Eine aufwandsbeschränkte Einführung in die modellbasierte Softwareentwicklung

Timo Kehrer, Udo Kelter

Praktische Informatik – Universität Siegen
57068 Siegen - Germany
Web: pi.informatik.uni-siegen.de
E-Mail: kehrer@informatik.uni-siegen.de,
kelter@informatik.uni-siegen.de

Zusammenfassung: Seit einigen Jahren gewinnt die modellgetriebene bzw. modellbasierte Entwicklung (MBSE) immer mehr an praktischer Bedeutung. Über die grundlegende Modellierungskompetenz hinaus sollte ein Informatikstudium daher auch Konzepte und Technologien der MBSE vermitteln, und zwar nicht nur vage verstandene Schlagworte, sondern die Kompetenz, eine konkrete MBSE-Infrastruktur praktisch einzusetzen. Aufgrund der Komplexität der Technologien ist es sehr schwierig, dieses Ziel mit beschränktem Aufwand zu erreichen. Dieses Papier stellt eine Unterrichtssequenz vor, mit der ein substantieller Qualifikationsgrad mit relativ geringem Aufwand (unter 30 Arbeitsstunden bzw. einem Leistungspunkt) erreicht wird.

1 Einleitung und Motivation

Um die Komplexität großer Systeme zu beherrschen, hat die Modellbildung als Mittel der Abstraktion und Problemanalyse längst einen festen Platz in der Informatik eingenommen [7, 11]. Dementsprechend definiert u.a. der „IEEE Guide to the Software Engineering Body of Knowledge“ [5] für den Themenkomplex Modellierung hohe bis sehr hohe Kompetenzstufen als Ziel einer Informatikausbildung an Hochschulen.
In klassischen Entwicklungsprozessen fungieren Modelle primär als autarkes Dokumentations- oder Spezifikationsmittel. Seit einigen Jahren gewinnt darüber hinaus die modellgetriebene bzw. modellbasierte Entwicklung (MBSE) immer mehr an praktischer Bedeutung; ohne sie sind manche komplexen Produkte kaum noch realisierbar. Modelle werden hier in mehreren, ganz oder teilweise automatisierten Schritten zu lauffähiger Software oder zu interpretierbaren Spezifikationen verfeinert. Hierdurch ergeben sich zwei neue Ausbildungsziele, die deutlich über die grundlegende Modellierungskompetenz, welche für klassische Entwicklungsprozesse vermittelt wird und die wir i. F. als Grundlage voraussetzen, hinausgehen:

1. Absolventen sollten zumindest die „einfache“ Nutzung einer MBSE-Technologie beherrschen und einen eigenen Eindruck von den Vorteilen und Grenzen der MBSE-Technologien gewinnen. Selbst ein solcher erster Einstieg stellt wegen der Komplexität der Werkzeuge eine Herausforderung dar und erfordert zumindest ein grundsätzliches Verständnis der Technologien.

2. In der Praxis müssen die MBSE-Technologien häufig an die konkreten Umstände angepasst werden; im Extremfall läuft dies auf die Entwicklung domänenspezifischer Sprachen (DSLs) hinaus. Hierzu ist ein noch deutlich tieferes Verständnis der MBSE-Technologien erforderlich.

Die Komplexität des Themas ist auch am Umfang von Lehrbüchern zum Thema MBSE (bzw. MDA) abzulesen: typisch sind 300–500 Seiten, s. Literaturdiskussion in Abschnitt 4. Dies entspricht 4–8 ECTS-Leistungspunkten, die allenfalls im Wahlpflichtbereich des Curriculums unterzubringen sind. Um das Thema im Pflichtbereich unterzubringen bzw. viele Studierende zu erreichen, wird eine sehr kompakte Einführung benötigt.
In diesem Papier stellen wir eine Unterrichtssequenz vor, welche die erste o.g. Qualifikationsstufe erreicht und nur 20–30 Stunden Arbeitsaufwand für die Lernenden, entsprechend ca. 1.0 ECTS-Punkten, verursacht. Die Unterrichtssequenz ist gedacht für Studierende im 3. oder 4. Fachsemester, die schon über grundlegende Modellierungskompetenzen verfügen und wenigstens zwei Modelltypen kennengelernt haben – neben Datenmodellen (i.d.R. Klassendiagramme der UML) zusätzlich einen weiteren (Petri-Netze, Zustandsautomaten o.ä.). Wir haben diese Sequenz sowohl als selbstständigen Kurs wie auch integriert in eine Softwaretechnik-Pflichtvorlesung angeboten.
Nach unseren bisherigen Erfahrungen treten beim Unterricht von MBSE-Technologien zwei Hauptprobleme auf: die Vielzahl voneinander abhängiger Begriffe und die semantischen (Meta-)Ebenen der auftretenden Daten, die sehr leicht verwechselt werden. Die Herausforderung an die Unterrichtssequenz liegt daher darin, die anfangs schwer überschaubaren Lerninhalte zu entwirren und schrittweise aufzubauen.
In Abschnitt 2 werden die angestrebten Qualifikationen detaillierter aufgeschlüsselt. Der Ablauf der Unterrichtssequenz wird in Abschnitt 3 ausführlich beschrieben, ferner die damit gemachten Erfahrungen. Abschnitt 4 diskutiert andere Ansätze, in das Themengebiet MBSE einzuführen. Wir schließen das Papier mit einem Fazit in Abschnitt 5.

2 Lernziele einer MBSE-Einführung

Im Folgenden werden die für das abstrakte Lernziel, MBSE prinzipiell verstanden zu haben und praktisch anwenden zu können, notwendigen Qualifikationen detailliert aufgeschlüsselt. Wir beginnen mit evtl. nachzuholenden Vorkenntnissen. Anschließend diskutieren wir die Aspekte Arbeitsumgebung, begriffliche Grundlagen und Werkzeug-Funktionen, welche im Rahmen einer kompakten MBSE-Einführung jeweils auf eine geeignete Auswahl reduziert werden müssen.

2.1 Nachzuholende Vorkenntnisse

Belastbare Kenntnisse in der Datenmodellierung setzen wir wie bereits erwähnt voraus. Diesbezügliche Schwächen wirken sich nach unserer Erfahrung äußerst negativ auf den Lernerfolg aus. Sofern diese Kenntnisse bei den Teilnehmern noch unsicher sind oder deren Ersterwerb länger zurückliegt, müssen diese Kenntnisse unbedingt vorab aufgefrischt und gefestigt werden. Geeignete Maßnahmen sind die Wiederholung früherer Übungen oder die Einführung von einigen einfachen Analysemustern, z.B. gem. [6]. Bei dem zweiten Modelltyp ist der Beherrschungsgrad weniger kritisch.

2.2 Auswahl der praktischen Arbeitsumgebung

Das Lernziel, MBSE praktisch anwenden zu können, erfordert natürlich, sich für ein konkretes MBSE-Framework zu entscheiden. Unsere Wahl fiel auf das Eclipse Modeling Framework (EMF) [2, 3, 13]. EMF zählt zu den bekanntesten MBSE-Frameworks, ist

frei verfügbar und dort, wo Eclipse als Entwicklungsumgebung eingesetzt wird, besonders naheliegend.
Auf die Lernprobleme im Zusammenhang mit MBSE hat die Wahl von EMF u.E. nur geringen Einfluss, d.h. die Lernprobleme treten bei anderen MBSE-Frameworks sehr ähnlich auf. Viele Lernprobleme liegen eher in der Komplexität der Konzepte, weniger in deren spezieller Implementierung in EMF. Die durch EMF selbst verursachten Lernprobleme liegen in der für Anfänger zu hohen Funktionsvielfalt und daraus resultierenden, unüberschaubaren Menüs sowie stellenweise irreführend, um nicht zu sagen falsch gewählten Bezeichnungen („Generatormodell“). Diese Probleme lassen sich indessen durch überschaubare Gegenmaßnahmen lösen.

2.3 Auswahl der zu erlernenden Begrifflichkeiten

Die zu erlernenden Prinzipien und Begriffe der MBSE können grob in zwei Bereiche unterteilt werden: (a) abstrakte, werkzeugunabhängige begriffliche Grundlagen: hierzu gehören die Unterscheidung von Transformation und Interpretation von Modellen, typische Transformationsketten, Repräsentation von Modellen und Metamodellierung, ferner Randthemen wie Produktlinien und generative SW-Entwicklung; (b) konkrete Ausprägungen der abstrakten Begriffe und ergänzende Details im benutzten MBSE-Framework, insb. die konkrete Funktion von Modelltransformatoren, die Nutzung von Ecore zur Modellrepräsentation und die Werkzeugbedienung.
Hier stellt sich vor allem die Frage, wie umfangreich abstrakte Begriffe überhaupt und welche davon vor den praktischen Unterrichtsanteilen eingeführt werden sollen. Auf keinen Fall sollte versucht werden, begriffliche Grundlagen in einem Umfang vorab einzuführen, der vergleichbar mit [8] ist. Dort wird die komplette Vielfalt möglicher Transformationsmethoden und -ketten beschrieben, ohne sich auf konkrete Technologien festzulegen, um die volle Bandbreite der Anwendungsmöglichkeiten aufzuzeigen. Anfängern sind die vielen, vage angedeuteten Technologien indes unbekannt, und die Vielfalt der Varianten ist zunächst verwirrend. Selbst mit einer Einführung, die nur eine stark reduzierte Menge an Begriffen enthielt, wurden in früheren Veranstaltungen schlechte Erfahrungen gemacht: Viele Teilnehmer hatten große Probleme, die vorab erklärten, abstrakteren Strukturen im EMF-Kontext wiederzuerkennen[1]. Anders gesagt hat die theoretische Einführung kaum Vorteile für die spätere praktische Einführung gebracht.
Im Endeffekt plädieren wir für die Anwendung einer induktiven Unterrichtsmethode, bei der abstrakte Begriffe zuerst nur im Kontext konkreter Ausprägungen (also konkreter Werkzeuge, Metamodelle usw.) eingeführt werden. Erst wenn diese konkreten Ausprägungen relativ gut verstanden sind, kann eine begrenzte Anzahl abstrakterer Begriffe erneut behandelt und verselbständigt werden, indem alternative Ausprägungen skizziert werden.

2.4 Auswahl der zu erlernenden EMF-Funktionen

Die möglichen Einsatzgebiete für EMF sind vielfältig. Neben den beiden Grundfunktionen als Quelltextgenerierungs-Framework sowie als Modellierungs- und Datenintegrations-Framework existieren eine Reihe darauf aufbauender Funktionen. Nahezu alle Operationen auf Modellen, wie bspw. die Transformation von Modellen, der Modellver-

1 Ferner wurden zugehörige Verständnisfragen in der Klausur meist nicht oder nicht korrekt beantwortet.

gleich, die Validierung von Modellen oder Abfragen auf Modellen, sind in Form konkreter Technologien auf Basis von EMF verfügbar. Einige dieser Technologien sind i.d.R. auch in den Arbeitsumgebungen der Studenten installiert. In Verbindung mit der ohnehin großen Funktionsvielfalt der Entwicklungsumgebung führt dies insbesondere bei unerfahrenen Studenten zu einer Reizüberflutung. Umso wichtiger ist es, sich anfangs klar auf die essentiellen Grundfunktionen zu konzentrieren. Hierzu zählen unserer Ansicht nach:

- die Erstellung von Ecore-Modellen mittels des grafischen Diagramm-Editors bzw. mittels des Baum-basierten Editors
- die Erstellung und Bearbeitung von Ecore-Modell-Instanzen mittels des reflektiven Editors
- die Generierung von Java-Quelltext
- die rudimentäre Anpassung des generierten Quelltextes durch Steueranweisungen (genmodel) für den Code-Generator
- die Benutzung der EMF API: obligatorisch sind hier das Laden und Speichern von EMF-Ressourcen sowie die Nutzung des generierten Quelltextes. Optional ist die Nutzung der reflektiven EMF API.

3 Struktur einer „minimalen“ MBSE-Einführung

Im Folgenden beschreiben wir den zeitlichen Aufbau sowie die vermittelten Lerninhalte der Unterrichtssequenz. Anschließend skizzieren wir Evaluationsmöglichkeiten und unsere bisherigen Erfahrungen.

3.1 Auswahl der zu erlernenden EMF-Funktionen

Abbildung 1 zeigt eine Übersicht über die Lektionen der Unterrichtssequenz, welche im weiteren Verlauf dieses Abschnitts näher erläutert werden. In der Spalte „Zeitbedarf“ gibt „P“ den Zeitbedarf für die Präsenzveranstaltung, „U“ den Zeitbedarf für selbstständig zu bearbeitende Übungen einer jeweiligen Lektion an.

	Zeitbedarf	Kurzbeschreibung	Lernziele	
			Konzeptuell	Technisch
1	P: 90 min. U: ca. 6h	Modell zu Code-Transformation	Generierungsansatz der MBSE	Grundlegende EMF-Funktionen: (1) Erstellung von Ecore-Modellen und -Instanzen (2) Generierung von Java-Quell-Text (3) Nutzung der EMF API
2	P: 180 min. U: 180 min.	Parametrisierung der Modell zu Code-Transformation	(1) Einflussnahme auf generierten Quelltext durch Steueranweisungen für den Generator (2) Einordnung der MDA-Begriffe PIM und PSM	(1) EMF Generator-Modell (genmodel) (2) Vertiefung von EMF API-Kenntnissen
3	P: 90 min. U: 180 min.	Perspektivenwechsel auf die Metamodellierungsebene	Kennenlernen des Metamodells eines konkreten Modelltyps in mehreren technologiegebundenen Varianten	-
4	P: 90 min. U: ca. 8h	Interpreteransatz der MBSE	Erkenntnis für die aus technischer Sicht identische Verarbeitung von konzeptuell auf verschiedenen Meta-Ebenen angesiedelten Daten bzw. Modellen	-
5	P: 90 min.	Reflexion & Einordnung	Einordnung der anhand von Beispielen eingeführten Konzepte in den terminologischen und konzeptuellen Rahmen der MBSE	-

Abb. 1: Lerninhalte und Ablauf: Übersicht

1. Lektion: Modell-zu-Code-Transformation im Kontext einer fachlichen Domäne

Ziele: Konzeptuell sollen die Teilnehmer in dieser Stunde anhand eines konkreten, alltäglichen Beispiels kennenlernen, wie aus einem (Daten-) Modell Quelltext generiert wird (Generierungsansatz der modellbasierten Entwicklung).
Technisch lernen die Teilnehmer die grundlegenden Funktionen von EMF kennen. Sie sollten nach dieser Stunde in der Lage sein, selbständig Ecore-Modelle und -Instanzen zu erstellen, zu editieren und daraus Java-Quelltext zu generieren. Auch der Umgang mit einer Teilmenge der EMF-API (Laden und Speichern von Modellen) sowie die Benutzung der auf Basis des Modells generierten Schnittstellen sollte nach dieser Stunde in Grundzügen beherrscht werden. Ein Einblick in die Implementierung der generierten Schnittstellen wird erst in der zweiten Unterrichtseinheit (s. unten) vorgenommen.

Methoden/Ablauf: Zu Beginn ist in einem einführenden Vortrag ein kurzer Überblick über die EMF-Modellierungstechnik Ecore sowie das Prinzip der „Modell zu Code"-Transformation in EMF zu liefern. Der Vortrag sollte dabei einen Umfang von 5–10 Folien nicht überschreiten. Die wesentlichen Konzepte können später anhand der Erstellung der konkreten Beispiel-Applikation sehr gut illustriert werden. Es empfiehlt sich dringend, eine solche Einführung im Rechnerpool vorzunehmen, so dass die benötigten Werkzeugfunktionen direkt nachvollzogen werden können. Details über die Funktionsweise des Codegenerators sind weitestgehend zu vermeiden.
Die fachliche Domäne sollte möglichst einfach sein, so z.B. das im EMF-Tutorial [14] eingeführte „Bibliothekswesen". Das Tutorial kann um die Erstellung einer einfachen

Applikation erweitert werden, für welche die Datenhaltungsschicht mittels EMF vollständig generiert werden kann. Der Gesamtbedarf dieser Einführung variiert, je nach Einführungstempo und Gruppengröße, zwischen 90 und 180 Minuten.
Vertieft wird der Stoff durch eine selbstständig zu bearbeitende Übungsaufgabe. Es empfiehlt sich die exakte Wiederholung der während der Präsenzübungen durchgeführten Funktionen anhand einer anderen, fachlichen Domäne. Je nach Umfang der auf der Datenhaltungsschicht aufsetzenden Applikation kann hier mit einem Bearbeitungsaufwand von ca. 4–8 Stunden kalkuliert werden.

2. Lektion: Parametrisierung der Modell-zu-Code-Transformation

Ziele: Auf konzeptueller Ebene lernen die Teilnehmer in dieser Lektion beispielhaft kennen, in welchem Rahmen die generierte Software durch Steueranweisungen für den Generator unterschiedlich aussehen kann. Ferner werden auch bestimmte Muster zur Umsetzung von Entwurfsdatenmodellen in eine objektorientierte Programmiersprache anhand der detaillierten Betrachtung des generierten Quelltextes nochmals wiederholt. Die Teilnehmer sollten nach dieser Unterrichtseinheit auch in der Lage sein, einen Bezug zu den MDA-Konzepten PIM (Platform Independent Model) und PSM (Platform Dependent Model) [8] herzustellen.
Auf technischer Ebene sollen die Teilnehmer den Sinn des Generator-Modells (genmodel), welches in der MDA-Terminologie die Funktion des PSM einnimmt, verstehen und einzelne Steueranweisungen für den Code-Generator kennen.

Methoden/Ablauf: Anhand der Diskussion verschiedener Varianten zur Umsetzung von Entwurfsdatenmodellen in eine objektorientierte Programmiersprache wird zunächst die Parametrisierbarkeit des Code-Generators motiviert. Diskutiert werden kann hier einerseits auf der Implementierungsebene (z.B. die Varianten zur Umsetzung von Mehrfachvererbungen oder Assoziationsbeziehungen in Java), andererseits aber auch auf der Ebene der generierten Schnittstellen (bspw. über Signaturen bzw. die Existenz von get- und set-Methoden). Zudem können globale Parameter wie bspw. das Zielverzeichnis oder spezifische Namenspräfixe für die generierten Quellen genannt werden. Der Effekt einzelner Steueranweisungen für den Code-Generator wird direkt am in Lektion 1 eingeführten Beispiel gezeigt. Die dazu notwendigen Anpassungen des Generator-Modells sollten sofort am Rechner nachvollzogen werden.

3. Lektion: Wechsel auf die Metamodellierungsebene

Ziele: Konzeptuell soll hier von der Modellierung einer alltäglichen Anwendung auf die Modellierung von Modellen eines gegebenen Modelltyps übergegangen werden. Dabei sollen Lösungen typischer Modellierungsprobleme wie die Modellierung von attributierten Kanten, mehrstelligen Beziehungen usw. erlernt werden. Ferner sollen die Teilnehmer technologiegebundene Repräsentationen der Modelle kennenlernen. In dieser Unterrichtseinheit werden keine neuen EMF-Funktionen erlernt.

Auswahl des Beispiel-Modelltyps: Klassendiagramme erscheinen als Anwendungsbeispiel naheliegend, sind aber völlig ungeeignet und sollten auf keinen Fall verwendet werden. Bei Klassendiagrammen kommt es sehr leicht zu Verwechslungen zwischen Modellierungsgegenstand und der zur Modellierung der (technischen) Repräsentation von Klassendiagrammen verwendeten Notation (in diesem Fall EMF-Ecore), welche den

Klassendiagrammen sehr ähnlich ist2[2]. Verhaltensmodelle sind hier deutlich geeigneter und führen zu weniger Missverständnissen. Wir empfehlen hier als Modelltyp eine einfache Variante von Zustandsautomaten und setzen dies i. f. Voraus.

Methoden/Ablauf: Diese Lektion besteht aus einer auf drei Stunden Bearbeitungszeit angelegten Übung, in der das Metamodell eines bestimmten Modelltyps zunächst konzeptuell zu entwerfen und anschließend mittels EMF-Ecore konkret umzusetzen ist.
Metamodelle sollten als abstraktes Konzept hier noch nicht thematisiert werden, sondern lediglich als Datenmodelle eingeführt werden, deren modellierte Anwendungsdomäne eine Modellierungssprache bzw. -technik ist. Der Begriff des Meta-Metamodells sollte hier noch komplett vermieden werden.

Die Übung sollte in folgenden drei Schritten ablaufen:

- Begonnen wird im Rahmen der Präsenzübungen mit der Konzeption des Metamodells für Zustandsautomaten in Form eines Analyseklassendiagramms. Dies kann bspw. in gemeinsamer Diskussion an der Tafel entstehen. Das (technologiefreie) Analyseklassendiagramm ist anschließend in Eigenarbeit in die beiden nachfolgend beschriebenen technologiegebundenen Metamodelle umzusetzen:
- Erstens mittels Standardmethoden in ein relationales Schema. Diese Standardmethoden sind ggf. an dieser Stelle kurz aufzufrischen.
- Zweitens in ein Ecore-Modell; das Ecore-Modell soll direkt im Werkzeug editiert werden (z.B. mit dem grafischen Ecore-Diagrammeditor).

4. Lektion: Einführung des Interpreteransatzes der modellbasierten Entwicklung

Ziele: Ziel dieser Lektion ist die Konstruktion einer generischen Applikation am Beispiel eines Interpreters für Zustandsautomaten. Konzeptuell lernen die Teilnehmer dabei den Interpreteransatz der modellbasierten Entwicklung kennen. Die aus technischer Sicht identische Verarbeitung von konzeptuell auf verschiedenen Meta-Ebenen angesiedelten Daten bzw. Modellen sollte den Teilnehmern aufgrund der Einführung von Metamodellen als spezielle Datenmodelle nicht befremdlich erscheinen. Dies ist eine zentrale Erkenntnis dieser Lektion.
Auf technischer Ebene wird in dieser Lektion kein neues Wissen vermittelt.

Methoden/Ablauf: Als Ausgangspunkt dieser Lektion fungiert das in der vorhergehenden Lektion erstellte Metamodell für Zustandsautomaten. Um das Verständnis für das Metamodell nochmals zu überprüfen und zu festigen, können während der Präsenzübungen bereits Testdaten für den später zu implementierenden Interpreter gemeinsam erstellt werden. Ausgehend von in der gewohnten, grafischen Notation skizzierten Zustandsautomaten (z.B. an der Tafel) wird deren statische Struktur extrahiert und mittels des reflektiven EMF-Editors editiert. Bei dieser Gelegenheit können die Begriffe der abstrakten bzw. konkreten Syntax einer Modellierungssprache eingeführt werden.
Abschließend kann die Grobarchitektur des Interpreters kurz diskutiert werden. Die Implementierung sollte den Teilnehmern überlassen werden, wobei ein infrastruktureller Anteil bereits vorgegeben und während der Präsenzübungen auch vorgestellt werden sollte. Ferner wird für die Bearbeitung der Übungsaufgabe eine einfache Simulationsumgebung bereitgestellt, welche zum Testen des Interpreters verwendet werden kann.

2 Dies wird insbesondere dann sehr deutlich, wenn zur Notation von Ecore-Modellen der grafische Diagramm-Editor verwendet wird.

Abb. 2 (links) zeigt die grafische Benutzeroberfläche (GUI) der Simulationsumgebung, welche nach dem Einlesen des zu simulierenden Zustandsautomaten, hier der in Abb. 2 (rechts) dargestellten Garagentorsteuerung, erscheint. Die GUI weist für jeden Trigger, der in dem Zustandsautomaten vorkommt, eine Schaltfläche auf. Ferner existieren Protokollausgaben, welche die eingetretenen Trigger, den daraufhin eingenommenen Zustand (bzw. die eingenommenen Zustände im Falle paralleler Regionen) und die auszuführenden Aktionen protokollieren. Durch manuelles Aktivieren der Trigger-Schaltflächen kann der Interpreter so schrittweise getestet werden.
Für den während der Präsenzübungen einzuführenden Anteil ist eine Dauer von 90 Minuten ausreichend. Für die Implementierung des Interpreters werden nach unseren bisherigen Erfahrungen durchschnittlich 8–10 Stunden benötigt.

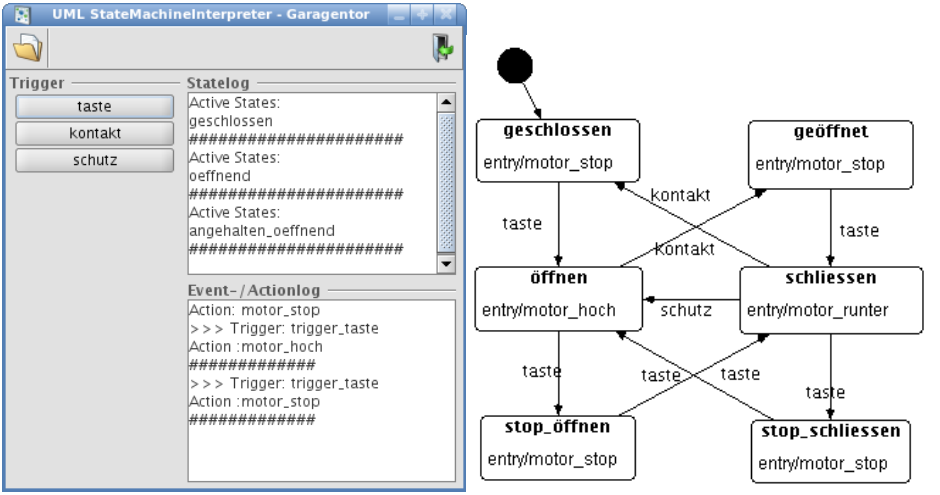

Abb. 2: GUI der Simulationsumgebung

5. Lektion: Reflexion und Einordnung

Ziele: Ziel dieser Unterrichtseinheit ist es, die während des bisherigen Verlaufs des Kurses größtenteils anhand von Beispielen eingeführten Konzepte in den terminologischen und konzeptuellen Rahmen der Modellierung und modellbasierten Entwicklung einzuordnen.

Methoden/Ablauf: Für die Reflexion des anhand von Beispielen eingeführten Stoffes empfiehlt sich ein Folienvortrag im Umfang von 45 bzw. 90 Minuten. Punktuell können dabei einige Aspekte auch erweitert werden. Insbesondere der Themenkomplex „Metamodelle“ kann hier nochmals vertieft betrachtet und in den Kontext der von der OMG spezifizierten (Meta-)Modellierungsebenen M0–M3 gesetzt werden. Optional kann hier ein kurzer Blick auf die Strukturen der UML-Spezifikationen geworfen werden. Weiterführende Themengebiete wie die Integrität von Modellen, spezialisierte Systeme für Modelltransformationen oder auch die Versionierung und Evolution von Modellen sollten kurz umrandet und in Aussicht gestellt werden.

3.2 Erste Erfahrungen

Eine auf alle Lektionen anwendbare Messtechnik zur Erfassung des Lernfortschritts ist die verbale Erfassung, entweder im Rahmen der studentischen Mitarbeit während der Präsenzphasen oder während der Abnahme bzw. Kontrolle der selbstständig zu bearbeitenden Übungsaufgaben(-Anteile). Die Archivierung der eingereichten Lösungen zu den Übungsaufgaben sowie über mehrere Semester hinweg gesammelte Klausurergebnisse bieten zudem die Möglichkeit der empirischen Validierung des Lernerfolgs.
Die bislang gesammelten, subjektiven Eindrücke sowie die qualitativen Aussagen der Teilnehmer waren durchweg positiv. Schnelle Erfolgserlebnisse bei der Arbeit mit EMF wirkten sich auf die Motivation der Teilnehmer positiv aus und abstrakte Konzepte oder Begrifflichkeiten wurden im Kern schnell verstanden. Die bewusst intensiv geführten Gespräche mit den Teilnehmern unterstreichen zudem die These, dass das bisherige Vorgehen, eine umfangreiche, theoretische Einführung vor der praktischen Umsetzung, die Teilnehmer meist überforderte.

4 Andere Ansätze einer MBSE-Einführung

Das Thema MBSE gehört zweifellos zu den fortgeschrittenen Themen der Informatik bzw. Softwaretechnik. In grundständigen, breit angelegten Softwaretechnik-Lehrbüchern wird das Thema daher zwar erwähnt, aber nur auf wenigen Seiten einführend abgehandelt. Vermittelt werden nur abstrakte Begriffe und Beschreibungen von MBSE-Infrastrukturen, nicht deren praktische Anwendung. Ähnliches gilt für breit angelegte Bücher zum Thema Architektur, z.B. [10], in denen MBSE bzw. MDA nur als einer von sehr vielen Architekturstilen abgehandelt wird, und für Bücher über die UML, die sich vor allem mit der Vielfalt an Diagrammtypen beschäftigen. Konkrete Technologien werden nicht vorgestellt, so dass Leser nicht in die Lage versetzt werden, die relativ abstrakt vorgestellten Methoden praktisch nachzuvollziehen und zu üben. Die dadurch ausbleibenden Erfolgserlebnisse führen unseren Erfahrungen nach schnell zu Frustration und gemindertem Interesse an der Thematik.
Beispiele für dedizierte Lehrbücher zum Thema MBSE, in denen praktische Kompetenzen angestrebt werden, sind [12] und [4].
Adressierte Zielgruppen von [12] sind laut eigener Aussage des Buches Architekten, Entwickler und Projektleiter. Primär verfolgtes Ziel ist es „den Leser davon zu überzeugen, dass MDSD[3] ein praktikabler Ansatz ist und in der Tat in vielen Fällen der traditionellen Entwicklung überlegen ist“. Zwar wird versucht, den Leser anhand eines ausführlichen Beispiels in die Thematik einzuführen, die Einstiegshürde ist für Einführungsveranstaltungen zur Thematik MBSE allerdings zu hoch. Es werden Konzepte, Technologien und Frameworks wie bspw. Metaprogrammierung, Template-basierte Codegeneratoren oder die Java Plattform „Enterprise Edition“ (JEE) vorausgesetzt, welche von einem Studenten im 3. oder 4. Fachsemester i. d. R nicht beherrscht werden. Konkrete Technologien werden nicht vorgestellt, so dass man nicht in die Lage versetzt wird, die relativ abstrakt vorgestellten Methoden praktisch nachzuvollziehen und zu üben.
In [4] wird versucht, einen stärkeren Bezug zu konkreten Technologien basierend auf Eclipse und EMF herzustellen. Im Gegensatz zum Aufbau der von uns vorgeschlagenen Unterrichtssequenz wird hier jedoch ein deduktiver Ansatz zur Einführung in die Thema-

3 Anm.: Die Autoren verwenden den Begriff MDSD (Model Driven Software Development) synonym zu dem von uns hier verwendeten Begriff MBSE (Modellbasierte Software-Entwicklung)

tik gewählt. Die Konzeption des Buchaufbaus folgt einer Dreiteilung: Teil I beschreibt die theoretischen Grundlagen des Ansatzes, Teil II beleuchtet die Auswirkungen der vorgestellten Konzepte auf die Prozesse und die Organisation von Unternehmen. Erst Teil III bringt die Ideen zur Anwendung und setzt sie mit konkreten Technologien zur Umsetzung in Beziehung. Studenten im 3. oder 4. Fachsemester sind nach der Vermittlung von Teil I des Buches nicht in der Lage, die essentiellen Konzepte der MBSE herauszufaktorisieren. Teil II des Buches ist allenfalls in einem Studiengang Wirtschaftsinformatik nutzbar. Die vorgeschlagene a-posteriori-Illustration der theoretischen Konzepte anhand konkreter Technologien (Teil III) ist unserer Erfahrung nach meist erfolglos. Abschließend ist ferner der Anspruch des Buches zu hinterfragen, sich gleichermaßen an „Manager, Berater und Projektleiter“ wie an „Entwickler und Architekten“ als Zielgruppe zu richten.
Einen anderen Ansatz, in die Thematik einzuführen, stellen technologiegebundene Tutorials wie bspw. [14] oder [9] dar. Anfängern wird hier auf wenigen Seiten ein schneller Einstieg in komplexe Technologien ermöglicht, so dass sich schnell erste Erfolgserlebnisse einstellen. So kann das Interesse geweckt werden und es wird eher das Angebot weiterführender Tutorials oder Literatur wahrgenommen, um Kenntnisse zu vertiefen. Nachteilig ist hier jedoch zu vermerken, dass durch die Fokussierung auf konkrete Technologien eine saubere Begriffsbildung oftmals vernachlässigt bzw. Begrifflichkeiten oder Konzepte falsch oder inkonsistent eingeführt werden. Als Beispiel ist hier bereits der Titel von [9] aufzuführen, welcher mit „Create UML models and generate code“ schlichtweg falsch gewählt ist. Auch wenn die grafische Notation für Ecore-Modelle an jene für UML-Klassendiagramme erinnert, wird hier nicht aus UML-Modellen, sondern aus Ecore-Modellen Java-Quelltext generiert. Eine Einführung mittels Tutorials sollte daher nicht ohne deren Reflexion im Rahmen von Präsenzveranstaltungen stattfinden.

5 Fazit

Die hier vorgestellte Unterrichtssequenz zeigt, dass man auch mit wenig Aufwand den Studierenden zentrale Konzepte der MBSE verbunden mit praktischen Erfahrungen vermitteln kann. Insbesondere legen unsere bisher gesammelten Lehrerfahrungen auf dem Gebiet der modellbasierten Entwicklung nahe, hier eine induktive Unterrichtsmethode anzuwenden, bei der abstraktere Begriffe zuerst nur im Kontext konkreter Ausprägungen (also konkreter Werkzeuge, Metamodelle usw.) eingeführt werden. Erst wenn diese konkreten Ausprägungen relativ gut verstanden sind, kann eine begrenzte Anzahl abstrakterer Begriffe erneut behandelt und verselbständigt werden, indem alternative Ausprägungen skizziert werden.
Insgesamt lässt sich so das komplexe Thema MBSE auch schon im dritten oder vierten Fachsemester vermitteln. So kann diese zunehmend an Bedeutung gewinnende Thematik auch in Informatik-Bachelorstudiengänge integriert werden.
In Zukunft werden wir die hier beschriebene Unterrichtssequenz im Rahmen unserer grundständigen Softwaretechnik-Vorlesung, welche für die meisten Studiengänge eine Pflichtveranstaltung im 2. Studienjahr darstellt, weiter einsetzen. Durch eine gezielte und längerfristig angelegte Evaluation werden wir Aufbau und Inhalt der MBSE-Einführung weiter verfeinern mit dem Ziel, optimalen Lernerfolg bei möglichst minimalem Arbeitsaufwand der Studierenden zu erreichen.

Literatur

[1] Eclipse UML2: Model Development Tools, UML2; www.eclipse.org/uml2/

[2] EMF: Eclipse Modeling Framework; http://www.eclipse.org/emf

[3] Gronback, Richard C.: Eclipse Modeling Project - A Domain-Specific Language (DSL) Toolkit; Addison-Wesley Longman; 2009

[4] Gruhn, Volker; Pieper Daniel; Röttgers Carsten: MDA - Effektives Softwareengineering mit UML2 und Eclipse; Springer; 2006

[5] IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK); IEEE; 2004; http://www.computer.org/portal/web/swebok/html/contents

[6] Kelter, Udo: Lehrmodul Analysemuster; Fachgruppe Praktische Informatik, Universität Siegen; 2009

[7] Ludewig, Jochen: Models in Software Engineering - An Introduction; Softw. Syst. Model., Springer, 2:1, p.5-14; 2003

[8] MDA Guide Version 1.0.1; OMG, Doc. omg/2003-06-01; 2003

[9] Powell, Adrian: Model with the Eclipse Modeling Framework, Part 1: Create UML models and generate code; 2004; https://www.ibm.com/developerworks/opensource/library/os-ecemf1/

[10] Reussner, Ralf; Hasselbring, Wilhelm: Handbuch der Software-Architektur; dpunkt-Verlag, 555p.; 2009; http://www.handbuch-softwarearchitektur.de

[11] Seidewitz, Ed: What Models Mean; IEEE Software 20:5, Sep./Oct. 2003, p.26-32; 2003

[12] Stahl, Thomas; Völter, Markus; Efftinge, Sven; Haase, Arno Modellgetriebene Softwareentwicklung - Techniken, Engineering, Management; dPunkt; 2007

[13] Steinberg, D.; Budinsky, F.; Patenostro, M.; Merks, E.: EMF: Eclipse Modeling Framework, 2nd Edition. Addison Wesley; 2008

[14] Tutorial aus der EMF-Hilfe: Generating an EMF Model; 2007; http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.emf.doc/tutorials/clibmod/clibmod.html

[15] UML: Unified Modeling Language; http://www.uml.org