XModeler$^{ML©}$: Eine wandlungsfähige Entwicklungs- und Ausführungsumgebung auch für die Lehre

Unterstützung eines interaktiven Zugang zur objektorientierten Modellierung, dem Entwurf von DSML und der Erstellung von Mehrebenenmodellen

Ulrich Frank[1], Pierre Maier[1]

Abstract: In diesem Beitrag wird die Sprachentwicklungs-, Modellierungs- und Ausführungsumgebung XModeler$^{ML©}$ vorgestellt. Der XModeler$^{ML©}$ erlaubt die gemeinsame Entwicklung von Sprachen, Modellen und ausführbaren Programmen. Nachdem zunächst an einen Einsatz in der Lehre nicht gedacht war, wurde bald deutlich, dass die gemeinsame Bearbeitung von Modellierungssprachen, Modellen und deren Instanzen auch in der Modellierungslehre gewinnbringend eingesetzt werden könnte. Deshalb wurde zunächst eine dedizierte UML-Version des XModeler$^{ML©}$ entwickelt, in der Konzepte der Mehrebenenmodellierung ausgeblendet wurden und weitgehend die Notation der UML übernommen wurde. Darüber hinaus wurden interaktive Lerneinheiten integriert. Später wurde der UML-Modus in den XModeler$^{ML©}$ integriert, so dass ein fließender Übergang von Objektmodellen zu DSML und schließlich zu Mehrebenenmodellen ermöglicht wurde. Der Beitrag ist vor allem darauf gerichtet, die durchgehende Nutzung des XModeler$^{ML©}$ in der Modellierungslehre darzustellen.

Keywords: Modeling tool; DSML engineering; model-based development; teaching; multi-level modeling

1 Einleitung

Ein konzeptuelles Modell sollte das Wissen über die jeweils fokussierte Domäne auf einem möglichst hohen Abstraktionsniveau repräsentieren, um Redundanzen vorzubeugen. Das gilt insbesondere für den Entwurf und die Nutzung domänenspezifischer Sprachen (DSML): Wenn als invariant angesehenes Domänenwissen nicht in der DSML enthalten ist, muss es in den mit der DSML erstellten Modellen redundant abgebildet werden. Gängige, an der MOF orientierte, Spracharchitekturen und darauf aufbauende Modellierungswerkzeuge leiden unter erheblichen Abstraktionsdefiziten [Fr22], die verhindern, vorhandenes Wissen angemessen zu repräsentieren. Darüber hinaus sind sie mit einem weiteren gravierenden Nachteil verbunden. So werden etwa die Klassen eines Objektmodells in gängigen UML-Modellierungswerkzeugen nicht als Klassen auf M1, sondern als Objekte auf M0 implementiert – eine notwendige Folge der Architektur gängiger Programmiersprachen, die nicht vorsieht, dass Klassen auch Objekte sein können. Das hat zur Konsequenz, dass

[1] Universität Duisburg-Essen, [ulrich.frank,pierre.maier]@uni-due.de

Objekt- und Instanzmodelle getrennt voneinander erstellt werden müssen (Klassen, die als M0-Objekte implementiert sind können ja nicht weiter instanziert werden), so dass sich Änderungen im Objektmodell nicht sofort auf das Instanzmodell auswirken.[2] Da Klassen im Modelleditor nicht instanziert werden können, ist eine Ausführung der korrespondierenden Instanzen nicht möglich. Um zu ausführbarer Software zu gelangen, ist deshalb eine weitere separate Repräsentation in Form von Programmcode erforderlich. Im besten Fall wird der Programmcode dabei generiert, was allerdings zu dem notorischen Problem führt, Code und Modell im Zeitverlauf zu synchronisieren [Fr22, S. 468 f.].

Diese Probleme haben die Entwicklung von Sprachen zur Realisierung von Mehrebenenmodellen („multi-level models“) motiviert [AK08; Fr14; Je19; NGS09]. Mehrebenenmodelle weisen jenseits spezifischer Unterschiede zwischen einzelnen Sprachen im Allgemeinen drei wesentliche Merkmale auf: 1. Es sind beliebig viele Klassifikationsebenen möglich. 2. Alle Klassen sind Objekte, haben also einen Zustand. 3. Die Eigenschaften (Attribute, Operationen, Assoziationen) einer Klasse beschränken sich nicht nur auf Merkmale ihrer direkten Instanzen. Vielmehr kann die instanzierung der Eigenschaften auf weiter darunter liegende Ebenen verschoben werden („deferred instantiation“). Mehrebenenmodelle integrieren also traditionelle Modelle und Modellierungssprachen wie auch ggfs. die (Meta-)Sprachen zur Spezifikation von Modellierungssprachen. Auf diese Weise können die Repräsentationsdefizite traditioneller Spracharchitekturen weitgehend überwunden werden.

Die Sprachtechnologie, auf der die hier vorgestellte Modellierungsumgebung beruht, ermöglicht darüber hinaus eine gemeinsame Repräsentation von Modellen und Programmcode. Sie basiert auf einem reflexiven, erweiterbaren Metamodell [CSW08, S. 40]. Der XModeler ist damit gleichsam eine Instanz seiner selbst und bietet die Grundlage für die Spezifikation und Implementierung einer großen Bandbreite von (Modellierungs- und Programmier-) Sprachen. Allerdings sieht der XModeler keine expliziten Klassifikationsebenen vor. In einem 2009 zusammen mit Tony Clark begonnenen Projekt [FC23] wurde das Metamodell des XModeler so erweitert, dass es die zentralen Eigenschaften von Mehrebenenmodellen auszudrücken erlaubt. Die entsprechende Mehrebenenmodellierungs- und Programmiersprache, FMML^{x} [Fr18], wurde um eine dedizierte konkrete Syntax erweitert. Die so entstandene Entwicklungsumgebung, XModeler$^{ML©}$ genannt, erlaubt neben der Erstellung von Mehrebenenmodellen (und ggfs. korrespondierender grafischer Notationen) auch deren Ausführung.

Die mit der Entwicklung des XModeler$^{ML©}$ verbundenen Forschungs- und Implementierungsaufgaben waren von beachtlicher Komplexität, so dass es mehr als zehn Jahre dauerte bis ein weitgehend vollständiger und hinreichend leistungsfähiger Prototyp bereitstand, der dann auch in der Lehre im Master-Programm eingesetzt wurde. Dabei wurde deutlich, dass der XModeler$^{ML©}$ prinzipiell auch für die Einführung in die objektorientierte Modellierung

[2] Während der Terminus „Objektmodell“ häufig verwendet wird (neben „Klassendiagramm“, ein Ausdruck, der genau genommen lediglich für eine bestimmte Form der Repräsentation steht), ist die Verwendung von „Instanzmodell“ nicht einheitlich. Mitunter wird stattdessen von „Objektdiagramm“ gesprochen. Wir meinen hier ein Modell, das Objekte auf M0 abbildet.

im Bachelor-Programm geeignet ist – nicht zuletzt deshalb, weil er die synchrone Erstellung von Klassen- und Objektdiagrammen erlaubt. Die Möglichkeit, Objektmodelle auszuführen, reichert das interaktive Lernerlebnis weiter an. Die Nutzung des XModeler$^{ML©}$ in der Lehre war somit ein zunächst nicht intendierter Seiteneffekt des eigentlichen Forschungsprojekts [FM25]. Unseres Wissens nach unterstützt kein anderes Mehrebenenmodellierungstool die Ausführbarkeit von Objekten.

Der Fokus dieses Beitrags liegt darauf, zu zeigen wie der XModeler$^{ML©}$ in der Lehre eingesetzt wird – und welche ergänzende Unterstützung dafür angeboten wird. Vor dem Hintergrund einer kurzen Analyse der Schwierigkeiten, denen sich Studierende häufig gegenüber sehen, wird zunächst der Einsatz des XModeler$^{ML©}$ im Rahmen der Einführung in die objektorientierte Modellierung dargestellt. Anhand eines durchgehenden Beispiels wird dann die Erweiterung hin zu einer DSML und schließlich zu einem Mehrebenenmodell gezeigt.

2 Häufige Hindernisse beim Zugang zur objektorientierten Modellierung

Trotz eines über viele Jahre gereiften Lehrprogramms mussten wir immer wieder die leidvolle Erfahrung machen, dass Studierende sich schwertun, grundlegende Konzepte der Objektorientierung zu verstehen und angemessen anzuwenden. Einige Studierende hatten bereits das Problem, den für ein solches Verständnis fundamentalen Unterschied zwischen Klassen und Objekten nachzuvollziehen. Zudem fehlte mitunter eine fundierte Vorstellung davon, wie ein Objektmodell zu einem lauffähigen Programm transformiert werden kann.

Ähnliche Schwierigkeiten werden seit langem von vielen Lehrenden berichtet. Eine umfangreiche Meta-Studie von Gutiérrez et al. [GGL22] synthetisiert Herausforderungen bei der Lehre von objektorientierter Programmierung aus Publikationen der letzten 30 Jahre. Dabei zeigte sich u.a., dass Studierende Objekte, Klassen, und deren Zusammenhänge häufig nicht korrekt begreifen. Um dieser Schwierigkeit, zu begegnen empfiehlt Brinda [Br03, pp. 18-19], dass zum besseren Verständnis von Objektmodellen, Studierende die entsprechenden Instanzen mithilfe eines Instanzmodells erstellen und analysieren sollen. Die Umsetzung dieser Empfehlungen sieht sich allerdings vor einer erheblichen Herausforderung, auf die bereits Moisan und Rigault mit Nachdruck hingewiesen haben [MR10, p. 49]: Objektmodelle und Instanzmodell werden in gängigen UML-Werkzeugen unabhängig voneinander repräsentiert.

Demgegenüber erlaubt der XModeler$^{ML©}$ den integrierten Entwurf von Klassen- und Objektdiagrammen, sodass die Studierenden die Möglichkeit erhalten, die Auswirkungen von Änderungen an Klassendiagrammen auf korrespondierende Objektdiagramme sofort zu sehen. Vor diesem Hintergrund haben wir 2022 und 2023 erste Übungen mit dem XModeler$^{ML©}$ in unsere Bachelorveranstaltung zur Unternehmensmodellierung eingebunden. Dieser zunächst experimenteller Einsatz des XModeler$^{ML©}$ verlief so ermutigend, dass

wir eine eigene UML-Version des Werkzeugs erstellt haben. Dazu haben wir einen auf Mehrebenenmodellierung basierten Dialekt der UML entworfen, welchen wir UML++ nennen [MS24] und mit einer dedizierten Variante des XModeler$^{ML©}$, UML-MX$^{©}$ (für „UML Modeling and eXecution“) genannt, angeboten. Um Studierende nicht zu verwirren, wurden spezfische Mehrebeneneigenschaften ausgeblendet. Die Einführung von UML-MX$^{©}$ führte zu einer ausgesprochen positiven Resonanz bei Studierenden, die durch eine begrenzte Evaluationsstudie in Teilen bestätigt wurde [MS24].

Eine weitere Herausforderung für die Lehre besteht darin, dass die Voraussetzungen und Fähigkeiten von Studierenden erheblich variieren. Das äußert sich nicht zuletzt darin, dass sich individuelle Lernpfade teilweise deutlich unterscheiden. Während die in einer Übung besprochenen Aufgaben für einige Studierende leicht zu bewältigen sind, brauchen andere wesentlich mehr Zeit oder haben gar Schwierigkeiten, überhaupt einen Zugang zu finden. Um individuelle Lernpfade zu unterstützen, wurde UML-MX$^{©}$ um interaktive Lerneinheiten ergänzt. Diese Lerneinheiten stellen zu verschiedenen Themen der objektorientierten Modellierung (wie bspw. Multiplizität von Assoziationen, Generalisierung/Spezialisierung, Spezifikation von Constraints) grundlegende Erläuterungen bereit. Darüber hinaus werden innerhalb der Lerneinheiten Beispielmodelle und Beispielaufgaben bereitgestellt.

Seit geraumer Zeit bildet der Entwurf von DSML einen wichtigen Gegenstand der Modellierungsveranstaltungen, die wir im Master-Programm anbieten – seit einigen Jahren ergänzt um eine Einführung in die Mehrebenenmodellierung. Dabei hat sich gezeigt, dass Studierende sich häufig schwertun, die grundlegende Motivation für eine Erhöhung des Abstraktionsniveaus zu verstehen. Eine ausführliche Analyse der Einschränkungen, die mit traditionellen Spracharchitekturen verbunden sind, sowie die Überwindung dieser Einschränkungen im gleichen Werkzeug stellt nach ersten Erfahrungen einen effektiven Ansatz dar, den Studierenden einen überzeugenden Zugang zunächst zu DSML und später zur Mehrebenenmodellierung zu bieten. Seit dem Release der Version 3 des XModeler$^{ML©}$ [MT25] werden daher sowohl UML++ als auch FMML^{x} im Tool unterstützt. Zwischen beiden Modi kann zur Laufzeit gewechselt werden. Außerdem wurde ein neuer „UML-Meta-Modus“ eingeführt, der reguläre UML-Klassendiagramme um vorhandene Metainformationen erweitert (bspw. Metaklasse der Klasse oder Zustand der Klasse).

3 Zu Beginn: Fokus auf objektorientierter Modellierung

Im Rahmen unseres Lehrprogramms erfolgt die Einführung in die objektorientierte Modellierung nach einer Einführung in die Datenmodellierung. Deshalb wird zunächst beispielhaft ein kurzer Vergleich zwischen Datenmodellen und Objektmodellen durchgeführt. Anschließend werden Schritt für Schritt grundlegende Begriffe der objektorientierten Modellierung vorgestellt. Dabei geht es zunächst um die Unterscheidung von Klassen und Objekten sowie um die Eigenschaften von Klassen (Attribute, Operationen, Assoziationen). Nach der Einführung einer einfachen Methode zur Erstellung von Objektmodellen, die auch Kriterien für häufige Modellierungsentscheidungen wie etwa die Unterscheidung von Attributen und

Assoziationen umfasst, beginnen die Studierenden mit dem Entwurf erster Objektmodelle. Zu diesem Zweck erfolgt eine kurze Einführung in den XModeler$^{ML©}$. Anschließend werden zunächst in Beispielen Generalisierung/Spezialisierung, abstrakte Klassen, Verkapselung, Polymorphie sowie Constraints vorgestellt.

Die folgenden Beispiele verweisen exemplarisch auf zwei Aspekte des interaktiven Lernens mit dem XModeler$^{ML©}$. Zum einen können Objektmodelle instanziert werden. Die entsprechenden Objektdiagramme werden gemeinsam mit dem zugehörigen Klassendiagramm im Diagrammeditor dargestellt. Änderungen an Klassen wirken sich unmittelbar auf das Objektdiagramm aus. Zum anderen können Operationen von Objekten im XModeler$^{ML©}$ ausgeführt werden. Der Screenshot in Abb. 1 zeigt einen Teil einer im XModeler$^{ML©}$ angebotenen Lerneinheit. Sie ist darauf gerichtet, die Studierenden für die kontra-intuitive Semantik der Spezialisierung in der objektorientierten Modellierung zu sensibilisieren, die darauf zurückzuführen ist, dass ein Objekt anders als in der Logik oder der natürlichen Sprache immer Instanz genau einer Klasse ist. Dazu sollen sie Instanzen der Klassen erstellen und den Fall bedenken, dass eine Studentin auch eine studentische Mitarbeiterin sein kann. Die dabei entstehende Redundanz sollte den Studierenden zu denken geben, was durch entsprechende Kommentare in der Lerneinheit gefördert wird. Im letzten Schritt der Lerneinheit wird Delegation als Lösung des Problems eingeführt (s. `www.le4mm.org/educational-material`).

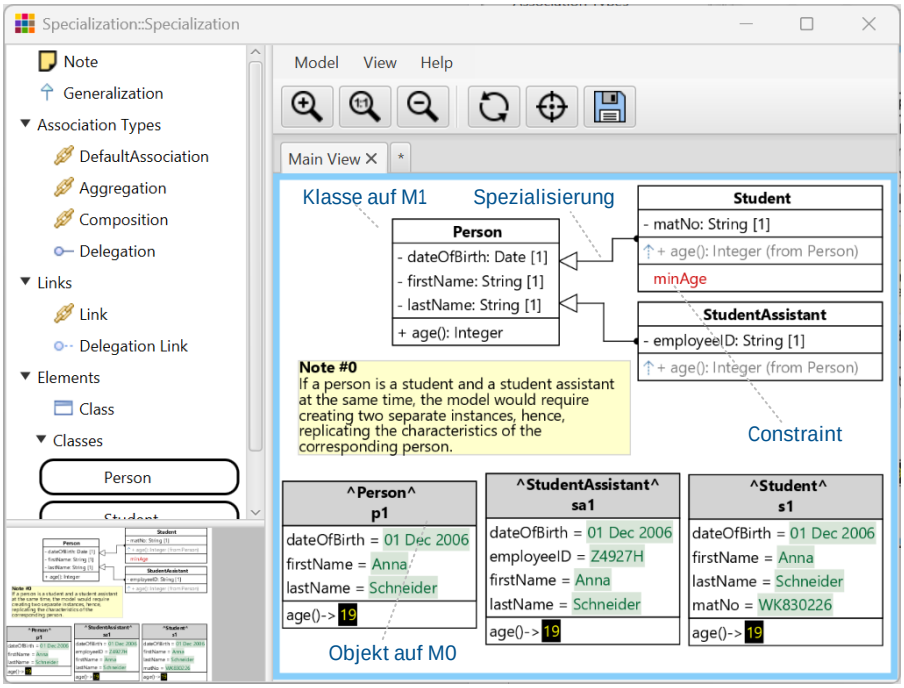

Abb. 1: Verdeutlichung der kontra-intuitiven Semantik von Spezialisierung im XModeler$^{ML©}$

Um die spätere Einführung in den Entwurf von DSML vorzubereiten, sind am Ende der Einführung in die objektorientierte Modellierung Beispiele vorgesehen, die bei näherer Hinsicht schwerwiegende Grenzen traditioneller Modellierungssprachen wie der UML verdeutlichen. Ein solches Beispiel ist in Abb. 2 dargestellt. Die dabei zu berücksichtigenden Produkte sind höherwertige elektronische Geräte im weitesten Sinn. Das Beispiel zeigt ein einfaches Objektmodell zur Erstellung von Rechnungen. Sein besonderer didaktischer Nutzen liegt darin, dass die meisten Studierenden es i.d.R. zunächst für plausibel halten.

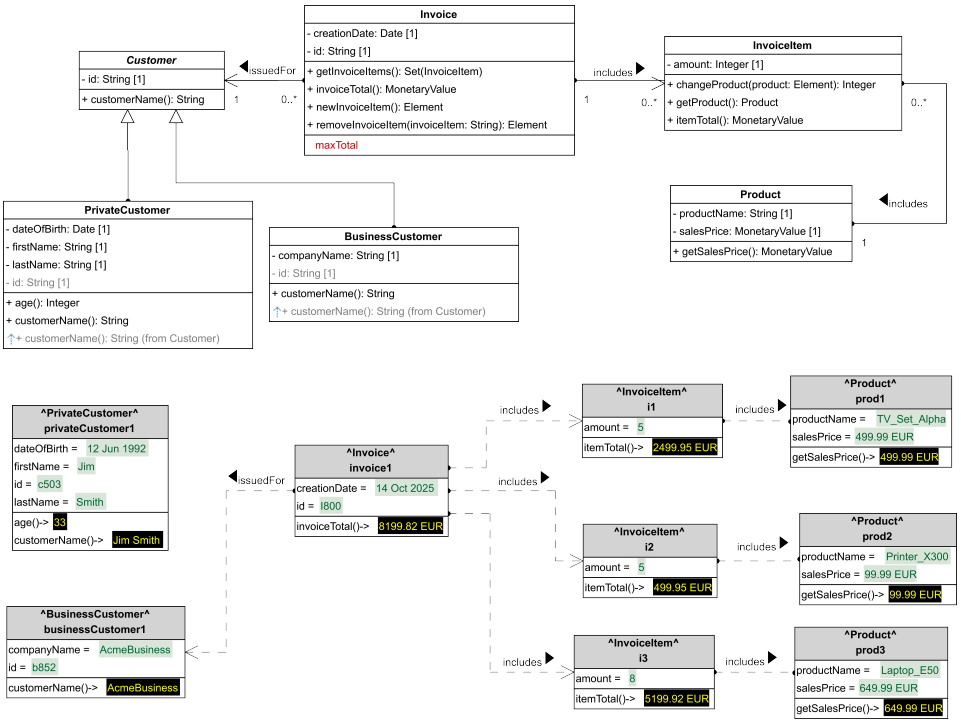

Abb. 2: Beispiel Fakturierung: Objektmodell und korrespondierendes Instanzmodell

Um auf die Einschränkungen aufmerksam zu machen, die mit einem solchen Modell verbunden sind, wird zunächst auf naheliegende Anforderungen verwiesen – mit besonderem Fokus auf die Modellierung von Produkten. So sollte jedes Gerät durch eine Seriennummer gekennzeichnet sein, die auch auf der Rechnung auszuweisen ist. Zudem sollten Preis und Gewicht für Geräte des gleichen Typs nur einmal erfasst werden und nicht redundant jedem einzelnen Gerät zugewiesen werden. Darüber hinaus sollte für jedes Produkt das Herstellungsdatum erfasst werden. Die Diskussion ist dann darauf gerichtet, zu zeigen, dass dazu eine eine Differenzierung von Produkttypen und Exemplaren erforderlich wäre, die allerdings – von möglichen Workarounds abgesehen – daran scheitert, dass dazu Metaklassen nötig wären. Zudem müsste es möglich sein, dass Klassen einen Zustand haben, da ihnen sonst keine Werte wie etwa ein Preis zugeordnet werden könnten.

4 Der nächste Schritt: Entwurf von DSML

Nachdem die Motivation und Zielsetzung von DSML durch einen allgemeinen Vergleich gegenüber GPMLs erläutert wurde, wird ein Modell der Art vorgestellt, wie es in Abb. 2 dargestellt ist – zusammen mit den ergänzenden Anforderungen. Dazu wird die naheliegende Idee diskutiert, das Wissen über Produkte in eine zunächst traditionelle DSML (die Sprache wird durch ein Metamodell auf M2 spezifiziert) abzubilden, um so Redundanz vorzubeugen und gleichzeitig die Modellintegrität zu fördern. Der XModeler$^{ML©}$ unterstützt zwei Ansätze zur Entwicklung einer DSML. Der Top-Down-Ansatz beginnt mit den Klassen auf M2, die anschließend in Modelle auf M1 instanziert werden. Der Bottom-Up-Ansatz sieht vor, neue (Meta-) Klassen einzuführen, um so ein Modell schrittweise um eine korrespondierende DSML zu erweitern. Didaktisch ist der zweite Ansatz deshalb gut geeignet, weil er anschaulich die Vorteile einer weiteren Klassifikationsebene verdeutlicht.

Wichtig ist dabei, dass die DSML-Entwicklung zunächst im Rahmen der MOF bleibt, auch wenn der XModeler$^{ML©}$ im Mehrebenenmodus verwendet wird. So ergibt sich zunächst eine sehr einfache DSML, die aus wenigen Klassen auf M2 besteht. Das Diagramm in Abb. 3 zeigt diese Klassen zusammen mit einigen Instanzen. Auch wenn es sich um eine überaus schlichte DSML handelt, kann der mit dieser ergänzenden Abstraktion verbundene Vorteil gut verdeutlicht werden. Das invariante Wissen über alle Produkttypen einer bestimmten Art wird durch die DSML repräsentiert. Es kann damit für die Erstellung von Modellen wiederverwendet werden und ist geeignet, die Qualität der Modelle zu fördern

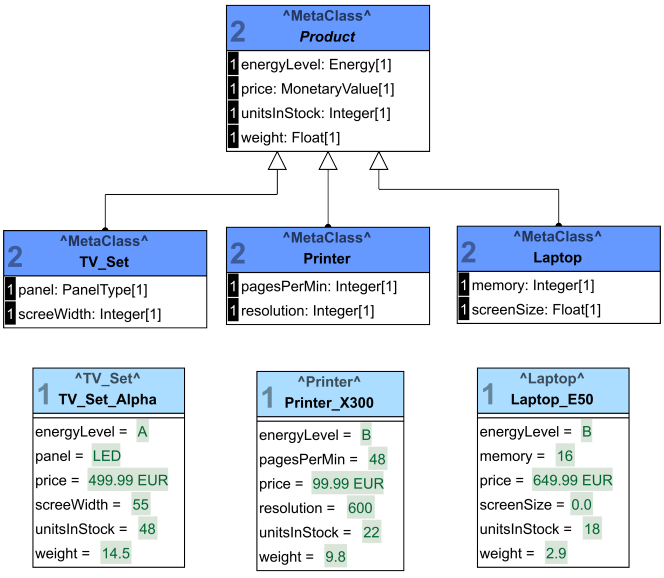

Abb. 3: Rudimentäre DSML zur Modellierung von Produkten

Nach der Erstellung weiterer DSML größeren Umfangs ist eine kritische Diskussion der MOF vorgesehen. Sie wird u.a. durch folgende Fragen angeregt: Wie kann man in einer DSML Eigenschaften von Instanzen auf M0 ausdrücken, im Fall einer Produkt-DSML ist etwa an eine Seriennummer oder ein Herstellungsdatum zu denken? Andernfalls ließen sich die mit ihr erstellten Modelle ja nicht weiter instanzieren, was ziemlich fatal für den Entwurf eines Informationssystems wäre, das ja letztlich auch der Verwaltung einzelner Instanzen dienen sollte. Wie kann man eine solche DSML für den Entwurf von Modellen zur Rechnungserstellung verwenden, wenn Klassen wie `Invoice` oder `InvoiceItem` keine weitere Klassifikation erfordern? Schließlich lässt die MOF nur Modelle zu, in denen sich alle Klassen auf der gleichen Klassifikationsebene befinden. Zudem wäre es durchaus sinnvoll, allgemeine Eigenschaften von Produktarten, die für alle zugehörigen Produkttypen gelten, wie etwa der Mehrwertsteuersatz oder eine Gefahrenklasse, den entsprechenden Klassen auf M2 zuzuordnen. Das würde allerdings eine weitere Klassifikationsebene auf M3 erfordern. Dabei gilt für korrespondierende Modellierungswerkzeuge die Einschränkung, dass alle Klassen unahbängig von ihrer konzeptuellen Klassifikationsebene als Objekte auf M0 abgebildet werden. Um zu ausführbaren Programmen zu gelangen, ist also die Generierung von Code erforderlich (wie dies in *Model-Driven Development* vorgesehen ist), was zu zwei getrennten Repräsentationen führt, deren Synchronisation im Zeitverlauf eine erhebliche Herausforderung darstellt.

Die Einführung zielt also nicht zuletzt darauf, die Ambivalenz traditioneller DSML zu verdeutlichen: Einer überzeugenden Idee – das relevante invariante Wissen über eine Domäne sollte in einer DSML repräsentiert werden – steht die Einschränkungen der MOF gegenüber, dieses Wissen angemessen auszudrücken.

5 Schließlich: Einführung in die Mehrebenenmodellierung

Die Einschränkungen, unter denen die MOF im Hinblick auf die Erstellung und Nutzung von DSML leidet, bilden die Motivation für die Einführung in die Mehrebenenmodellierung. Dazu können zunächst weitere Anforderungen formuliert werden.

In Bezug auf das Beispiel in Abb. 3 könnte etwa gefordert werden, dass für alle Produktkategorien eine Garantiedauer bestimmt werden kann oder dass Pakete von Produkten verkauft werden dürfen, falls das zwischen den entsprechenden Produktkategorien zulässig ist. Durch die schrittweise Erstellung eines Mehrebenenmodells im XModeler$^{ML©}$ können die Studierenden erfahren, wie die Einschränkungen der MOF zu überwinden sind. Dazu ist es erforderlich, in die spezifische Terminologie der Mehrebenenmodellierung einzuführen. Das Beispiel in Abb. 4 stellt ein entsprechendes Mehrebenenmodell dar. Es sieht nicht nur weitere Klassifikationsebenen für Produkte, sondern auch für Kunden vor. Es kann im XModeler$^{ML©}$ bottom-up aus dem Modell in Abb. 3 entwickelt werden.

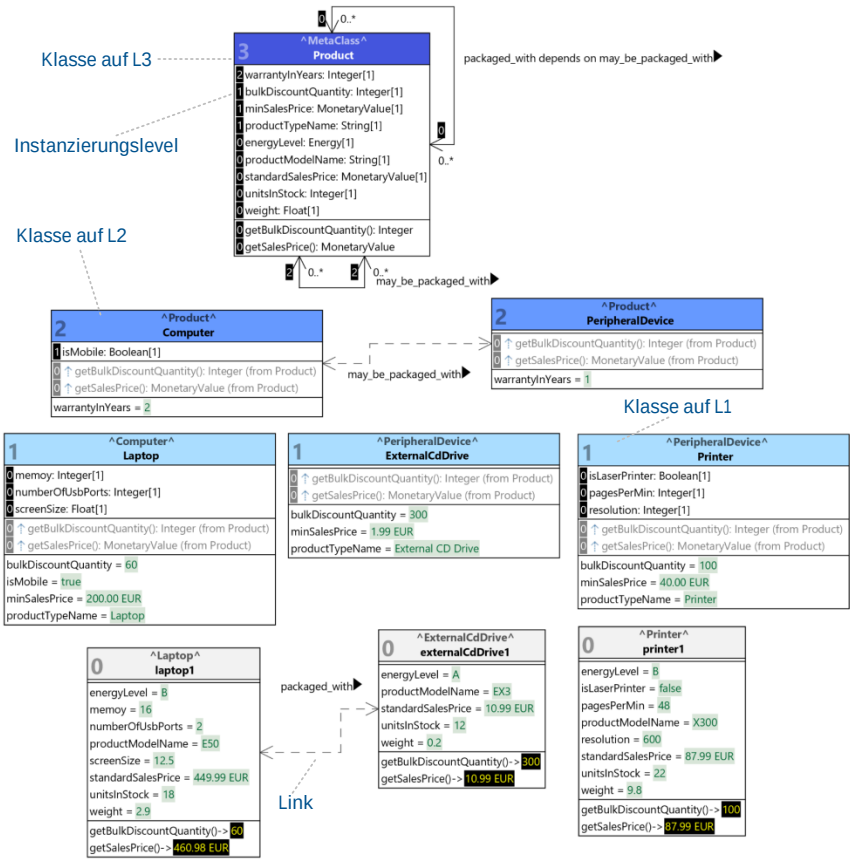

Abb. 4: Ausschnitt eines Mehrebenenmodells zur Modellierung von Produkten

6 Abschließende Bemerkungen

Anders als in bisherigen Veröffentlichungen zum XModeler$^{ML©}$ ist die Beschreibung der Modellierungsumgebung in diesem Beitrag weniger auf die grundlegende Sprachtechnologie und spezifische Leistungsmerkmale gerichtet. Vielmehr steht ihre Anwendung im Rahmen der Modellierungslehre im Vordergrund. Vor diesem Hintergrund ist der vorliegende Beitrag mit der Intention verbunden, zur Nutzung des XModeler$^{ML©}$ auch in der Lehre anzuregen – auch um auf diese Weise ergänzende Erfahrungen zu sammeln, die in die weitere Entwicklung des Werkzeugs einfließen können. Der XModeler$^{ML©}$ sowie die hier gezeigten Beispiele sind zusammen mit Videoaufzeichnungen von Lehreinheiten, Screencasts, weiteren Beispielmodellen und umfangreichen Literaturhinweisen auf den Webseiten des Projekts „Language Engineering for Multi-Level Modeling“, LE4MM, verfügbar (`www.le4mm.org` bzw. mit direktem Bezug zu diesem Beitrag: `www.le4mm.org/educational-material`).

Literaturverzeichnis

[AK08] Atkinson, C.; Kühne, T.: Reducing accidental complexity in domain models. Software & Systems Modeling 7 (3), S. 345–359, 2008.

[Br03] Brinda, T.: Student Experiments in Object-Oriented Modeling. In (Cassel, L.; Reis, R. A., Hrsg.): Informatics Curricula and Teaching Methods: IFIP TC3 / WG3.2 Conference on Informatics Curricula, Teaching Methods and Best Practice (ICTEM 2002) July 10–12, 2002, Florianópolis, SC, Brazil. Springer, Berlin und Heidelberg, S. 13–20, 2003.

[CSW08] Clark, T.; Sammut, P.; Willans, J.: Applied Metamodelling: A Foundation for Language Driven Development. Ceteva, 2008.

[FC23] Frank, U.; Clark, T.: Language Engineering for Multi-Level Modeling (LE4MM): A Long-Term Project to Promote the Integrated Development of Languages, Models and Code. In (Font, J. et al., Hrsg.): Proceedings of the Research Projects Exhibition at the 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023). CEUR, S. 97–104, 2023.

[FM25] Frank, U.; Maier, P.: How an unintended Side Effect of a Research Project led to Boosting the Power of UML, 2025, https://arxiv.org/pdf/2505.09269.

[Fr14] Frank, U.: Multilevel Modeling – Toward a New Paradigm of Conceptual Modeling and Information Systems Design. BISE 6 (6), S. 319–337, 2014.

[Fr18] Frank, U.: The Flexible Multi-Level Modelling and Execution Language (FMMLx), Version 2.0: Analysis of Requirements and Technical Terminology, ICB Research Report, Essen, 2018, https://doi.org/10.17185/duepublico/47506.

[Fr22] Frank, U.: Multi-Level Modeling: Cornerstones of a Rationale. Software and Systems Modeling 21, S. 451–480, 2022.

[GGL22] Gutiérrez, L. E.; Guerrero, C. A.; López-Opsina, H. A.: Ranking of Problems and Solutions in the Teaching and Learning of Object-Oriented Programming. Education and Information Technologies 27, S. 7205–7239, 2022.

[Je19] Jeusfeld, M.: DeepTelos for ConceptBase: A Contribution to the MULTI Process Challenge. In: Proceedings of MULTI 2019. CEUR-WS.org, 2019.

[MR10] Moisan, S.; Rigault, J.-P.: Teaching Object-Oriented Modeling and UML to Various Audiences. In (Ghosh, S., Hrsg.): Models in Software Engineering: Workshops and Symposia at MODELS 2009, Denver, CO, USA, October 4-9, 2009. Reports and Revised Selected Papers. Springer, Berlin und Heidelberg, S. 40–54, 2010.

[MS24] Maier, P.; Schwarz, T.: UML++: Enhancing Student Learning of Object-Oriented Modeling through Executable Objects. In: ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria. 2024.

[MT25] Maier, P.; Töpel, D.: 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.

[NGS09] Neumayr, B.; Grün, K.; Schrefl, M.: Multi-level Domain Modeling with M-objects and M-relationships. In: Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modeling - Volume 96. APCCM '09, Australian Computer Society, Inc., Wellington, New Zealand, S. 107–116, 2009.