MDA und KI: Domänenspezifische Modellierung und Umsetzung wissensintensiver Prozesse

Steffen Kruse, Malte Zilinski, Hilke Garbe, Claus Möbus

Learning Environments and Knowledge-based Systems,
Carl von Ossietzky Universität Oldenburg,
Ammerländer Heerstr. 114-118, 26129 Oldenburg
s.kruse, m.zilinski, h.garbe, claus.moebus@uni-oldenburg.de

Abstract: [Erschienen in: Software Engineering 2008, Workshopband, ISBN 978-3-88579-216-1]
In dieser Veröffentlichung wird ein Verfahren vorgeschlagen, domänenspezifische, wissensintensive Prozesse zu erfassen und diese für einen Softwareentwicklungsprozess nutzbar zu machen. Dabei wird mit Hilfe von kognitiven Mustern das von Domänenexperten zur Verfügung gestellte Wissen strukturiert und in Form von spezialisierten Modellen aufbereitet. Es wird gezeigt, dass sich diese als *Computational Independent Models* (CIM) für die Softwareentwicklung nach *Model Driven Architecture* (MDA) nutzen lassen. Als Zielplattform für den MDA-Prozess werden von uns Methoden bzw. Sprachen aus dem Bereich der Künstlichen Intelligenz (KI) angestrebt, da sie für die Umsetzung wissensintensiver Prozesse besonders geeignet sind. Es wird beispielhaft gezeigt, wie dieses Verfahren auf die Studienplanung angewendet und auf die Planungsdomäne mit ausführbaren Planungssprachen abgebildet wird. Es wird gezeigt, wie die Integration bewährter Ansätze aus dem Bereich des Software Engineering (MDA) und der Künstlichen Intelligenz (in dem hier vorgestellten Beispiel Planungsprobleme) mithilfe von geeigneten Metamodellen für die Lösung komplexer und hoch-spezialisierter Probleme möglich ist. Dabei ist der entwickelte Ansatz innerhalb der Problemklasse der Planungsprobleme generisch und kann für verwandte Probleme in anderen Anwendungsdomänen wiederverwendet werden. Das vorgeschlagene Verfahren ermöglicht es den beteiligten Rollen (Domänenexperten, Software Engineers und KI-Experten) für sie spezialisierte Modelle zu verwenden.

1 Einleitung

Ansätze der modellgetriebenen Softwareentwicklung (*Model Driven Development*, MDD), insbesondere der modellgetriebenen Architektur (*Model Driven Architecture*, MDA) etablieren sich zunehmend als Standard im Bereich des Softwareengineering [GPR06]. Gerade die MDA bietet ein Vorgehen, das für die Entwicklung von Softwaresystemen benötigte Wissen ganzheitlich in Modellen zu fassen und über deren Ausdrucksstärke für unterschiedliche Domänen und Experten besser zugänglich zu machen. Gestützt wird die MDA von Methoden der Modelltransformation bzw. generativen Ansätzen, die es ermöglichen, die Erstellung von plattform- bzw. domänenspezifischen Artefakten weitestgehend zu automatisieren und die so einen hohen Grad an Wiederver-

wendung und Robustheit von entwickelten Lösungen bieten [MM03].
Die Generierung von Code aus Modellen lässt sich sehr einfach für statische Artefakte eines Softwaresystems durchführen, zum Beispiel für Datenbankenstrukturen, Objektmodellen oder Schnittstellen. Dynamische Aspekte sind ungleich schwerer vollständig zu modellieren und werden häufig durch das manuelle Auffüllen von generierten Stubs eingefügt oder in sehr speziellen Templates für die Codegeneratoren verborgen. Im Umfeld von betrieblichen Softwaresystemen wird häufig ein anderer Ansatz gewählt: die dynamischen Eigenschaften werden als Geschäftsprozesse modelliert und zur Laufzeit von einer Geschäftsprozessengine interpretiert. Ein solches Vorgehen tritt häufig in Verbindung mit serviceorientierte Architektur (SOA)[MLM+06] auf. Im Sinne der MDA stellt die Geschäftsprozessengine eine Plattform dar, die es ermöglicht, das plattformspezifische Modell (PSM) oder sogar das plattformunabhängige Modell (PIM) als berechnungsvollständiges Modell ohne weitere Anpassungen auszuführen, da die plattformspezifischen Details in der Geschäftsprozessengine verborgen bleiben.
In vielen Arbeiten zu MDA liegt das Augenmerk auf den letzteren Schritten des MDA-Prozesses, also der Generierung von Code aus den PSMs und der Modelltransformation von PIM (und Plattformmodell) zu PSM. Eine formale Beschreibung des berechnungsunabhängigen Modells (CIM) und ein systematisches Vorgehen zur Erstellung (bzw. Transformation) des PIM aus dem CIM werden ausgelassen.
Das hier vorgestellte MINT-Vorgehensmodell stellt einen Ansatz vor, das Wissen von Domänenexperten zu erfassen und als CIMs für die MDA verfügbar zu machen. Dafür wird eine Planungsmethode aus dem Bereich der KI adaptiert. Darüber hinaus wird gezeigt, wie komplexe Probleme für den Entwicklungsprozess modelliert und spezialisierte Softwarekomponenten (in diesem Fall „off-the-shelf"-Planer) analog zu Geschäftsprozessengines als Plattformen für die Lösung verfügbar gemacht werden können. Für Software Engineers erleichtert dieses Vorgehen die Integration spezieller Sprachen, ohne in der Regel deren Details kennen zu müssen.

2 MINT Vorgehensmodell

Die Betrachtung wissensintensiver Prozesse im Kontext der MDA ist interessant, da sie in vielen Anwendungen auftreten, ihre Lösung aber nicht trivial ist. Im Bereich der KI existieren spezialisierte Sprachen, die sich für die Beschreibung und Lösung solcher Probleme bewährt haben, deren Anwendung bisher aber Expertenwissen erforderte. Das im Folgenden beschriebene MINT-Vorgehensmodell ermöglicht es Domänenexperten, Planungsprobleme fachlich zu beschreiben und diese für die Softwareentwicklung von spezialisierten planungsspezifischen Details zu abstrahieren.
Dieses MINT-Vorgehensmodell wird auf die Domäne Studienplanung angewandt. Als Ziel dient ein Softwaresystem, welches Studierenden entsprechend ihrer Studiensituation und ihren individuellen Anforderungen bei der Studienplanung unterstützt. Abbildung 1 zeigt eine Übersicht des intendierten Zielsystems und Abbildung 2 den gewünschten

Geschäftsprozess als MINT XL[1]-Diagramm.

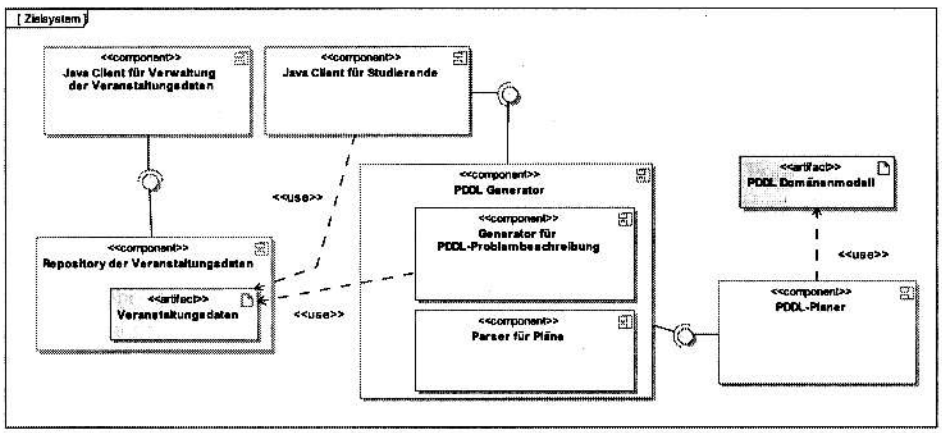

Abbildung 1: Intendiertes Zielsystem

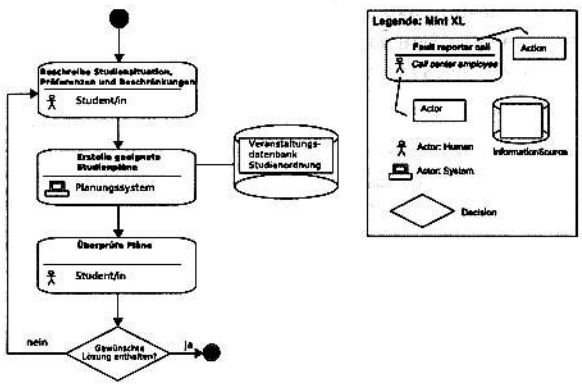

Abbildung 2: Geschäftsprozess der Studienplanung

Zur Beschreibung der planungsspezifischen Prozesse auf CIM-Ebene werden Modellmuster für wissensintensive Prozesse (siehe [SAA+02, GRC98]) verwendet und angepasst. Zur Definition relevanter fachlicher Details werden die Musterelemente mit klassischen MDA-Methoden verfeinert. Die Studienplanung lässt sich auf ein Scheduling-Problem abbilden. Auf PIM-Ebene wird deshalb ein generisches Scheduling-Metamodell bereitgestellt, welches die fachlichen Anforderungen des CIM fasst und um Scheduling-Details erweitert. Das Studienplanungsproblem wird als partielle Instanz des Scheduling-Metamodells beschrieben. Das Problem wird auf plattformspezifischer Ebene als Planning Domain Definition Language (PDDL)-Planungsproblem (siehe [McD98, FL03]) oder als

[1]MINT XL ist die im MINT-Projekt entwickelte domänenspezifische Sprache zur Modellierung von Geschäftsprozessen im MDA-Kontext (siehe `http://www.mint-projekt.de`)

Constraint Satisfaction Problem (CSP) repräsentiert [RN04]. Somit ist das Modell deklarativ und maschinell lösbar.

2.1 CIM

CommonKADS ist eine Methodologie zur Erfassung, Strukturierung, Formalisierung und Operationalisierung von Wissen. Für die Modellierung wissensintensiver Prozesse stellt sie eine deklarative, für Domänenexperten verständliche Sprache bereit. Es werden vorgefertigte, generische Problemlösemuster, die „Template Knowledge Models" (TKMs), für verschiedene wissensintensive Prozesse angeboten [SAA+02], welche von Experten für die Domänenmodellierung verwendet werden sollen. Da es sich bei den TKMs um generische Problemlösungen handelt, beschreiben sie den gewünschten Prozess in der Regel nicht genau den Anforderungen der Domäne entsprechend und es muss eine Adaption der Muster erfolgen. Das adaptierte instantiierte TKM *Planning*[2] (siehe Abbildung 3) eignet sich als Verfeinerung des in Abbildung 2 dargestellten Geschäftsprozesses, da dieser einen wissensintensiven Planungsprozess impliziert. CommonKADS definiert für alle Elemente des Musters Rollen, an denen erkennbar ist, welche Funktion die Daten bzw. Aktivitäten im Prozess übernehmen. Bei Daten wird zwischen dynamischen und statischen Rollen unterschieden: erstere bezeichnen Daten mit geringer und zweitere mit längerer Lebensdauer. Die Studienordnung hat in der Beispieldomäne mit einer Lebensdauer von mehreren Jahren die längste Gültigkeit. Für die Definition von Planungsproblemen wird im MINT-

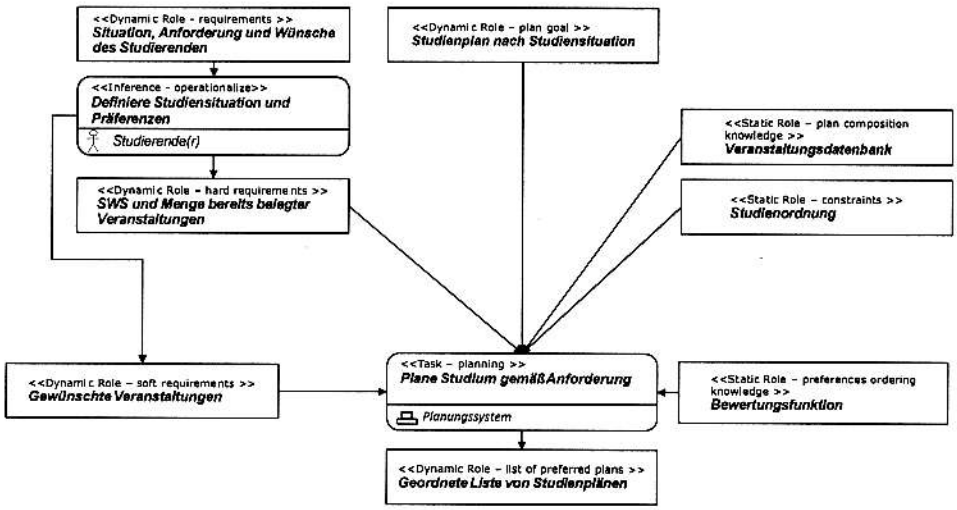

Abbildung 3: Planungsmuster

Vorgehensmodell die Domänenmodellierung über die Strukturierung der Domänenobjekte hinaus auf die Modellierung der statischen Struktur des Planungsproblems erweitert. Für

[2]Das originale TKM *Planning* findet sich in der CommonKADS Methodologie[SAA+02].

die Domäne Studienplanung wird dafür das Studienordnungsmetamodell zur Verfügung gestellt.
Neben dem gezeigten Geschäftsprozess, dem Planungsmuster und dem Modell der Studienordnung, werden weitere Modelle erstellt, die die Struktur der Daten, sowie aller anderen Anforderungen an das restliche Softwaresystem enthalten. Diese werden gemäß des MDA-Ansatzes entwickelt, aber hier nicht weiter beschrieben.

2.2 PIM

Das gewählte Problem der Studienplanung lässt sich auf ein nicht-präemptives Scheduling-Problem mit harten und weichen Beschränkungen abbilden. Dabei wird das zeitliche Pensum des Studierenden als Kapazität einer Maschine im Sinne des Scheduling aufgefasst. Die in der Studienordnung definierten Module entsprechen Aktivitäten; mit Eigenschaften wie Dauer und Ressourcenverbrauch.
Für die plattformunabhängige Beschreibung von Scheduling-Problemen wurde ein generisches Metamodell entwickelt (siehe Abbildung 4), das sich an einer Scheduling-Ontologie orientiert [Raj04]. Dieses erlaubt die Abbildung der Domänenobjekte auf Jobs, Aktivitäten, Vorbedingungen und anderen Scheduling-spezifischen Eigenschaften, wie Fälligkeiten, Kapazitäten usw..
Mit diesem Scheduling-Metamodell wird gemäß der auf der CIM-Ebene definierten Anforderungen ein Modell erstellt, welches die berechnungsabhängigen, planungsrelevanten Aspekte der Domäne beschreibt. Dies wird über die Annotation des Studienordnungsmodells der CIM-Ebene anhand eines Scheduling-Profils erfolgen und der anschließenden automatisierten Transformation mittels noch dafür bereitzustellender Templates. Das Scheduling-Modell ist an dieser Stelle nur partiell instantiiert, da die dynamischen Anteile (Ressourcen) sich aus der Anfrage des Studierenden ergeben und erst zur Laufzeit bekannt sind. Zur Lösung des Scheduling-Problems müssen diese dem Planer zu Verfügung gestellt werden, dies erfolgt über den Einsatz eines Generators zur Laufzeit. Dieses Scheduling-Modell fügt sich zu anderen Modellen, die nach gewöhnlichen MDA Methoden entwickelt werden und zusammen alle Komponenten des Softwaresystems abbilden.

2.3 PSM

Als Plattform zur Lösung des Scheduling-Problems wird der PDDL-Planer *sgplan*[3] gewählt. Dieser wird zur Laufzeit in das Softwaresystem eingebunden. Als Eingabe benötigt der Planer die Studienordnungen, sowie die konkrete Anfrage des Studierenden (Studiensituation, gewünschte Veranstaltungen usw.). Das auf PIM-Ebene definierte Scheduling-Modell (Studienordnung) wird zu einer PDDL-Domänenbeschreibung trans-

[3] `http://manip.crhc.uiuc.edu/programs/SGPlan/index.html`

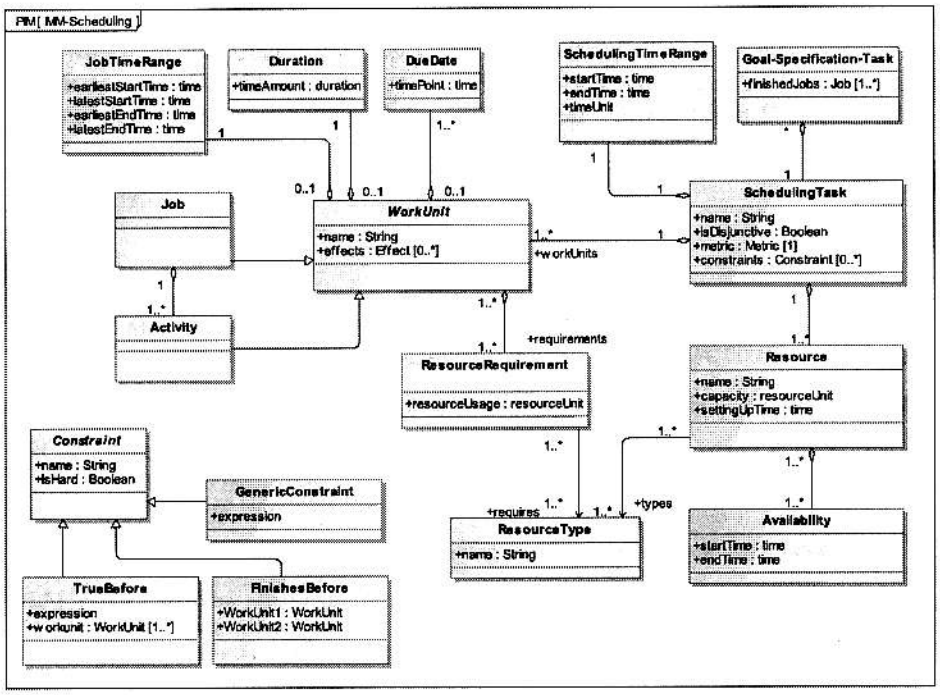

Abbildung 4: Scheduling Metamodell

formiert.
Die dynamischen Anforderungen der Studierenden werden von einem Generator zur Laufzeit in eine PDDL-Problembeschreibung übersetzt und dem Planer zur Lösung übergeben. Die Ergebnisse werden wiederum übersetzt und dem umgebenen Softwaresystem zur weiteren Verarbeitung und schließlich dem Client zurückgegeben.
Als alternative Planungsplattform für Scheduling-Problem ermöglichen CSPs eine deklarative und maschinell lösbare Beschreibung. Entsprechende Modelle können mit ähnlichen Methoden aus dem generischen Scheduling-PIM transformiert werden. Eine Übersicht des resultierenden Zielsystems ist in Abbildung 1 dargestellt.

3 Zusammenfassung und Ausblick

Es wurde ein Verfahren vorgeschlagen mit dem wissensintensive Prozesse der Planungsdomäne modelliert und mittels MDA in ausführbaren PDDL-Code transformiert werden können. Es wurde ein Metamodell zur Definition von Studienordnungen auf CIM-Ebene entwickelt. Das erstellte, generische Metamodell auf PIM-Ebene erlaubt die Abbildung von Problemen unterschiedlicher Scheduling-Klassen. Bisher befindet sich eine Transformation zu einer PSM-Repräsentation in PDDL für nicht-präemptives Scheduling in der

Konzeptionsphase. Eine Erweiterung auf andere Scheduling-Problemklassen ist geplant; dabei wird auch eine Überprüfung der Übertragbarkeit auf allgemeine Planungsprobleme langfristig angestrebt. Desweiteren soll das MINT-Vorgehensmodell an Planungsproblemen aus dem Energieversorgungssektor und damit an einer anderen Domäne evaluiert werden.

Literatur

[FL03] Maria Fox und Derek Long. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res. (JAIR)*, 20:61–124, 2003.

[GPR06] Volker Gruhn, Daniel Pieper und Carsten Röttgers. *MDA*. Springer, Heidelberg, 2006.

[GRC98] Karen Gardner, Alexander R. Rush und Michael Crist. *Cognitive Patterns: Problem-Solving Frameworks for Object Technology (Managing Object Technology Series)*. Cambridge University Press, 1998.

[McD98] D. McDermott. PDDL — the planning domain definition language, 1998.

[MLM+06] C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F. Brown, Rebekah Metz und Booz Allen Hamilton. *Reference Model for Service Oriented Architecture 1.0 Committee Specification 1*. OASIS, August 2006. Last seen: 25/09/2006.

[MM03] Joaquin Miller und Jishnu Mukerji. *MDA Guide Version 1.0.1*. Object Management Group (OMG), June 2003.

[Raj04] Dnyanesh Rajpathak. *A Generic Library of Problem-Solving Methods for Scheduling Application*. Dissertation, The Open University, 2004.

[RN04] Stuart Russell und Peter Norvig. *Künstliche Intelligenz. Ein moderner Ansatz*, Jgg. 2. Pearson Studium, 2004.

[SAA+02] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde und B. Wielinga. *Knowledge Engineering and Management*. The MIT Press, 2002.