Integrationsaspekte der Simulation:
Technik, Organisation und Personal
Gert Zülch & Patricia Stock (Hrsg.)
Karlsruhe, KIT Scientific Publishing 2010

Modellierung mit SysML zur Abbildung von Produktionsprozessen

Modelling of Production Systems with SysML

Oliver Schönherr, Oliver Rose
Technische Universität Dresden, Dresden (Germany)
oliver.schoenherr@tu-dresden.de, oliver.rose@tu-dresden.de

Abstract: In this paper, we present an approach for developing a simulation-tool-independent description for discrete processes and for converting such a general model into simulation-tool-specific models. Our aim is to develop models by means of SysML and to build converters from SysML models to models of a large variety of simulation tools. Based on this architecture, we develop a general model description for discrete processes which permits to create comprehensive scenarios. Our main domain is production systems but we show which elements are not domain specific and can be generalized to an approach for a standard to model discrete production planning and control problems.

1 Einleitung

In vielen Bereichen der Wissenschaft, wie der Informatik, der Elektrotechnik oder im Bauingenieurwesen haben sich Beschreibungssprachen wie die UML durchgesetzt (vgl. WEILKIENS 2006). Im Bereich der Modellierung diskreter Prozesse ist dem nicht so (WEILKIENS 2006, 261). Doch gibt es hier aus den zwei folgenden Gründen einen deutlichen Bedarf.

1. Durch Beschreibungssprachen können Projekte nach den Prinzipien des Systems Engineering wie Design-Synthese und System-Überprüfung umgesetzt werden, um auch bei großen Projekten den Überblick zu wahren und die Unsicherheit der Modelle (Diskrepanz zwischen Modell und Realität) zu reduzieren.
2. Beschreibungssprachen sind ein zentrales Element der automatischen Modellgenerierung.

In der Softwaretechnik ist die automatische Codegenerierung von UML-Modellen mittels sogenannter *CASE-Tools* weit verbreitet und standardisiert (FOWLER 2003, S. 23). In der Modellierung diskreter Prozesse hingegen gibt es zwar viele Ansätze, die unter dem Thema "*Model Based Software Engineering*" (MBSE) betrachtet werden (z.B. *Stateflow Coder*, ASCET, das Projekt ADAGE), jedoch setzte sich

bisher keiner ausreichend durch (vgl. Fachausschuss Software Engineering 2004). Dies kann durch das Fehlen einer einheitlichen, ausreichend mächtigen oder nicht-proprietären Beschreibungssprache bedingt sein. Doch gerade bei der Modellierung diskreter Prozesse im Bereich der Produktion ist die automatische Codegenerierung sinnvoll, da in diesen eine Fülle an verschiedenen Werkzeugen eingesetzt wird, die mangels Standardisierung oft nicht in der Lage sind, ihre Modelle untereinander auszutauschen. Durch die automatische Codegenerierung aus einem Standardmodell können Modelle für verschiedene Werkzeuge automatisch generiert werden.

Die *Object Management Group* (OMG) nahm sich des Problems an und veröffentlichte im April 2006 die *Systems Modeling Language* (SysML) 1.0, einen Standard, der auf der *Unified Modeling Language* (UML) 2 aufbaut und sich im Bereich des *Systems Engineering* als standardisierte Beschreibungssprache etablieren soll. In der kurzen Zeit seit der Veröffentlichung von SysML, wurde die neue Beschreibungssprache kontrovers diskutiert (vgl. GfSE 2008). Die schnell steigende Nachfrage erkennt man daran, dass alle Modellierungswerkzeuge relevanter Herstellern wie ARTiSAN, Telelogic, I-Logix und Sparx Systems mittlerweile SysML unterstützen.

Diese Arbeit versucht, einen allgemeinen Ansatz für die Modellierung von diskreten Systemen bereitzustellen und eine Lösung zur automatischen Modellgenerierung von diskreten Prozessen zu geben. Der Ansatz sieht vor, Modelle mit SysML zu entwerfen, um sie anschließende für eine möglichst breite Menge an Simulationstools aufbereiten zu können. Ursprünglich wurde die Lösung von den Autoren für die Domäne der Produktion entworfen und ist daher für diese besonders geeignet. Um die Eigenheiten der Modellierung diskreter Modelle zu verstehen, wurde neben einer umfangreichen Literaturrecherche und einer Expertenbefragung eine weitreichende Marktanalyse von Produktions- und Simulationstools durchgeführt, welche in früheren Arbeiten eingesehen werden können (vgl. SCHÖNHERR, ROSE 2009). Die gewonnenen Erkenntnisse bildeten die Grundlage für die Erstellung eines allgemeinen Modells für die Modellierung von diskreten Prozessen, mit dem möglichst umfassend verschiedenste Szenarien abgebildet werden können.

Im ersten Teil des Papers wird die automatische Modellgenerierung, vom Entwurf eines Szenarios mit SysML bis zur Übersetzung eines Simulationswerkzeugs, beschrieben. Im zweiten Teil der Arbeit wird eine Domänenübergreifender Modellierungsansatz für diskrete Modelle beschrieben. Abschließend wird die Arbeit zusammengefasst und ein Ausblick

2 Ein Ansatz zur Automatischen Modellgenerierung

In unserer Arbeitsgruppe wurde ein Programm entwickelt, das aus SysML-Modellen automatisch Modelle für Simulationswerkzeuge generiert. Um das Programm möglichst effektiv zu gestalten, wurde eine mehrschichtige Architektur gewählt (vgl. Abb. 7). Zuerst muss das SysML-Modell mit Hilfe eines Modellierungstools entworfen werden, das das Modell in einem zur Weiterverarbeitung geeigneten Austauschformat bereitstellt.

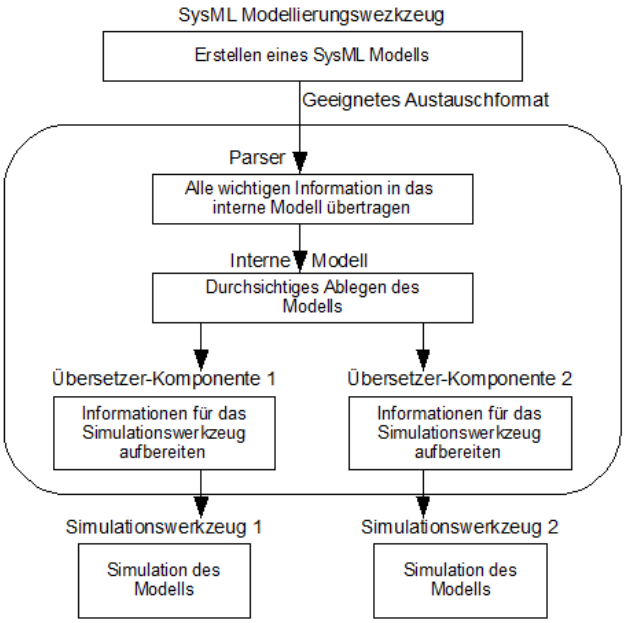

***Abbildung 1**: Systemarchitektur*

Wenn das SysML-Modell in einem geeigneten Austauschformat bereit steht, kann es in das Äquivalent eines Simulationswerkzeugs transformiert werden. Da es möglich sein soll, ein in SysML vorliegendes Modell in verschiedene Simulationsprogramme transformieren zu können, muss für jedes Programm eine eigene Ausgabe erzeugt werden. So erhält jedes Programm ein für sich geeignetes Modell in speziellem Format. Um den Arbeitsaufwand dafür möglichst gering zu halten, wurde die Erzeugung des Modells in zwei Schritte geteilt (vgl. Abb. 1). Im ersten Schritt kommt ein Programm, das als Parser bezeichnet wird, zum Einsatz. Dieses liest das im Austauschformat gespeicherte SysML-Modell ein und filtert alle wichtigen Informationen, die es anschließend in ein "internes Modell" überführt. Im zweiten Schritt wird ein "Übersetzer-Komponente" eingesetzt, das das "interne Modell" für ein Simulationsprogramm übersetzt. Dafür filtert das "Translator-Plugin" alle für das Simulationsprogramm wichtigen Daten aus dem "internen Modell" und bringt sie in das spezielle Format des Simulators. Da jeder Simulator ein eigenes Format hat, muss jeweils ein eigenes Translator-Plugin geschrieben werden.

Für die Erstellung der Modelle wird das SysML ModellierungswerkzeugMagicDraw verwendet. Aktuell arbeiten wir an einem freien Modellierungswerkzeug, das speziell auf die Bedürfnisse von Systemingenieuren angepasst wird. Die Entwicklung des internen Modells und des Parsers sind für das Verhaltens- und Strukturmodell abgeschlossen. Für die Simulationswerkzeuge Anylogic, Flexsim und Simcron Modeler sind bereits Übersetzer-Komponente entwickelt.

3 Ein allgemeiner Modellierungsansatz

Auf der Basis konzeptioneller Forschungsarbeiten im Bereich der Modellierung von Produktionssystemen (vgl. SCHÖNHERR, ROSE 2009), wurde ein allgemeiner Modellierungsansatz zur Abbildung diskreter Systeme entworfen. Diesem soll sich ein Großteil der diskreten Optimierungsprobleme unterordnen lassen. Besonders geeignet ist der Ansatz für die Domäne der Produktion und für die Beschreibungssprache SysML. Wie UML 2 unterteilt auch SysML das Gesamtmodell in Struktur- und Verhaltensteile. Während im strukturellen Modell die statische Struktur eines Systems bestehend aus Objekten und ihren Beziehungen beschrieben wird, beschreibt das Verhaltensmodell das dynamische Verhalten eines Systems über der Zeit. Systeme die von außen gesteuert werden, benötigen zusätzlich ein Kontrollmodell. Das Kontrollmodell beschreibt die Produktionsplanungs- und Produktionssteuerungskomponenten innerhalb eines Systems.

3.1 Das strukturelle Modell

Das strukturelle Modell beschreibt welche Objekte sich in einem System befinden und welche Beziehungen zwischen diesen Objekten bestehen. Es können drei Klassen, imaginäre Objekte, reale Objekte und Hilfsobjekte, unterschieden werden (vgl. Abb. 2). Die imaginären Objekte "Ankunft- und Abgangsprozess", "Warteschlange" und "Prozess" sind Teil vieler Anschauungen, wie der Warteschlangentheorie. Das Fließobjekt betritt ein System durch den Ankunftsprozess, verlässt es durch den Abgangsprozess und wird in Warteschlangen gelagert. Prozesse beanspruchen reale Objekte für eine festgelegte Zeit und können eine Zustandsänderung zufolge haben. Sie können zur Ausführung Ressourcen benötigen.

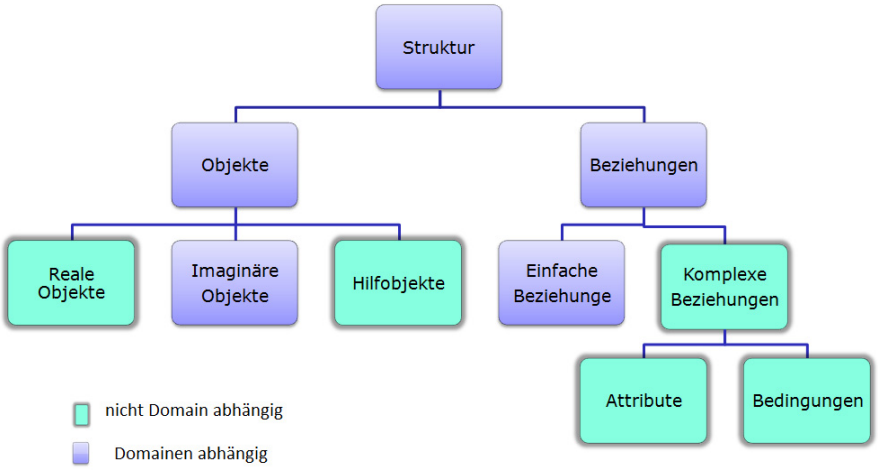

***Abbildung 2**: Übersicht Modellierung der Struktur*

Während diese imaginären Objekte in jedem diskreten System zu finden sind, wird die Domäne des Modells durch die realen Objekte und Hilfsobjekte bestimmt. Reale Objekte sind die Fließobjekte, welche das System durchlaufen (z.B. Werkstücke, Partien usw.) und die in ihren Ausprägungen sehr vielfältigen Ressourcen (Arbeiter, Maschine, Raum usw.). Hilfsobjekte werden nicht unbedingt benötigt, vereinfachen

aber die Modellierung. Beziehungen zwischen den Objekten können einfache Beanspruchungen sein, wie die Reservierung eines realen Objektes durch einen Prozess. Doch können Beziehungen durchaus auch komplexer und mit Attributen oder Bedingungen belegt sein. Ein Beispiel hierfür wäre eine spezielle Ablaufeigenschaft, bei der ein Prozess eine benötigte Ressource nicht bis zu seinem Ende benötigt und daher früher freigibt. Ein weiteres Beispiel wäre eine Nachbedingung, welche einen Folgeprozess verpflichtet, die gleiche Ressource zu binden.

3.2 Das Verhaltensmodell

Im Verhaltensmodell werden das dynamische Verhalten der Objekte selbst und jenes zwischen ihnen beschrieben. Ein Beispiel aus der Produktion ist das Durchlaufen eines Werkstückes durch einen Maschinenpark. Die Modellierung des Verhaltensmodells kann nach der angestrebten Detailstufe und abhängig davon, ob bei dem Verhalten ein Rezept einzuhalten ist, klassifiziert werden (vgl. Abb. 3).Grundlegend muss zwischen Vorgängen, bei denen die Reihenfolge der Schritte eine Rolle spielt und anderen Vorgängen, unterschieden werden. Für die Abbildung der Vorgänge, bei denen die Reihenfolge der Schritte eine Rolle spielt, wird das SysML-Aktivitätsdiagramm von den Autoren favorisiert, bei der Abbildung von Ressourcen mit dynamischem Verhalten das Zustandsdiagramm.

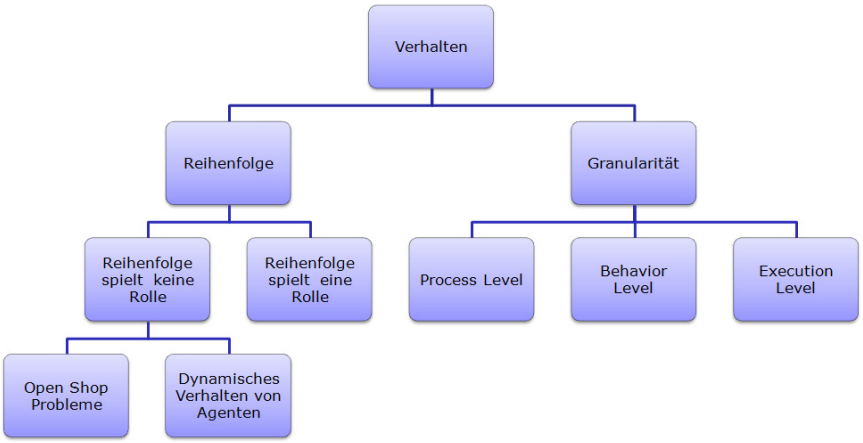

***Abbildung 3**: Übersicht Modellierung des Verhaltens*

Die Modellierung des Verhaltens kann zudem in verschiedenen Granularitätsstufen durchgeführt werden. So beschreiben beispielsweise Pieper und Röttgers, wie sie die Abbildung des Verhaltens von Workflows in Granularitätsstufen unterteilen (PIEPER, RÖTTGERS 2006, S.46 ff.). Auch STÖRRLE (2005, S. 194) weist darauf hin, dass Aktivitätsdiagramme verschiedene Granularitätsstufen beschreiben können.

Im Folgenden sollen die verschiedenen Granularitätsstufen für Modelle diskreter Prozesse anhand eines Beispiels aus der Produktion mit Hilfe von SysML-Aktivitätsdiagrammen erläutert werden. Die oberste Ebene soll allgemein als "Prozessebene" bezeichnet werden. Hier wird die Abfolge der einzelnen Prozesse

abgebildet, die für den Durchlauf eines Werkstücks durch den Maschinenpark steht. Das Beispiel in Abbildung 4 zeigt die Bearbeitung eines Werkstücks, welches das System im Ankunftsprozess betritt, auf einer Maschine bearbeitet wird, anschließend abkühlt und das System wieder verlässt. Für viele Simulatoren wird die in Abbildung 4 gezeigte Darstellung als Eingabe genügen (z.B. Anylogic, Flexsim, Simcron). Andere Simulationswerkzeuge jedoch benötigen eine genauere Spezifikation.

Jeder Prozess der Prozessebene kann als imaginäres Objekt klassifiziert werden (Ankunftsprozess, Abgangsprozess, Prozess, Warteschlange). Diesem können nun in der nächsten Granularitätsstufe "Verhaltensebene" festgelegte Folgen von Verhaltensmuster zugeordnet werden. In dem gewählten Beispiel wird "maschinell bearbeiten" beschrieben. In einem ersten Schritt bindet der Prozess die benötigten Ressourcen. Anschließend kann das Werkstück maschinell bearbeitet werden. Abschließend werden die gebundenen Ressourcen wieder freigegeben (vgl. Abb. 5).

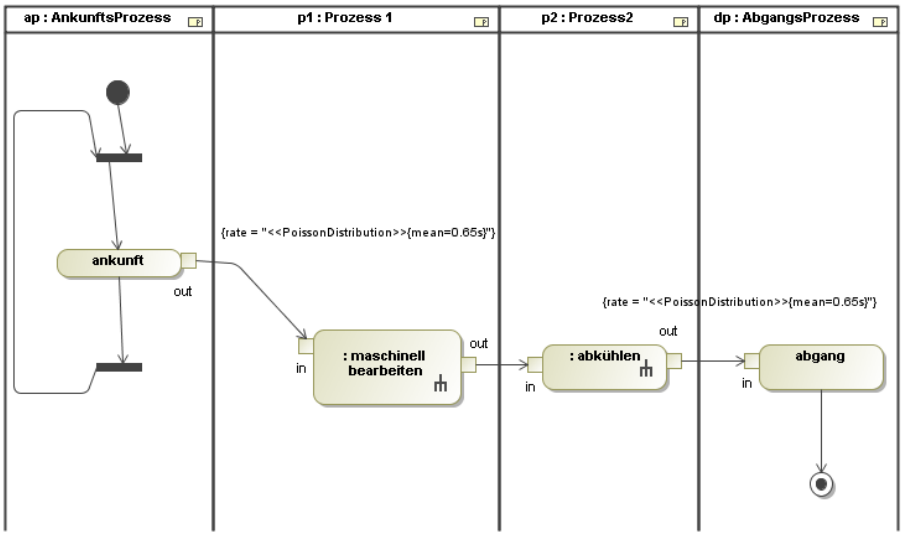

***Abbildung 4**: Beispiel Granularitätsstufe Prozessebene*

Während nun der Objekt- und Kontrollfluss des Verhaltens deterministisch und ausführbar beschrieben wird, fehlt noch die genaue Beschreibung der Aktionen, sobald diese zur Ausführung kommen. Hierfür stellt die OMG ca. 40 Aktionen als Elemente der UML 2 bereit, die eine Grundlage für die detailierte Beschreibung des Verhaltens darstellen (PIEPER, RÖTTGERS 2006, S.47). Die Aktivitäten werden in verschiedenen Veröffentlichungen ausführlich besprochen (WEILKIENS, OESTEREICH 2006, S 161 ff.; vgl. WEILKIENS 2006).

***Abbildung 5**: Beispiel Granularitätsstufe Verhaltensebene*

Das feingranulare Verhalten der Verhaltensmuster wird in der "Ausführungsebene" beschrieben. Im angeführten Beispiel wird das Verhalten der Aktion "Ressourcen binden" spezifiziert. Zuerst wird die Anzahl der benötigten Arbeiter mit den zur Verfügung stehenden Arbeitern verglichen. Sobald genug Arbeiter zur Verfügung stehen, werden die benötigten Arbeiter reserviert und von dem Ressourcenpool abgezogen (vgl. Abb. 6).

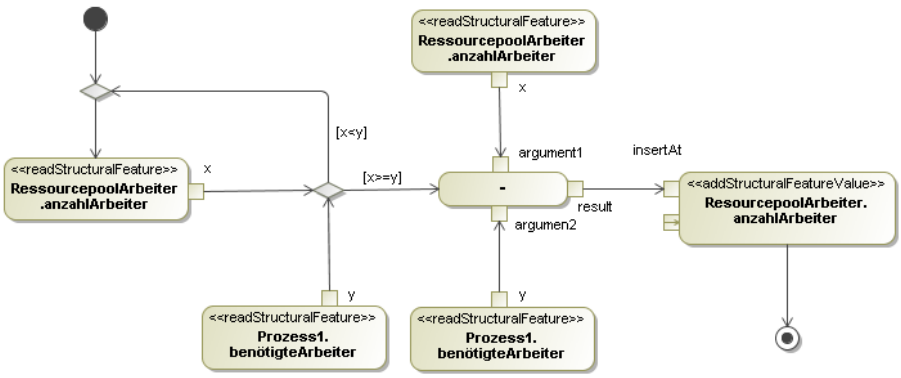

***Abbildung 6**: Beispiel Granularitätsstufe Ausführungsebene*

3.3 Kontrollebene

Das Kontrollmodell beschreibt die Produktionsplanung und Steuerung innerhalb eines Systems. Für die Modellierung der Steuerung sind die Schnittstellen zum Verhaltensmodell und Strukturmodell, die ausgetauschten Informationen und die Abbildung der Steuerungselemente wichtig. Elemente der Steuerung sind Eingangs-, Ausgangsrouten, Warteschlangen und das Anfordern von Ressourcen. Die Autoren konnten zeigen, dass sich das Strukturelle Modell und das Verhaltensmodell mit SysML abbilden lassen, das Kontrollmodell wird aktuell untersucht (vgl. SCHÖNHERR, ROSE 2009).

In derzeitigen Untersuchungen haben wir die Schnittstellen, Informationen und Elemente der Steuerung von Produktionsmodellen untersucht. In diesen Fällen war SysML mächtig genug, um sie abzubilden. Genauere Ausführungen zum Kontrollmodell würden jedoch den Rahmen dieses Artikels überschreiten.

4 Zusammenfassung und Ausblick

Es wurde versucht, der Modellierung von diskreten Systemen, insbesondere in Bezug auf Produktionsplanungs- und -steuerungsaufgaben, einen theoretischen Rahmen zu geben und deren signifikante Eigenschaften zu identifizieren und zu strukturieren (vgl. Kapitel 3). Sicher konnten nicht alle Eigenheiten identifiziert werden, da gerade die Steuerung solcher Systeme in der Regel sehr komplex ist. Der Entwurf von Produktionsszenarien und deren Übersetzung in verschiedene Simulationswerkzeuge konnte erfolgreich umgesetzt werden. Interessant wird es sein, Problemstellungen aus anderen Domänen, wie der Krankenhauslogistik oder dem Bauwesen zu überführen.

Durch eine abstrakte Sichtweise und einer großen Anzahl von Modellierungsmöglichkeiten ist die Anwendung von SysML für Ingenieure schwierig. Aktuell entwickeln wir ein auf TOPCASED basierendes Modellierungstool, das den Ingenieur in seinen Anforderungen unterstützt. Neben der Übersetzung der allgemein modellierten Szenarien für verschiedene Simulationswerkzeuge, arbeiten wir an einem SysML-Simulator. Dieser kann dann ohne Übersetzung direkt die mit SysML modellierten Szenarien simulieren.

Literatur

Fachausschuss Software Engineering: Code Generierung und modellbasierte Softwareentwicklung für Luft- und Raumfahrtsysteme. http://www.t6.dglr.de/Veranstaltungen/2004_MDSE/DGLR_T64_2004_bericht.html , Stand: 31.10.2009.

FOWLER, Martin: UML konzentriert: Eine kompakte Einführung in die Standardobjektmodellierungssprache. Heidelberg: Pearson Education, 2003.

GfSE – Gesellschaft für Systems Engineering e.V.: Tag des Systems Engineering. http://se-zert.com/index2.php?option=com_docman&task=doc_view&gid=9&Itemid=132, Stand: 27.04.2010.

PIEPPER, Daniel; RÖTTGERS, Carsten; GRUHN, Volker: MDA: effektives Software-Engineering mit UML 2 und Eclipse. Berlin: Springer Verlag, 2006.

SCHÖNHERR, Oliver; ROSE, Oliver: First Steps towards a general SysML model for discrete processes in production systems. In: Proceedings of the 2009 Winter Simulation Conference; Energy Alternatives. Hrsg.: ROSSETTI, M. D.; HILL, R. R.; JOHANSSON, B.; DUNKIN, A.; INGALLS, R. G. New York, NY: Association for Computing Machinery Order Department; Piscataway, NJ: IEEE Service Center, 2009, pp. 2335-2340.

STÖRRLE, Harald: UML 2 für Studenten. München: Pearson Verlag, 2005.

WEILKIENS, Tim: System Engineering mit SysML/UML. Heidelberg: dpunkt Verlag, 2006.

WEILKIENS, Tim; ÖSTEREICH, Bernd: UML-2-Zertifizierung : Fundamental, Intermediate und Advanced. Test-Vorbereitung zum OMG Certified UML Professional. Heidelberg: dpunkt Verlag, 2006.