2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)

2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) | 979-8-3315-7990-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/MODELS-C68889.2025.00106

From the Lowlands of UML to the Highlands of MLM: An Evolutionary, Tool-Supported Multi-Level Method for Teaching Multi-Level Modeling

Ulrich Frank
University of Duisburg-Essen
Essen, Germany
ulrich.frank@uni-due.de

Pierre Maier
University of Duisburg-Essen
Essen, Germany
pierre.maier@uni-due.de

***Abstract*—The dissemination of multi-level modeling (MLM) depends not least on graduates who learned to appreciate the benefits of this modeling paradigm. To this end, effective approaches to teaching MLM are of vital relevance. Knowledge about teaching MLM is scattered and limited. Therefore, there is a need to capture and further develop this knowledge. This article presents a multi-level model that supports the entire cycle from the preparation, implementation, and evaluation of courses on conceptual modeling in general and on MLM in particular. It is executable and, thus, also supports course monitoring and management. The multi-level model should allow for further instantiations to represent courses on other topics.**

***Index Terms*—Teaching, Abstraction, Modeling, Programming**

I. Introduction

Multi-level language architectures offer a number of substantial advantages over traditional architectures. They allow the knowledge required for system design to be represented at a higher level of abstraction. In this way, conceptual redundancy can be reduced, which significantly promotes reusability, integration, and adaptability. The application to programming languages also enables new architectures of application systems that are characterized by a common representation of models and programs. This empowers users to navigate the models (and modeling languages), on which the program is conceptually based, during the runtime of a program.

Despite these obvious and undisputed advantages of such language architectures, they have so far failed to achieve a breakthrough in practice. In academia, too, research on architectures takes place in a small niche, although no serious criticism of the advantages of these architectures can be found in the literature.

Various approaches to the dissemination of MLM have been discussed in the community. On the one hand, they are aimed at consolidating existing approaches and thus striving for standardization in the long term [1], [2]. On the other hand, they aim to demonstrate the advantages of the architectures through suitable case studies and corresponding prototypes, e.g., [3]–[5]. Although such approaches are important, they are hardly sufficient if teaching is neglected as a central aspect of the dissemination of MLM. This includes teaching in the context of relevant degree courses at universities and also corresponding continuing education programs for professionals.

The design of appropriate teaching and learning methods is challenging in that previous experience can only be used to a limited extent. After all, it is not least about teaching a new paradigm. In this paper, we present a method for preparing, conducting, and evaluating MLM courses that addresses these requirements. The method and its possible variations are represented by a multi-level model that can be adapted to preparing courses on other subjects, too.

II. Previous Experience and Objectives

Since unity of research and teaching is important to us, we began integrating smaller units on MLM into the Master's program in information systems about ten years ago. The emphasis was on a more analytical approach. To this end, characteristic weaknesses of the traditional paradigm were presented and explained with reference to the underlying language architecture. Subsequently, the concepts of multi-level language architectures were introduced step by step and demonstrated with the version of XModeler$^{ML©}$ available at the time. The instructional videos on the LE4MM pages illustrate this approach (see, e.g., https://www.wi-inf.uni-due.de/LE4MM/motivation/#oo-limitations, https://www.wi-inf.uni-due.de/LE4MM/prospects/#new-paradigm).

The teaching materials were carefully prepared and structured in a comprehensible way. In addition, the corresponding unit of the lecture was supplemented by a tutorial in which smaller example models were developed. Unfortunately, the result of our efforts did not meet our expectations. Instead of the enthusiasm that we had hoped to arouse, the majority of students showed little interest in the potential of the new paradigm.

979-8-3315-7990-6/25/$31.00 ©2025 IEEE
DOI 10.1109/MODELS-C68889.2025.00106

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

A. Prerequisite: Appropriate Understanding of Object-oriented Modeling

As we grappled with the question of how we could teach MLM more successfully, it gradually became clear to us that even the introduction to modeling with UML, which has long been an integral part of the Bachelor's program, was not bringing the desired success. Quite a few students obviously struggled with understanding basic concepts. We therefore saw the sobering experiences associated with teaching MLM in the context of the problems that students have when learning object-oriented modeling with UML. Against this background, we decided to rethink our approach to teaching MLM. It seemed reasonable to assume that a holistic approach that focuses on MLM in the narrow sense but also covers the introduction to conceptual modeling, which is often done with UML, is more suitable.

The difficulties our students had with the design of object models correspond with the results of various studies. They had problems dealing with abstraction appropriately. In particular, they sometimes found it difficult to differentiate between classes and objects. We saw one reason for this in the fact that the focus in teaching was primarily on classes. Objects on M0 were usually not accounted for because they lack the level of abstraction that one typically, and for good reasons, associates with modeling. To overcome this narrow focus on classes, we decided to also include the design of instance models in the course. Students should aim to carry out a conjoint analysis of class and instance diagrams to foster an appropriate understanding of the relationship between classes and objects. However, the UML class modeling tool, which we were using at the time, did not allow integration of object and instance diagrams. Therefore, students could not see how changes to object models would affect the corresponding instance models – a limitation that also existed for all other UML modeling tools known to us.

The language architecture on which XModeler$^{ML©}$ is based, as well as the multi-level language FMML^{X} [6], offer a favorable prerequisite for realizing tight integration of object and instance models. Unlike conventional UML tools, classes in XModeler$^{ML©}$ are implemented on M1 and objects on M0. In this way, both models can not only be integrated and displayed together, but the tool also allows for the execution of objects. In order to use this advantage for teaching without burdening students with familiarizing themselves with a MLM tool, we developed a version of XModeler$^{ML©}$, called UML-MX$^{©}$, that only included common UML concepts, but no specific multi-level concepts [7]. Initial experiences with the use of UML-MX$^{©}$ in teaching have been encouraging [8]. Not only did the students apparently find it easier to distinguish between classes and objects, they were also motivated by the fact that instance models can be executed in the tool, which made the connection between model and program clearer to them.

In order to support a seamless transition from UML to MLM, we integrated the UML editor with XModeler$^{ML©}$, so that users can switch between the two modes [9]. This is also important for another reason. Experiences gathered from the introductory course on MLM suggest that students find it sometimes easier to follow a bottom-up path for developing a multi-level model. In previous versions of XModeler$^{ML©}$ the support for this was limited. The current implementation allows to change the class of every class in a model, thus enabling to add further levels to an existing model.

B. Teaching MLM: Gradual Increase of Abstraction and Student Feedback as Success Factors

The challenges we faced with teaching MLM and the encouraging experience we made with UML-MX$^{©}$ lead to the following hypotheses:

- A problem-oriented approach is particularly promising for making multi-level modeling more accessible to students.
- To avoid the confusion caused by the introduction of a new language paradigm, it is advisable to expand traditional object-oriented modeling into multi-level modeling gradually.
- This should go hand in hand with developing multi-level models from bottom-up in addition to a top-down approach.
- A modeling tool that supports bottom-up development and allows the execution of models is a critical success factor.

However, it turned out that accounting for these hypotheses alone is not sufficient since the students' learning success was compromised by another obstacle. Feedback from students and analysis of modeling assignments indicated that individual learning paths vary to a substantial degree. At the same time, many students are reluctant to ask questions or to answer questions. Therefore, the opportunities to provide students with individual support were limited. Based on the lessons we had learned and the positive effect the introduction of UML-MX$^{©}$ had on the motivation of students, we decided to redesign both the introduction to object-oriented modeling and the introduction to MLM. We then realized that the redesign of both courses – more precisely, of major parts of these courses – showed, despite all the differences, clear structural similarities. It also became clear that both the use of the modeling tool as well as of the learning units we offered as part of the UML-MX$^{©}$ environment suggest to have students' individual learning paths monitored by the tool in order to draw conclusions for adapting the teaching content.

C. Focus and Objectives

Against this background, the original intent of redesigning courses mutated into a project that seemed to be a suitable contribution to MLM research in various respects. First, it describes a use case for which MLM seems particularly suitable. Second, the design of a course for the introduction of MLM would serve as a contribution to the important discussion on objectives and methods for teaching MLM. To address these two aspects, this paper presents an executable

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

multi-level model that supports designing, monitoring, and managing courses. It can be instantiated into a various kinds of courses and, if required, adapted to more specific needs. In addition, it includes specific instantiations that describe the prototypical design of an introductory course on MLM. They also include examples that illustrate the support for monitoring and managing a particular course. In addition, the paper describes how support for bottom-up development of multi-level models is implemented in XModeler$^{ML©}$.

III. Course Planning and Implementation as Projects

Planning a course involves a number of tasks. They include, among other things, the definition of learning objectives, the division into subject areas, the selection of required resources, the design of teaching content, and the selection of teaching methods. This process is characterized by considerable contingencies. They concern prior experience, students' motivation, the uncertainty about how effective certain explanatory approaches are, as well as the time required to achieve the learning objectives. In order to reduce the risk that is implied by these contingencies, lecturers need a certain degree of freedom of choice in order to react to specific problem constellations that may arise during the implementation of a course. In addition, the experience gathered with each particular course implementation should serve as feedback to revise the design of a course.

This constellation is reminiscent of a common pattern of project planning and implementation. Although projects are often described as unique, many companies have to manage projects of a certain type. In such cases, it is advisable not to plan a project from scratch each time, but to draw on existing project knowledge. Project repositories such as those used by consulting firms are an example of how to organize and (re-) use this kind of knowledge. A project repository provides descriptions of reference projects of a certain kind. If a project manager has to plan a new project, she would look up the repository for a suitable reference project and adapt it to peculiarities of the new project. Once a project has been completed, any knowledge gained in the project is stored in the repository in the form of a further project description. In this way, a contribution is made to company-wide knowledge management, which promotes joint learning and counteracts redundant extra work (see, e.g., [10]). However, often, these project-focused knowledge management systems make use of a copy-and-paste approach, which creates a serious threat to their consistency.

The approach of choice to plan a project is the use of a method. A method consists of some kind of linguistic structure, e.g., a technical language or a conceptual model that serves the representation of the targeted domain, including required resources and objectives, and a corresponding process model. Multi-level modeling is especially promising for this purpose, since it allows integrating models of methods with models of particular method applications. Thus, it provides a powerful foundation for tools that integrate support for method engineering with the management/monitoring of corresponding projects [11]. Against this background, we decided to develop an executable multi-level model that represents an adaptable method for designing courses and, at the same time, can be used to support monitoring and management of particular courses. Fig. 1 illustrates the idea. Note that the levels shown in the figure do not necessarily represent levels of the multi-level model to be developed. In particular, the top level is not mandatory. The hierarchy illustrates that higher levels include knowledge that does not only serve the specification of objects at the level directly below, but also objects at levels further down the hierarchy. In addition, it indicates how languages and models may change over time through feedback from lower levels. The lowest level serves the representation of particular courses such as a course on MLM in the fall semester of 2025. It is created gradually over the course of a semester. In the sense of the conceptualization we chose, the implementation of a specific course is seen as a project that represents an instantiation of a corresponding project type above it, that is, the implementation of a corresponding method for designing courses of a certain type. Accounting for particular executions of courses within the multi-level model allows not only to reuse experiences over multiple course iterations, it furthermore supports monitoring the progress of the current course as well as of all its attendants (see Sec. VI-C).

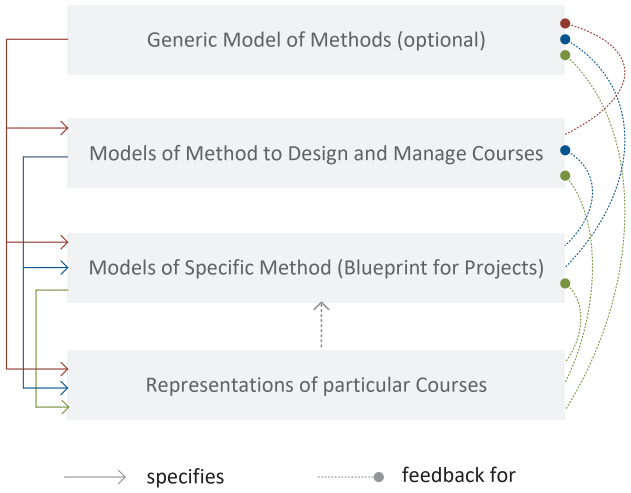

Fig. 1. Illustration of Language and Model Hierarchy

The design of a multi-level model recommends the use of a modeling method (not to be confused with the method for designing courses). We decided to combine a top-down approach that in part builds on the multi-level model for method engineering [11] with a bottom-up approach. It follows the guidelines proposed in [12]. The bottom-up approach focuses on the management and design of courses on multi-level modeling.

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

IV. Assumptions and Requirements

In order to develop a method for teaching MLM, it is necessary to first clarify general learning objectives. In doing so, we assume a standard that we consider appropriate for a Master's degree in business informatics or computer science. The following objectives are based on the assumption that enabling a reasonable understanding of MLM requires at least four, but favorably more, lessons comprising a lecture and a tutorial of 90 minutes each. Note that the following requirements aim to yield a general outline for a course on multi-level modeling. Certain details, like the choice of covered multi-level modeling languages, are intentionally faded out.

Students should be able to understand the motivation for MLM. They should be able to explain the benefits enabled by additional abstraction and make use of them in practical applications. To do this, they must not only be familiar with the syntax and semantics of the modeling language as well as functionality of the corresponding modeling tool but also be able to make well-founded modeling decisions. To this end, students need to acquire criteria for evaluating the quality of conceptual models in general, and of multi-level models, in particular.

On a more theoretical level, they should be able to discuss the idea of a modeling paradigm, and why MLM represents a paradigm shift, which, however, does not mean completely abandoning the existing paradigm. In addition, they should be able to reflect on MLM in a wider context. In particular, they should be able to discuss the advantages, requirements, and possible pitfalls of powerful abstraction. Even if the method aims not least to get students excited about modeling, the impression should not be promoted that it is a silver bullet. Therefore, the specific challenges associated with the creation and maintenance of multi-level models need to be clarified and related to general design guidelines such as tight or loose coupling.

The design of a method for teaching MLM first requires a careful analysis of requirements. This would usually involve consulting prospective users of the method. The users of the method are initially teachers. The requirements that arise from the abilities and interests of the students are also important. However, conducting a comprehensive requirements analysis in this sense is hardly possible. Due to limited experience with teaching MLM, we assume that most teachers cannot yet express clear requirements. And if one asks students about their ideas on how MLM should be taught or learned, one is faced with the problem that most of them have no clear idea of MLM. We spoke retrospectively with those who attended a corresponding course with us about their experience. Even if no specific requirements could be derived from this, the insights we gained from discussions with students have been incorporated into our analysis.

Due to the still contingent nature of the subject, our analysis resulted in preliminary requirements that are mostly based on assumptions, which we regard as reasonable but may turn out to be wrong. To emphasize this fact, the attribute "evolutionary" in the subtitle is intended not only to express that the path to MLM requires a gradual increase in the level of abstraction, but also to emphasize that the method itself should be seen as evolutionary, since it is based on conjectures that will hopefully continue to develop over time and should lead to a gradual improvement of the method. In addition, "evolutionary" also refers to the fact that the amount of time available to teach MLM can increase, up to an independent course that extends over an entire semester. We distinguish between mandatory ("M") and optional ("O") requirements to make clear that a method for teaching multi-level modeling does not have to fulfill all requirements.

The first requirement calls for a critical reflection of traditional goals of introductory courses on object-oriented modeling, which will usually be taught using UML:

Requirement O1: A method for teaching MLM should contain guidelines for an introduction to object-oriented modeling that prepare for a subsequent course on MLM. *Assumption*: In general, the design of a course requires the definition of learning objectives. If the objectives include preparation for a subsequent course, the transition should be easier for students.

Requirement M1: Like any modeling course, a short course on multi-level modeling should combine lecture elements and exercise elements. *Assumption*: This requirement reflects, as we assume, a common pattern of teaching conceptual modeling. In case of MLM, the lecture part is important to develop an appropriate understanding of concepts, objectives, and challenges related to MLM.

The following two requirements relate to the objectives for teaching UML to prepare students for a subsequent course on MLM (also Req. O1).

Requirement O2: A course on object-oriented modeling should not only emphasize the dichotomy of classes and objects but also focus on classification/instantiation compared to generalization/specialization. *Assumption*: Students often lack a sufficient understanding of classification, or, in other words, of the relationship between classes and objects. An initial understanding of this distinction is essential for approaching MLM.

Requirement O3: Such a course should alert students to the fact that it would sometimes be useful if classes could have state. *Assumption*: This insight would be a step towards realizing that UML is not the end of the story of object-oriented modeling.

The following requirements are directly related to teaching MLM.

Requirement M2: Teaching MLM should not start with an introduction to concepts of MLM languages. Instead, it should invite students to a learning experience that enables them to gradually understand the limitations of the traditional paradigm. *Assumption*: According to our experience, most students cannot understand MLM solely on the basis of a theoretical introduction. Instead, examples that raise doubts about previous certainties are suitable for developing interest in possible solutions.

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

Requirement M3: The method should support students with gaining quick wins. *Assumption*: When learning leads to success, it fosters the motivation and interest of students. However, this does not mean providing students with simple assignments only.

Requirement M4: The method should support both, top-down and bottom-up development of multi-level models. *Assumption*: Many students struggle at first with a top-down approach to designing multi-level models. Providing them with the option of using a bottom-up approach is likely to foster their understanding of multiple levels of classification. For other students or students in a later phase of the learning process, a top-down approach is usually more suitable.

These requirements concern adapting/instantiating the model to support a range of different courses:

Requirement M5: It should be possible to adapt the method to specific needs. *Assumption*: Teaching takes place under different conditions and with varying objectives. In order to meet individual requirements, it is necessary that the method can be adapted accordingly.

Requirement M6: The method should promote the reuse of knowledge at different levels of abstraction. *Assumption*: The selective reuse of knowledge is likely to reduce the effort to prepare and conduct a course. Reuse may refer to general knowledge such as prerequisites of a bottom-up approach, but also to more specific knowledge such as particular assignments, examples, or references to literature.

The following requirements focus on support for monitoring students' learning paths and on gaining feedback to improve teaching.

Requirement M7: A method for teaching MLM should provide different learning paths to facilitate learning processes that fit individual skills. *Assumption*: Students have more or less difficulties learning MLM. The ability of students to understand the concepts of modeling can vary considerably. It is therefore helpful if different learning paths are offered.

Requirement M8: A method for teaching MLM should support the analysis of student learning processes. *Assumption*: We still know little about how students develop an understanding of the additional abstraction enabled by multi-level language architectures – or struggle to do so. Therefore, teaching should in part be regarded as a research process that aims at improving students' learning experience. Among other things, this suggests to monitor individual learning paths.

Requirement M9: The method should help teachers document and evaluate the experience gained during a course in a structured way. *Assumption*: Teaching MLM still happens in an experimental model that calls for accompanying research. Time is usually short, which is why appropriate support is particularly important.

In addition, there are requirements that reflect the goal of placing MLM in a larger context to promote a deeper understanding.

Requirement O4: The method should account for the theoretical foundation of MLM with respect to the design of software systems. *Assumption*: The additional abstraction provided by multi-level language architectures enables the design of software systems that feature a higher degree of reuse, integration, adaptability, and integrity. However, these advantages do not come for free. Therefore, students need to understand the prerequisites for using MLM successfully – and how to avoid pitfalls of inappropriate design decisions.

Requirement O5: Even though the method is focused on modeling, it should also take into account multi-level programming languages. *Assumption*: To take full advantage of multi-level language architectures, it is useful to cover the entire lifecycle of software systems. That requires to account for the transformation into programming languages. In an ideal case, programming languages are multi-level, too, since that would avoid the loss of semantics.

Requirement O6: The method should also support teaching the importance of MLM outside of system design. *Assumption*: Understanding MLM requires understanding abstraction. To develop a deeper appreciation of virtues and challenges of abstraction, it is helpful to think more generally about abstraction to grasp its fundamental importance for human cognition.

V. Structure of the Method

The multi-level model presented in this section is primarily focused on the preparation and realization of courses on MLM. It also serves to guide the evaluation of students learning paths in order to provide for continuous improvement of the method itself (see requirement M8). In this way, the method should also indirectly support students with a methodical approach not only to learn the design of DSMLs and multi-level models in general, but also to develop a deeper understanding of the theoretical background.

As outlined in Section III, a multi-level model is especially suited for this task. However, one may regard the effort involved in such an approach as excessive. In fact, we could have described the method's components in natural language. However, using a multi-level model offers three advantages. First, it covers a wider range of possible methods. Second, it supports the consistent maintenance of these methods. Last but not least, the model can be executed, thereby forming a strong foundation for a tool that not only supports the construction of teaching methods but also the execution – and evaluation – of concrete applications of these methods. Note that the complete multi-level model may be difficult to understand at first due to its sheer size. The model should thus be accompanied by multiple views that increase the comprehensibility for the different stakeholders involved. We implemented some example views that can be accessed when opening the model within the XModeler$^{ML©}$.

A. Core Concepts of the Method

We pointed out above that a method consists of a language that contains concepts for structuring the respective object and a corresponding process model. The core concepts of the method presented here are illustrated in Fig. 3. Their description should make it easier to understand the section of the model shown in Fig. 2. A sequence of *learning stages*

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

represents the process model or, in other words, a *learning path* that is prototypical for a certain course type (represented by instances of classes specialized from `Course`). The concept of a learning stage is represented at L2 by the class `LearningStage`. It can be instantiated into classes that represent types of learning stages at L1. A learning stage at L1 may be defined as optional. A learning path at L1 must not be cyclic, which is expressed by the constraint `noCyclicLearningPaths`. This is, in principle, also the case for learning paths at L0. However, there may be reasons to temporarily go back to an earlier stage.

We distinguish between two different types of courses: *lectures* and *tutorials*, which are modeled as specializations of the abstract class `Course`. We assume that this distinction of course types applies to many universities. Further course types could be added simply by adding further specializations to `Course`. A course is taught through a series of course sessions (represented by objects at L0), each held on a specific date. Each course session must relate to at least one learning stage.

Each learning stage may have multiple *learning objectives* (class `LearningObjective`) associated with it, which also serve to assess learning success at the end of a stage. There are *exercises*, which are done within a course. Other exercises require tool support. They can be done by students at any time as long as they meet a given deadline: a *construction* exercise requires students to create or complete a model. A *selection* exercise comprises multiple choice tasks. The constraint `correctCourse` ensures that a particular course exercise is assigned only to learning stages that refer to the correct course for the exercise. If tool-based *learning units* (class `ToolLearningUnit`) are available, they are a further contribution to promoting individual learning paths, since they allow students to repeat them as often as they like.

The key concepts are represented through a multi-level model which (1) ensures that the semantics of the domain concepts are enforced as specified and (2), by being executable, allows to calculate the values of derivable attributes. The model has become rather extensive and may seem complicated at first glance. Indeed, the format of the paper does not allow for a complete representation of the model in one figure. The diagram in Fig. 2 shows an excerpt of the model that illustrates the core concepts of the method. The model was created using FMML^{X}. As FMML^{X} has already been described in detail elsewhere, we limit ourselves to referring the reader to the respective literature, e.g., [13].

B. Support for Course Monitoring and Management

Course management and course monitoring chiefly depend on objects at L0 that represent the execution of a particular course. For example: Students submit solutions to a course exercise. Corresponding assessments (either *passed* or failed) are stored in objects that represent a particular exercise at L0. This is expressed through the association `submits` between `Student` and `StudentExercise`. The results of student exercises are used for monitoring learning stages and individual student progress (stored in descendants of `LearningStage` and `StudentExercise`, both at L0.) `StudentExercise` is a contingent-level class [14], meaning that it may act as a L3 and L4 class depending on its context. As an L4 class, it serves to instantiate `CourseExercise`. As an L3 class, it serves to instantiate `LearningUnitExercise` (learning units are discussed in Sec. VI-B).

VI. Demonstration of the Method

The complete model comprises 55 classes at L1 and higher and more than 60 objects at L0. Therefore, it is too extensive to be displayed or explained in detail in this paper. The complete model can be downloaded at www.le4mm.org/multi25, where we also provide a screencast that helps navigate the model. In order to illustrate the possible use of the model for planning and executing courses, selected use scenarios are described below.

A. Focus on Course Planning and Course Management

Planning a course from scratch starts with a preliminary design of the learning path, represented by learning stages. The example instantiation illustrated by Fig. 4 is a reflection of assumptions presented in Secs. II and III. Subsequently, the required learning resources (`LearningMaterial`) and exercises are specified.

If a certain course type is already represented in the model, there are two options. In the simplest case, the existing course type is a perfect fit. Then, the execution of the particular course requires only the instantiation of existing classes and the reuse of objects that represent resources or learning units. In case a course type has to be adapted, it needs to be checked whether the required modifications are monotonic extensions of the existing model. In that case, the adaptation is restricted to these modifications. If they are not monotonic, one needs to create a copy of the model first. Otherwise, access to historic course data would be compromised – unless the system that implements the multi-level model features a sophisticated version management, which is not the case for XModeler$^{ML©}$.

Course assignments are a crucial instrument to assess a student's learning progress. They also support students in studying the teaching content and may support student self-evaluation. L2 instances of `CourseExercises` represent meta types of exercises that may be reused throughout semesters. An example exercise could comprise the development of a DSML for managing a restaurant. Particular exercise types are represented by L1 descendants of `CourseExercise`. Unlike L2 meta types, the assignment of L1 exercise types may differ between semesters. L1 exercise types may be included in a learning stage of a particular semester only if the L2 meta type is linked to a semester-invariant learning stage on L1. This is expressed through the association dependency [3] between the `includes` and `may_include` associations.

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

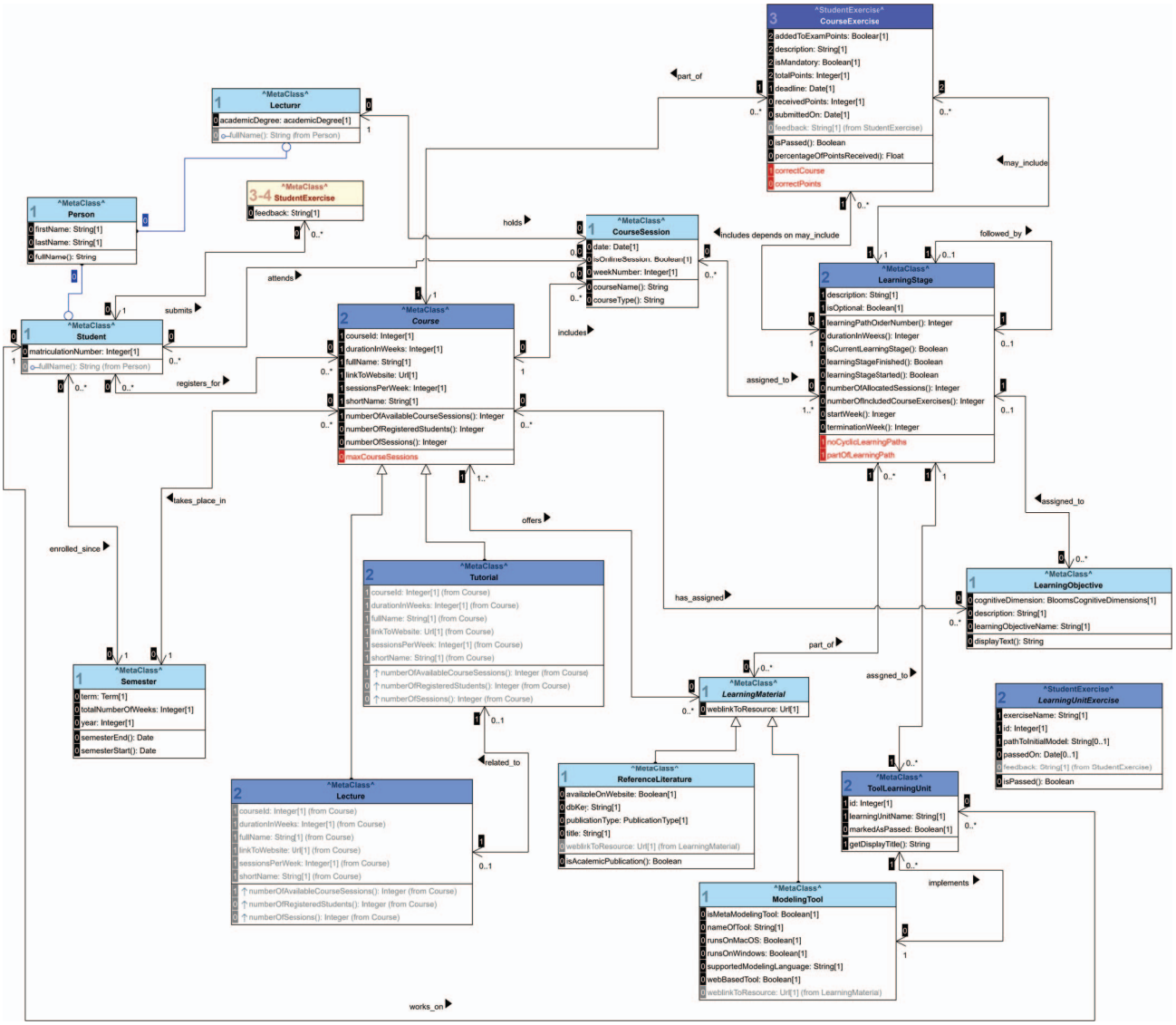

Fig. 2. Excerpt of the multi-level model of the method illustrating generic domain concepts.

To capture lecturer notes, an additional L1 class `LecturerNote` is added in the model (not displayed in Fig. 2). It is an abstract class that specializes into two types of notes: group notes and individual notes. Individual notes refer to particular students. Each note consists of a problem statement, a problem context, the date the note was entered into the system, and may additionally list lessons learned. Impressions of a lecturer can thus be captured by instantiating one of the lecturer note types.

Even though the model is executable, its presentation by a graphical diagram is not appropriate for using it as an application. That would require adding a GUI which facilitates browsing the model and changing its state. The XModeler$^{ML©}$ offers rudimentary support for generating GUIs from a model. Fig. 5 shows a corresponding simple GUI that is focused on objects for course management.

B. Outline of Tool-Supported Learning Paths

The presented model offers three approaches to provide support for individual learning paths. First, tool-based learning units allow students to learn at their own pace. The current version of XModeler$^{ML©}$ includes ten learning units aimed at fundamentals of object-oriented modeling [8]. We further propose fifteen learning units that align with the learning stages presented in Sec. VI-A. Details about these specific tool-supported learning units can be accessed through the complete model available for download at our website.

A learning unit must provide a detailed description of its theoretical background. It may be supplemented with an example model that illustrates the core lessons of the learning

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

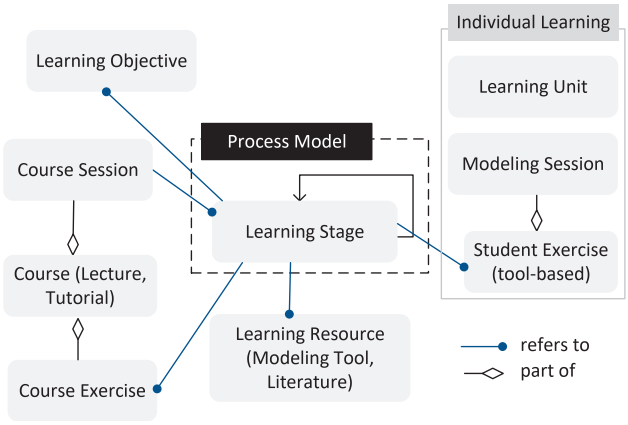

Fig. 3. Illustration of core Concepts of the Method

unit. Learning units comprise also rather simple tasks. This is to foster students' motivation by allowing for early feelings of success.

C. Monitoring Student Progress

Student progress metrics are accounted for through the L2 class `StudentProgress` that is not displayed in Fig. 2 but is included in the complete model available online. The class is associated with a specific student and a particular learning stage. L0 descendants of `StudentProgress` calculate progress metrics for each learning stage. On L1, these metrics can be aggregated to produce a synthesized view of a single student's progress. Fig. 6 shows two snapshots of the progress a student has made with respect to particular learning stages. L0 descendants of `StudentProgress` violate the constraint `learningStageProgress` if a learning stage is not considered passed by a student but has already been finished within the course.

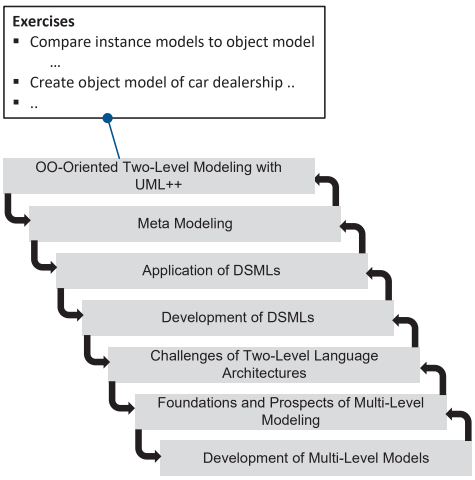

Fig. 4. Proposed learning path for teaching MLM.

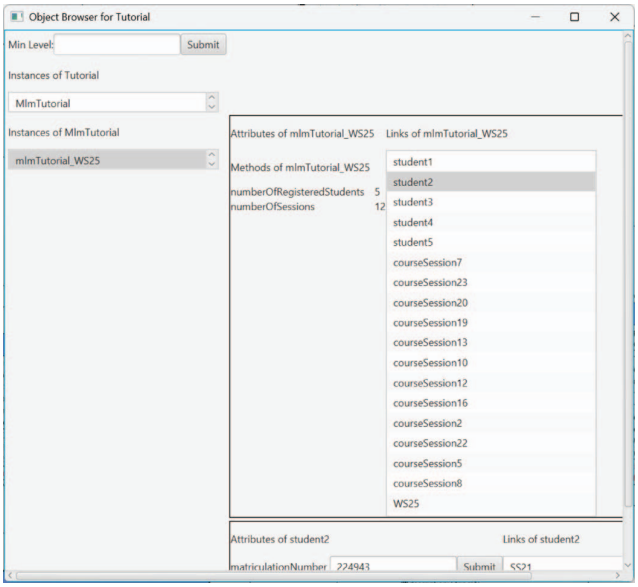

Fig. 5. Example of generated GUI

A further instrument to support the assessment of learning success is provided by statistics calculated by instances of `LearningStage` at L0. They comprise, e.g., the number of passed exercises, the ratio of passed exercises, and the number of students who made use of tool-based learning units.

VII. Preliminary Evaluation and Future Research

The evaluation of the model presented comprises three aspects. First, the model is evaluated against the requirements (see Tab. I). Next, the creation and maintenance costs are analyzed in relation to the expected benefits. Finally, the proposed method for teaching MLM needs to be evaluated.

The design of multi-level models faces specific challenges. They include the question of how to decide whether or not an additional level is required, and, related to that, how the highest level within a model can be determined. Answering these questions is crucial for adaptability and the range of reuse (Req. M5). To this end, it is first required to check whether a certain class may exist in different varieties, for example, are there different kinds of course types? In cases where the difference concerns only properties (attributes, associations,

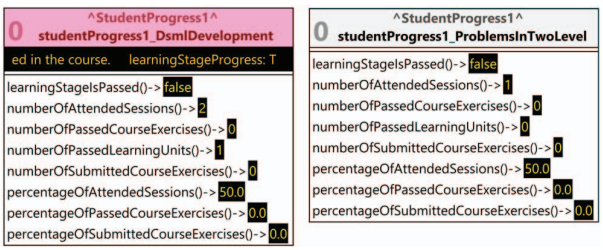

Fig. 6. Example of two facets of progress made by one student.

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SATISFACTION OF REQUIREMENTS

Req.		Comments
O1	+	These guidelines exist, but are not presented in the paper. They are supported by features of UML-MX©.
M1	+	Both types of courses are accounted for in the model.
O2	○	Addressed by teaching material and (tool) exercises; to be extended.
O3	+	Comprehensively addressed by example models, teaching material and exercises; to be extended.
M2	+	Accounted for by the sequence of learning stages and corresponding exercises.
M3	+	Supported by (tool) exercises in general, and in particular by XModeler$^{ML©}$that allows for executing models.
M3	−	The higher levels of the models allow for instantiations to various kinds of courses. However, since the requirements analysis was mainly focused on a course on MLM, it is likely that further specific properties need to be added.
M4	+	Accounted for with various exercises and supported by XModeler$^{ML©}$.
M5	○	The top level classes allow for instantiations that represent a range of courses. However, since requirements analysis was focused on teaching MLM, accounting for peculiarities of other course types may require adapting top level classes.
M6	+	The model is designed to represent the relevant knowledge at the highest level of abstraction, which allows for its reuse at lower levels.
M5	○	Supported by tool exercises the students can repeat as often as they like. Additional support is subject of future research.
M7	○	Students' learning processes can be monitored in part. That concerns their work with tool exercises, and any model they create with XModeler$^{ML©}$.
M8	+	In part addressed by tool exercises. Also supported by monitoring the creation of models in XModeler$^{ML©}$.
M9	○	Lecturers can enter structured feedback concerning problems, success factors related to learning stages and to particular students.
O4	+	Addressed by learning stage and teaching material (literature).
O5	+	Addressed by learning stage and supported by FMML^{X}, which is executable.
O6	+	Addressed by learning stage and teaching material (literature).
		+ fully implemented ○ partially implemented − not yet implemented

operations), pure generalization is preferable. However, if the various types differ in terms of their states, a supplementary classification makes sense.

To take advantage of the additional abstraction allowed for in multi-level models, and not to suffer from it, it is essential that classification relationships (we rather speak of "intrinsic classification" to express that it is not pure classification) are invariant [12]. Whether or not an abstraction is considered invariant chiefly depends on the available knowledge, not only about the domain as it is but also about its possible future development.

In this respect, the evaluation of the presented model is subject to two limitations. First, we have not performed a comprehensive analysis of existing courses. Rather, our analysis was largely limited by the types of courses known to us. It can be assumed that there are other courses with specific peculiarities, for example in the humanities or in disciplines where practical teaching takes place in laboratories or hospitals. This is not a problem as long as these special features can be realized by monotonic extensions to the existing model, for example by specializing **`Course`** or by adding further properties to this class. If it turns out that these measures are not sufficient to adequately represent other types of courses, additional levels may be required. While XModeler$^{ML©}$ supports this kind of bottom-up extension, it may require further adaptations within affected parts of the model hierarchy. Whether or not this is likely to happen cannot be decided without a comprehensive analysis of the entire domain.

Second, while the domain of higher education has been rather static for a long time, there are indications that dynamic, possibly radical, changes are to be expected. They are related to the growing importance of (commercial) online platforms, the support of individual learning processes through AI, and the increasing attention paid to *micro-credentials*. The design of the models presented did not account for these changes. Too little is yet known about what the expected changes will look like.

The design of the model required considerable effort, which is quite typical for multi-level models. Ideally, this effort is offset by a considerable gain in reuse and integrity for those who instantiate new course types and courses. Nevertheless, one might ask whether it is not too much of a good thing to develop a multi-level model where a textual description would do. To refute the first objection, it is important to realize that the model does not only serve to represent certain course types and their instances. Instead, it is part of a method for developing course types. In addition, since it is executable, it also supports managing and monitoring courses. We believe that the benefits of this support should not be underestimated. After all, most university teachers not only know little about their students, but also have limited knowledge about the effectiveness of their teaching. Therefore, the support for individual learning paths and partially automated student feedback are promising features to improve teaching. The usefulness of such a system also depends on the degree to which lecturers actually record problems, solutions, and success factors that they have noticed during the implementation of a course. With respect to the notorious lack of time researchers suffer from, this is a serious concern, which recommends appropriate incentives and a culture that rewards work colleagues will benefit from.

As far as the proposed method for teaching MLM is concerned, our preliminary evaluation is restricted to experience gathered over a period of about ten years. The method addresses various problems encountered in the past. As far as we can tell, it contributed to improving the efficiency of learning. Nevertheless, teaching multi-level modeling is still characterized by considerable contingencies, a fact that is addressed by monitoring and feedback features provided by the model.

There are various options for future work. They include further development of the model, e.g., by covering additional

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.

course types or tool exercises. It would also be important to expand support for individual learning paths. This includes, for example, the management of individual workflows to support students in their learning process. The introduction and review of milestones would be a possible option, too. To provide students with more comprehensive support, workflows related to individual courses could be combined into an overarching workflow (or project) that accompanies and monitors students throughout their entire studies.

VIII. Conclusions

The multi-level model presented in this paper is intended to support the design, management, and monitoring of courses at universities. The focus was primarily on teaching MLM. In this sense, the model is also intended to stimulate discussion about the content and methods of teaching MLM. To promote the urgently needed dissemination of MLM, a community-supported reference course would be helpful. Such a discussion would also offer the opportunity to develop a jointly maintained repository of tasks and models that can be used in teaching.

However, to improve teaching, it would be wrong to limit oneself to the support of such a model or software in general. Learning implies interest, which often has to be awakened first. This requires teachers who are enthusiastic about the content they teach and who can pass this enthusiasm on to their students. Open exchange with students, characterized by mutual trust, is of central importance for this. In an ideal case, students perceive attending a course not as a necessity to pass an exam, but rather as a rewarding, if not exciting, experience. It is doubtful whether traditional teaching formats, especially lectures, provide a suitable atmosphere for promoting this. Therefore, research to improve teaching cannot ignore a critical examination of current teaching methods as well as the organization of the university.

References

[1] M. A. Jeusfeld and U. Frank, "Unifying multi-level modeling: A position paper," in *24th International Conference on Model-Driven Engineering Languages and Systems*, (Piscataway, NJ), pp. 536–540, IEEE, 2021.

[2] T. Kühne, J. P. A. Almeida, C. Atkinson, M. A. Jeusfeld, and G. Mezei, "Field Types for Deep Characterization in Multi-Level Modeling," in *ACM/IEEE Models Conf.*, pp. 639–648, IEEE, 2023.

[3] T. Kühne and P. Maier, "FMMLx and DLM – A Contribution to the MULTI Collaborative Comparison Challenge," ACM Digital Library, pp. 800–809, ACM.

[4] A. Lange, U. Frank, C. Atkinson, and D. Töpel, "Comparing lml and fmml x : A contribution to the multi collaborative comparison challenge," in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems companion*, (Piscataway, NJ), pp. 669–678, IEEE, 2023.

[5] U. Frank and T. Clark, "Multi-level design of process-oriented enterprise information systems: 10:1-50 pages / enterprise modelling and information systems architectures (emisaj), vol. 17 (2022)," 2022.

[6] U. Frank, "Multi-Level Modeling: Cornerstones of a Rationale," *Software and Systems Modeling*, vol. 21, no. 1, pp. 451–480, 2022.

[7] U. Frank and P. Maier, "UML-MX: Boosting Power of Object-Oriented Modeling and Enriching User Experience," in *Enterprise Design, Operations, and Computing - CBI. Vienna* (M. Kaczmarek-Heß, ed.), 2024.

[8] P. Maier and T. Schwarz, "Uml++: Enhancing student learning of object-oriented modeling through executable objects," in *MODELS Companion '24: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, pp. 107–114, 2024.

[9] P. Maier and D. Töpel, "Xmodelerml v3: Integrating executable uml with a multi-level language engineering, modeling, and execution environment," in *ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2025.

[10] R. Arar, G.-J. Ren, S. Jiang, V. Becker, L. Huang, and E. Y. Liu, "Applying user-centered design to business modeling: Cbm.next as a case study," in *Proceedings of 2018 IEEE 20th Conference on Business Informatics (CBI)*, vol. 02, pp. 164–169, 2018.

[11] U. Frank, "Specification and management of methods - a case for multi-level modelling," in *Enterprise, Business-Process and Information Systems Modeling* (J. Gulden, I. Reinhartz-Berger, R. Schmidt, S. Guerreiro, W. Guédria, and P. Bera, eds.), Lecture Notes in Business Information Processing, pp. 311–325, Cham: Springer, 2018.

[12] U. Frank, "Prolegomena of a multi-level modeling method illustrated with the fmml^{x}," in *Proceedings of the 24th ACM/IEEE International Conference on Modell Driven Engineering Languages and Systems: Companion Proceedings*, IEEE, 2021.

[13] U. Frank, "The Flexible Multi-Level Modelling and Execution Language (FMML^{X}) – Version 2.0," ICB Research Report 66, University Duisburg-Essen, 2018.

[14] U. Frank and D. Töpel, "Contingent Level Classes: Motivation, Conceptualization, Modeling Guidelines, and Implications for Model Management," in *Proceedings of the 23rd Models: Companion Proceedings* (E. Guerra and L. Iovino, eds.), (New York, NY, USA), pp. 622–631, 2020.

Authorized licensed use limited to: University Duisburg Essen. Downloaded on November 27,2026 at 12:02:45 UTC from IEEE Xplore. Restrictions apply.