodels?

In order to address these research questions, a new method based on pure EMF metamodeling has been introduced. The core idea is to eliminate the implicit shortcomings within the profiling mechanism, and rather than using UML/SysML profiles, the semantics from the existing profiles should be merged into the EMF metamodels using a formal and predefined method.

### 3.1. Mapping method

The UML specification Clause [11, section 12.3.3.1.3] provides the guiding principles for transforming UML profiles to Ecore metamodels. The conformance of this clause requires that MOF-equivalent semantics of UML profiles should be preserved during their transformation to the Ecore metamodels. Using these guidelines, the tools should ensure that the behavior of the UML profile should be consistent with MOF semantics. Our proposed approach is aligned with this guidelines, and preserves the semantics and structure of UML profiles during their transformation to Ecore metamodels. In this way, the mapping process of UML profiles to Ecore is naturally facilitated by this clause because it explicitly directs to treat profiles as packages and stereotypes as classes, which is in line with the internal structure of Ecore. Furthermore, detailed instructions mentioned in the specification on serializing profiles using XMI, such as using nsURI and nsPrefix to identify profiles provide opportunities for seamless integration with Ecore metamodeling approach, since Ecore also utilizes the same concepts for package and class definitions.

| UML element | Ecore class |
|---|---|
| Metaclass | EClass (abstract) |
| Stereotype | EClass (concrete) |
| Tag definition | EAttribute |
| Enumeration | EEnum |
| Association | EReference |
| Constraint | EOperation |
| DataType | EDataType |

| UML concept | Ecore concept |
|---|---|
| extension | generalization |
| profile application | instantiation |

**Table 1**
Mappings of UML elements/concepts and Ecore classes

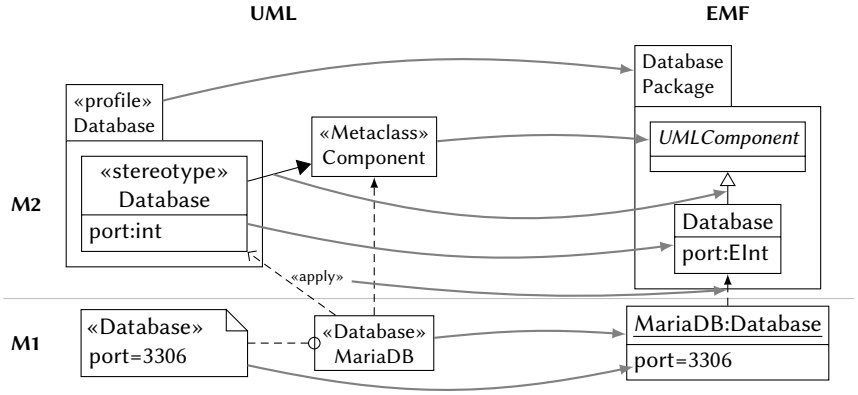


**Figure 3:** Proposed transformation of a UML profile to an Ecore-based metamodel shown at the database example

Table 1 shows the mapping of source to target classes of the proposed transformation scheme. An application of the scheme is shown in Figure 3: Every profile package is mapped to an EPackage of the same name. Each UML element that is used as a metaclass by a stereotype is re-created in the Ecore metamodel. Here the class Component is transformed into the abstract EClass of the name UMLComponent[4]. The stereotype Database is mapped to a concrete EClass of the same name. The tag definition port of integer type will be mapped to an EAttribute of EInt type with the same identifier. The extension relationship in the profile is mapped to a generalization relationship between Database and UMLClass. It is pertinent to mention that the application of a stereotype the user models on the UML side is replaced on the EMF side by the actual instantiation (at level $M1$) of domain-specific elements defined with the ECore-based metamodel at the $M2$ level.

The proposed technique of mapping UML profile elements to Ecore elements aligns with both formal MOF semantics as well as the practical necessities of Ecore which allows an enriched and consistent transformation of constraints, datatypes, and associations. This alignment provides an assurance that even after serialization of the profiles in different contexts, their behavior remains predictable and also standard-compliant. This conformance to the standard facilitates interoperability of various tools and model exchangeability to achieve key advantages in both UML and Ecore ecosystems. Nevertheless, this UML specification clause is a foundation for such transformation efforts which provides the assurance that the resultant Ecore metamodels maintain the semantics, integrity and usability of the original UML profiles. Since Ecore-based MM contains corresponding elements for all elements in the UML profile and in the UML MM, the resulting MM preserves the semantics of the profile elements while eliminating the limitations of the profiles and providing the flexibility to extend the mapped elements through customized domain-specific elements.

[4]Re-creating UML classes in the transformed metamodel in an on-demand way may affect the reusability of the metamodel negatively. See section 6 for a discussion of limitations.

### 3.2. Application of the method

In the current, early state of our work, the process of defining the metamodel and the corresponding transformation rules is manual, by following the rules described in subsection 3.1. To automatically create these elements, two transformation definitions would be necessary:

First, a model-to-model transformation from UML to Ecore would have to be defined that analyzes the profile and creates corresponding abstract classes for all UML concepts that are used as metaclasses, and all concrete classes according to the mapping of Table 1.

Second, a higher-order transformation would have to be defined that takes the UML profile as an input and would produce as output a model-to-model transformation, which itself takes a UML model with the applied profile as input, and produces an instance of the MM created in step one.

## 4. Case Study: RAAML General Concepts Profile to Ecore Metamodel

For evaluation purpose, we have applied the proposed technique to one of the standard languages from OMG released as an extension to the SysML Profile: the *Risk Analysis and Assessment Modeling Language (RAAML)* [12]. The language has been released as a set of seven profiles and seven libraries; we have selected one of the profiles to provide a proof of concept.

### 4.1. Proof of Concept

In our example, we have selected the *RAAML General Concepts Profile*. In RAAMLGeneralConceptsProfile, (Mitigation, Recommendation, Detection, Prevention) are subclasses of ControllingMeasure, and together all of these stereotypes *extend* the metaclass Dependency. RelevantTo, being the superclass of PresentIn, also extends Dependency from the UML metamodel. Similarly, (FailureMode, Error, Hazard, and Fault) inherit from the stereotype Situation, and all of these stereotypes extend the metaclass Class from the UML metamodel. FailureState extends State. An ElementGrouptBasedItem inherit from ElementGroup, and together they extend the metaclass Comment. A SingElementItem inherits from Item, and both of these extend the metaclass Element, which is also extended by an additional stereotype Undeveloped.

For merge/transformation of the given profile, the proposed approach has been applied manually following the transformation rules as mentioned in Table 1. As a result, we get a package with the name *RAAMLGeneralConcepts*. All metaclasses (Dependency, Element, Class, Comment, and State) have been mapped to their corresponding abstract classes, with a prefix of the name of the reference metamodel. In our example, all metaclasses have been mapped with the prefix "UML", as shown in Figure 5.

All elements of the UML metamodel that are used as metaclasses in the profile are mapped to abstract classes in the Ecore-based metamodel. In our example, these are UMLDependency, UMLClass, UMLElement, UMLState, and UMLComment. The profile extension relationship is transformed to an inheritance relationship in the Ecore-based metamodel. Tag definitions such as member and boundarymember, contained in the stereotype Item in the profile, are mapped to named references from the Item to UMLElement in the Ecore-based metamodel.

We have translated all RAAML profiles and libraries manually into a monolithic RAAML metamodel, which can be accessed on GitHub[5]. For bringing the semantics from the profile to the Ecore-based metamodel, we have introduced the concept of selective semantic transfer, which means that only essential and required semantics should be carried forward to the Ecore metamodel. For a demonstration of this concept in our example, consider the metaclass Dependency in the UML metamodel: Suppose, we intend to bring the metaproperties and semantics of Dependency to our Ecore metamodel. For this purpose, we deliberately created two additional EReferences client and supplier from UMLDependency to UMLElement, and we enforced that client and supplier must be different using the OCL constraint shown in Listing 1.

[5] https://github.com/masimminhas/RAAMLEcoreMetamodel/tree/main/edu.kit.sdq.dsis.metamodels.raaml

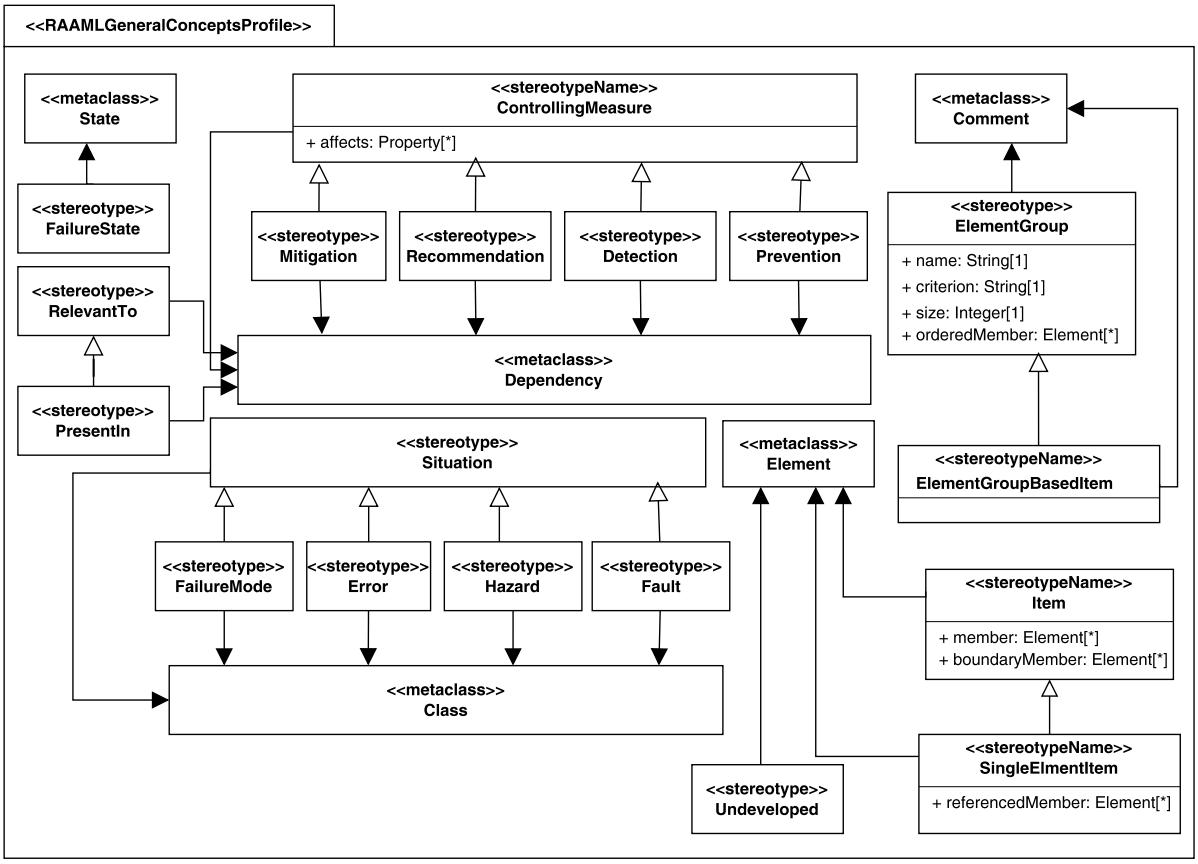


**Figure 4:** RAAML General Concepts Profile from OMG

```
context UMLDependency
inv: self.client->forAll(c | self.supplier->excludes(c))
```

**Listing 1:** Constraint to ensure that clients and suppliers must be different

Selective semantic transfer provides many benefits. For example, if our target Ecore metamodel only requires a subset of the profile semantics, we can simply bring them by applying the transformation. This also provides performance gains, as the whole UML infrastructure is not required to be linked with the metamodel. Furthermore, this approach simplifies the transformation mechanism, making it easier to manage and maintain the metamodel. This approach is particularly beneficial for legacy system integration and consistency management, for which we only need to transfer semantics that are compatible with the already existing infrastructure. Selective semantic overlap addresses our first research question related to the semantics of the source profiles.

## 4.2. Managing Profile Evolution

The proposed transformation rules provide traceability between source UML/SysML profiles and target ecore metamodels. As these models evolve over time, this traceability facilitates maintaining consistency. Furthermore, when changes are made to either source profiles or the target metamodels, traceability helps conduct an impact assessment. Maintainability is enhanced when a change impact assessment computes the set of changed elements. The other aspect is the decoupling of the extension layer, which contains a mapping of meta classes as set of abstract Eclasses that provides a separation of concern to adapt profile-based changes within the Ecore metamodels. As model-driven engineering evolves, approaches like selective semantic transfer will play an increasingly important role in bridging different

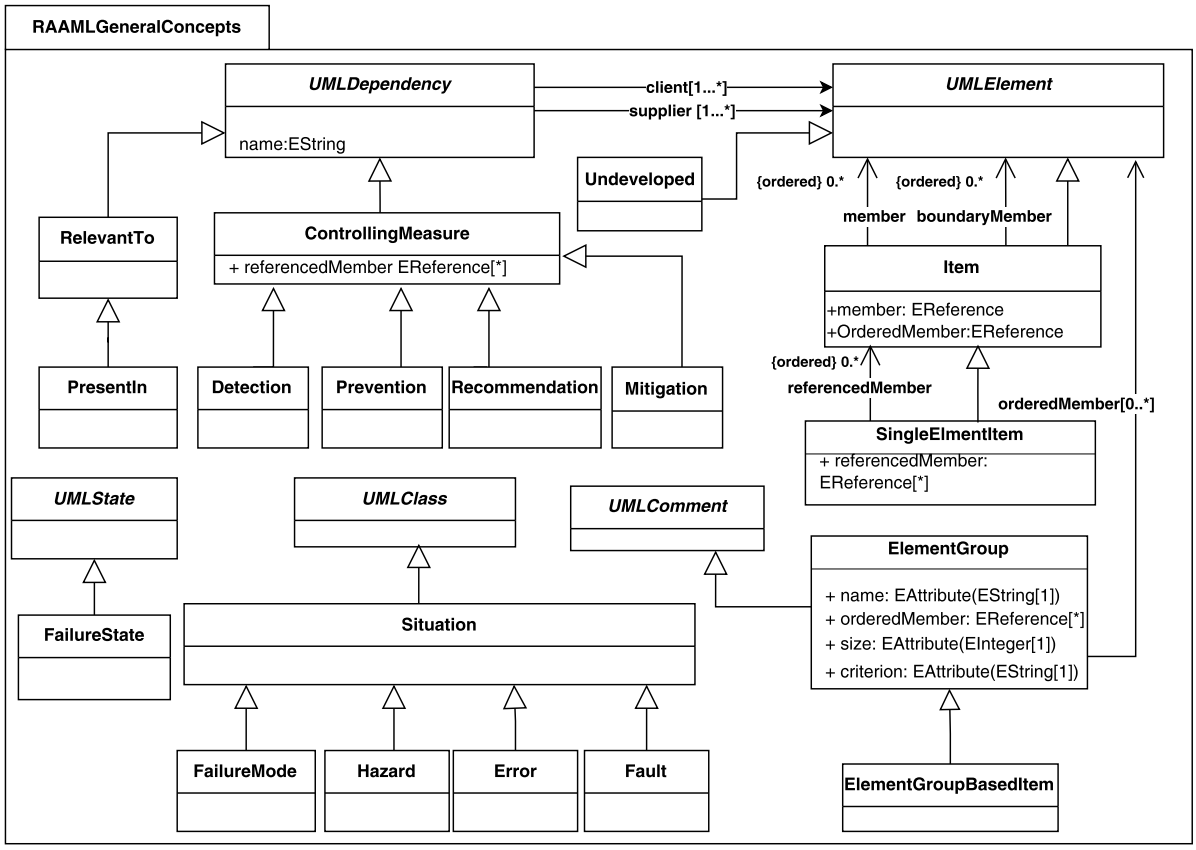


**Figure 5:** Ecore Metamodel after Profile Merging

modeling paradigms and facilitating the development of complex and variable systems.

### 4.3. Discussion

By mapping UML profile elements to the Ecore constructs, domain-specific concepts can be represented as first-class metamodel members, increasing semantic clarity and reducing ambiguity [2]. Particularly, transformation of stereotypes to the EClass elements can simplify the definition and specification of complex ideas and domain-specific concepts, mainly because of the enhanced flexibility and expressiveness of the metamodeling. The resulting Ecore-based metamodels provide seamless integration with EMF. This integration into EMF is particularly valuable as it bridges the gap between various modeling paradigms: Using SysML, as a Systems Engineering language might become challenging in certain environments. To overcome this issue the transformation with semantic preservation to Ecore produces a standardized representation, which can be processed by a wider range of tools. As an example, a transformation from SysML to Promela by Iqbal, Arcuri, and Briand showed that "SysML is a vast language" where specific elements can be handled through transformation approaches [4]. Furthermore, the transformation of selected semantic properties is essential to model complex scenarios; otherwise, full semantic transfer might produce unnecessary complexity or inconsistencies. The approach allows engineers to tailor the transformation to their specific needs by focusing on the semantic properties that are most relevant to their particular application domain.

## 5. Related Work

The *EMF profiles* project by Langer et al. [7] follows the core idea of defining a profile metamodel, and defining the stereotype metaclass of this metamodel as a subclass of Ecore EClass. This way, a

stereotype can inherit instantiation capability at level $M2$, and the profile application is represented by instantiating a specific stereotype. To keep track of the relationship between the stereotype and the model elements to which it has been applied, a separate metamodel package ProfileApplication has been introduced. This package contains the class SterotypeApplication with a reference appliedTo to arbitrary elements of type EObject. Each new profile is, by default, a subclass of StereotypeApplication, which makes it possible to inherit the appliedTo reference. An annotation mechanism has been proposed using EMF profiles in [8] without the need to make changes in the metamodels.A fully automatic approach for bridging UML profiles to MOF-based MM has been presented by Malavolta et al[9]. An MOF MM is generated from any UML profile and bi-directional model transformations between UML models and their MOF-based representations. Phase 1 of the proposed UML bridge utilizes a complete MM of UML, following a lossless transformation contrary to our approach, in which we bring only relevant concepts from the UML metamodel in a single step, thus eliminating the need for the whole UML metamodel. In phase 2, a slicing algorithm has been proposed, which is a semi-automated way and requires an MOF MM generated in the first step and an annotated model that provides input to the slicing algorithm related to instantiated meta classes in the target MOF model. In our approach, we deliberately selected a lossy transformation to avoid complexity. Although EMF profiles offer an adaptation mechanism to the UML profile with reusability and evolvability benefits, the offered approach still has several limitations. EMF profiling inherits some of the issues from UML profiling, such as the complexity of profile management. Particularly for large and complex systems, this complexity might result in maintainability and scalability issues. As EMF tools do not support EMF profiles natively, additional plugins such as EMF profile plugins, are required for their proper usage.

EMF Facet[6] was a project similar to EMF profiles that provided a non-intrusive extension of Ecore-based metamodels. The project has been archived since 2020.

## 6. Conclusion

We have presented a method to make UML models with applied profiles accessible to the Eclipse Modeling Framework (EMF), so that they can be used for model-driven purposes such as model transformations and concistency preservation. We have identified two limitations of our method:

First, since we recreate the UML elements used as metaclasses in the profile as classes in the Ecore-based metamodel, instances of this metamodel cannot be used without the knowing the exact set of metaclasses. This means that any transformation, consistency specification, editors and other artefacts that depend on the generated metamodel have to be adapted or even recreated for every profile that is supported. This is a drawback compared to UML, where models can also be exchanged without the profile application. Second, if our process is applied to different UML profiles, the resulting metamodels will not be compatible with each other per se, and classes (such as UMLComponent) will be duplicated in different namespaces. This is a design decision, since building on a common UML core would have meant that we would have to replicate the complete UML metamodel. This would actually combine the worst of both worlds since the extension of a metamodel by inheritance is less flexible than profile application, but the full complexity of UML is still preserved.

In future work, we plan to evaluate the effects of using a standalone metamodel that is independent from UML on the creation of consistency specifications in the Vitruvius approach [5], to answer research question RQ-II that we formulated in section 3. Furthermore, we plan to establish an automated process of creating the Ecore-based metamodel and the corresponding model-to-model transformation, as we have outlined in subsection 3.2. In a first step, the transformation will be uni-directional, and offer only the import of UML models with applied profiles; in a second step, we will extend this to a bidirectional transformation, so that EMF instances can also be transformed back in to UML instances, and roundtrip engineering can be supported. Finally, we plan to extend our approach so that multiple profile applications are supported with the use of multiple inheritance within EMF.

[6] https://projects.eclipse.org/projects/modeling.emft.emf-facet. Last accessed on 10 April 2025

## Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1608 – 501798263. It was supported by the pilot program Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF), and by KASTEL Security Research Labs. Thanks to our textician.

## Declaration on Generative AI

The authors have not employed any Generative AI tools.

## References

[1] Colin Atkinson and Thomas Kühne. "Profiles in a strict metamodeling framework". In: *Science of Computer Programming* 44.1 (2002). Special Issue on Unified Modeling Language (UML 2000), pp. 5–22. URL: https://www.sciencedirect.com/science/article/pii/S0167642302000291.

[2] Colin Atkinson and Thomas Kühne. "Model-Driven Development: A Metamodeling Foundation". In: *IEEE Softw.* 20 (2003), pp. 36–41.

[3] Lorenzo Bettini. *Implementing Domain Specific Languages with Xtext and Xtend - Second Edition*. 2nd. Packt Publishing, 2016.

[4] Muhammad Zohaib Iqbal, Andrea Arcuri, and Lionel Briand. "Transforming UML and SysML models to Promela for formal verification". In: *2012 19th Asia-Pacific Software Engineering Conference*. Vol. 1. IEEE. 2012, pp. 154–163.

[5] Heiko Klare et al. "Enabling consistency in view-based system development — The Vitruvius approach". In: *Journal of Systems and Software* 171 (2021), p. 110815. URL: https://www.sciencedirect.com/science/article/pii/S0164121220302144.

[6] Barath Kumar and Juergen Jasperneite. "UML Profiles for Modeling Real-Time Communication Protocols." en. In: *The Journal of Object Technology* 9.2 (2010), p. 178. URL: http://www.jot.fm/contents/issue_2010_03/article5.html.

[7] Philip Langer et al. "From UML Profiles to EMF Profiles and Beyond". In: *Objects, Models, Components, Patterns*. Ed. by Judith Bishop and Antonio Vallecillo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 52–67.

[8] Philip Langer et al. "EMF Profiles: A Lightweight Extension Approach for EMF Models." en. In: *The Journal of Object Technology* 11.1 (2012), 8:1. URL: http://www.jot.fm/contents/issue_2012_04/article8.html.

[9] Ivano Malavolta, Henry Muccini, and Marco Sebastiani. "Automatically Bridging UML Profiles to MOF Metamodels". In: *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*. ISSN: 2376-9505. Aug. 2015, pp. 259–266. URL: https://ieeexplore.ieee.org/document/7302461/.

[10] *OMG Meta Object Facility (MOF) Core Specification*. Version 2.5.1. Object Management Group. Oct. 2016. URL: http://www.omg.org/spec/MOF/2.5.1/.

[11] *OMG Unified Modeling Language (OMG UML)*. Version 2.5.1. Object Management Group. Dec. 2017. URL: http://www.omg.org/spec/UML/2.5.1/.

[12] *Risk Analysis and Assessment Modeling Language*. Version 1.1beta. Object Management Group. June 2024. URL: https://www.omg.org/spec/RAAML.

[13] Bran Selić. "A Systematic Approach to Domain-Specific Language Design Using UML". In: *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)* (2007), pp. 2–9.

[14] Christof Simons. "CMP: A UML Context Modeling Profile for Mobile Distributed Systems". In: *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. ISSN: 1530-1605. Jan. 2007, 289b–289b. URL: https://ieeexplore.ieee.org/document/4076968.

[15] Herbert Stachowiak. *Allgemeine Modelltheorie*. Wien: Springer Verlag, 1973.

[16] David Steinberg et al. *EMF: Eclipse Modeling Framework 2.0*. 2nd. Addison-Wesley Professional, 2009.