# UML-basierte Modellierung

Peter Fettke

**Die Unified Modeling Language (UML) ist eine etablierte, objektorientierte, standardisierte und werkzeugunterstützte Modellierungssprache für die Visualisierung, Beschreibung, Spezifikation und Dokumentation von Systemen.**

## Einführung

Ausgangspunkt der Entwicklung der UML ist der Ende der 1980er beziehungsweise Anfang der 1990er Jahre herrschende Wildwuchs an objektorientierten Modellierungsansätzen. Daher beschlossen Mitte der 1990er Jahre Grady Booch, Ivar Jacobson und Jim Rumbaugh, drei bedeutende Vertreter verschiedener objektorientierter Methoden, einen konsolidierten Ansatz zu entwickeln, der die unterschiedlichen Varianten auf syntaktischer und semantischer Ebene vereinheitlicht. Ein erster Konsolidierungsvorschlag wurde von dem Unternehmen Rational Rose, in dem sich die drei Methodenexperten zusammengefunden hatten, entwickelt. Die Object Management Group (OMG) hat diesen Vorschlag aufgegriffen und 1997 mit der Version UML 1.1 erstmalig verabschiedet. In den darauf folgenden Versionen wurden Ergänzungen und Verbesserungen vorgenommen. Seit Ende der 1990er Jahre haben zahlreiche Personen und Institutionen intensiv an der UML Version 2.0 gearbeitet, die zunächst für das Jahr 2001 geplant, aber letztendlich erst im Jahr 2005 vollständig fertig gestellt wurde [Fettke 2007]. Aktuell ist die Version 2.5 gültig. Eine Standardisierung durch die International Standardization Organization (ISO) hat diese Version allerdings (noch) nicht erreicht. Diese bleibt bisher nur der Version 1.4.2 vorbehalten (ISO/IEC 19501:2005).

Die zwischenzeitlich eingeführte Aufteilung der Sprachspezifikation in zwei separate Dokumente wurde aufgegeben, sodass die gesamte Sprachspezifikation der UML 2.5 wieder aus einem einzigen Dokument besteht [OMG 2015]. Ergänzend sind die Spezifikationen der Object Constraint Language (OCL) und des Diagram Interchange von Interesse. Mithilfe der OCL können Invarianten sowie Vor- und Nachbedingungen von Operationen spezifiziert werden, wodurch die Präzision der Ausdruckmittel weiter erhöht wird [OMG 2012]. Die Speicherung und der Austausch von Diagrammen inklusive von Layoutinformationen wird vom Spezifikationsteil Diagram Interchange abgedeckt [OMG 2005].

## Diagrammarten und wesentliche Konzepte

Die UML 2.5 umfasst insgesamt 14 Diagrammarten, die in drei Gruppen untergliedert werden. Tabelle 1 gibt einen Überblick über die wesentlichen Konzepte aller Diagramme.

| Fokus | Diagramm (engl. Bezeichnung) | Zweck | Wesentliche Konzepte | Unterstützt seit |
|---|---|---|---|---|
| Struktur | Klassendiagramm (class diagram) | Klassendiagramm (class diagram) | Klasse, Merkmal, Beziehung | UML 1 |
| | Objektdiagramm (object diagram) | Exemplarische Konfiguration einer Instanz | Objekt, Verknüpfung | UML 1 (inoffiziell) |
| | Komponentendiagramm (component diagram) | Struktur und Verbindungen zwischen Komponenten | Komponente, Schnittstelle, Abhängigkeit | UML 1 |
| | Kompositionsstrukturdiagramm (composite structure diagram) | Dekomposition einer Klasse oder Komponente zur Laufzeit | Teil, Schnittstelle, Konnektor, Port | UML 2.0 |
| | Paketdiagramm (package diagram) | Zusammenhänge zwischen Paketen | Paket, Abhängigkeit | UML 1 (inoffiziell) |
| | Verteilungsdiagramm (deployment diagram) | Verteilung von Komponenten auf Knoten | Knoten, Komponente, Abhängigkeit | UML 1 |

| | | | | |
|---|---|---|---|---|
| | Profildiagram (profile diagram) | benutzerdefinierte Stereotype etc. | Paket, Profil | UML 2.2 |
| Verhalten | Anwendungsfalldiagramm (use case diagram) | Benutzerinteraktionen mit dem System | Anwendungsfall, Akteur | UML 1 |
| | Aktivitätsdiagramm (activity diagram) | Ablaufverhalten | Aktivität, Aktion, Übergang, Synchronisation | UML 1 |
| | Zustandsautomat (state machine diagram) | Ereignisse und Zustände während der Lebenszeit eines Objekts | Zustand, Übergang, Ereignis, Aktion | UML 1 statechart diagram |
| Interaktion | Sequenzdiagramm (sequence diagram) | Objektinteraktionen mit der Betonung von Sequenzen | Interaktion, Nachricht | UML 1 |
| | Kommunikationsdiagramm (communication diagram) | Objektinteraktionen mit der Betonung der Objektkonfiguration | Objektkonfiguration, Interaktion, Nachricht | UML 1 collaboration diagram |
| | Timing-Diagramm (timing diagram) | Objektinteraktionen mit der Betonung der zeitlichen Abstimmung | Objekt, zeitliche Abhängigkeit, Zustand, Ereignis | UML 2.0 |
| | Interaktionsübersichtsdiagramm (interaction overview diagram) | Zusammenspiel zwischen Aktivitäten und Sequenzen | Kombination von Sequenz- und Aktivitätsdiagramm (siehe dort) | UML 2.0 |

Tab. 1: Überblick über die Diagramme der UML 2.5

## Strukturdiagramme

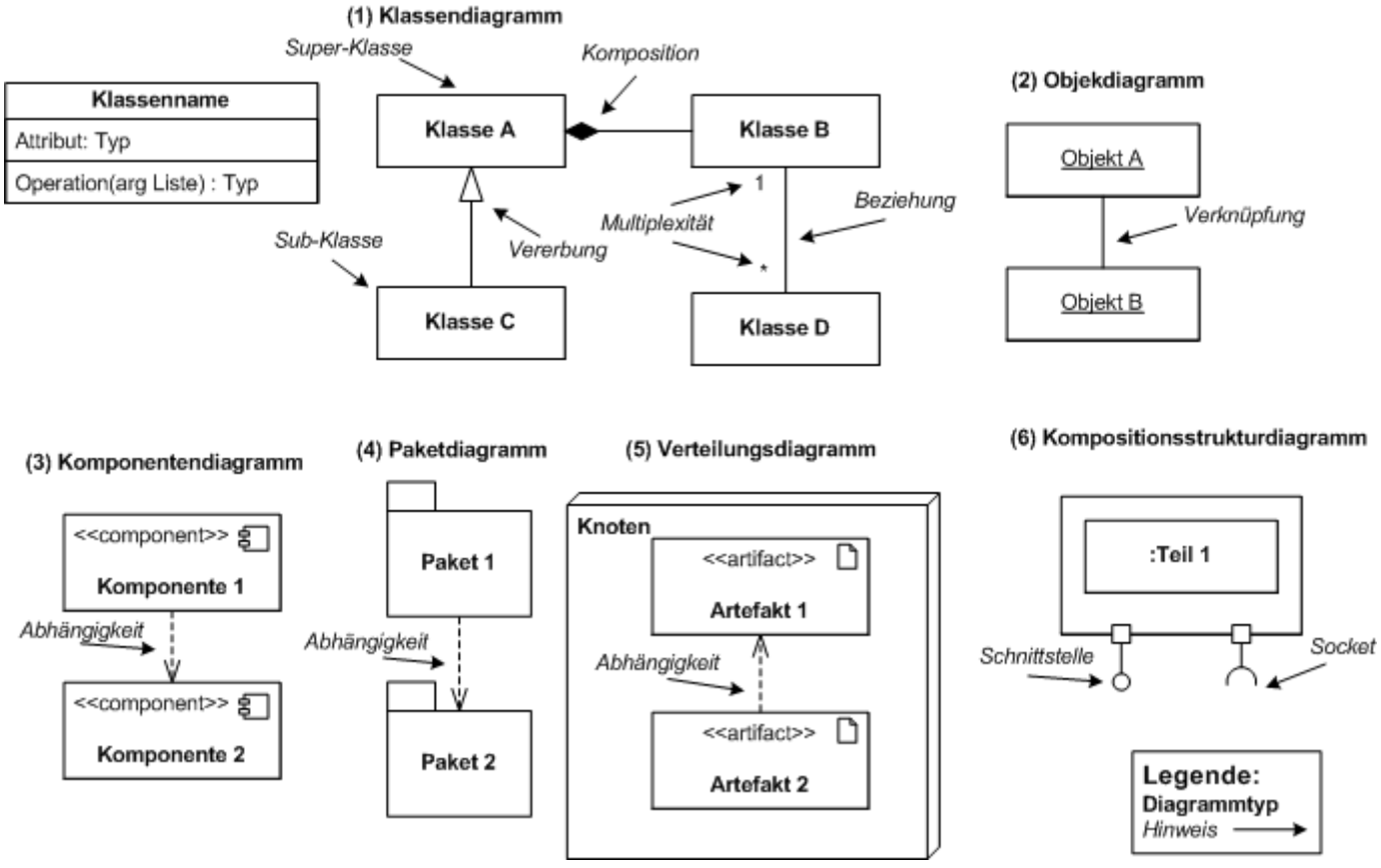


Abb. 1: Strukturdiagramme

Strukturdiagramme beschreiben statische Aspekte eines Gegenstands (Abbildung 1). Zur Beschreibung werden Klassen, Attribute, Operationen, Beziehungen, Schnittstellen und Konnektoren eingesetzt.

- Klassendiagramm: Ein Klassendiagramm repräsentiert verschiedene Klassen. Eine Klasse beschreibt ein wesentliches Konzept des zu modellierenden Gegenstands. Sie ist eine allgemeine Beschreibung einer Menge von Objekten, die eine ähnliche Struktur, ein ähnliches Verhalten und ähnliche Beziehungen besitzen. Ein Objekt ist eine bestimmte Instanz einer Klasse. Jede Klasse hat eine eindeutige Bezeichnung in einem bestimmten Geltungsbereich (normalerweise eines Pakets). Eine Klasse kann durch verschiedene Merkmale und Attribute beschrieben werden. Attribute besitzen einen Namen und werden durch einen bestimmten Datentyp näher spezifiziert. Operationen definieren Dienste, die von anderen Objekten ausgeführt werden können. Zwischen den Objekten können unterschiedliche Beziehungen definiert werden.
- Objektdiagramm: Ein Objektdiagramm veranschaulicht den Zustand eines Systems zu einem bestimmten Zeitpunkt. Es enthält eine Menge relevanter Objekte, die Werte ihrer Attribute und die Beziehungen zwischen den Objekten.
- Komponentendiagramm: Komponentendiagramme beschreiben die physische Struktur eines Softwaresystems zur Laufzeit. Die UML versteht unter Komponenten sowohl Quelltext- als auch binäre Dateien. Einfache Komponenten können zu komplexen Komponenten aggregiert werden, um Teil-Ganzes-Beziehungen abzubilden. Zwischen verschiedenen Komponenten können Abhängigkeiten spezifiziert werden, die besagen, dass eine Komponente sich auf eine andere Komponente abstützt.
- Paketdiagramm: Pakete erlauben es, verschiedene UML-Konstrukte zusammenzufassen und als eine zusammenhängende Einheit zu verstehen. Typischerweise werden unterschiedliche Klassen in einem Paket zusammengefasst. Es ist auch möglich, dass Pakete Bestandteil weiterer Pakete sind, sodass eine hierarchische Struktur zwischen Paketen gebildet wird. Das Paketdiagramm veranschaulicht die Pakete, die zur Beschreibung eines Systems benötigt werden.
- Verteilungsdiagramm: Während das Komponentendiagramm hauptsächlich die Abhängigkeiten zwischen Systemkomponenten zur Entwicklungszeit verdeutlicht, beschreiben Verteilungsdiagramme die Anordnung der Systemkomponenten auf unterschiedliche Rechnerknoten zur Laufzeit.
- Kompositionsstrukturdiagramm: Das Kompositionsstrukturdiagramm verdeutlicht die Aufteilung einer Klasse in unterschiedliche Teile. Die dekomponierten Teile werden über Schnittstellen repräsentiert. Jedes Teil benötigt oder erfordert eine bestimmte Schnittstelle.
- Profildiagramm: Das Profildiagramm visualisiert benutzterdefinierte Stereotypen, Eigenschaftswerte ("tagged values") und Randbedingungen ("Constraints").

## Verhaltensdiagramme

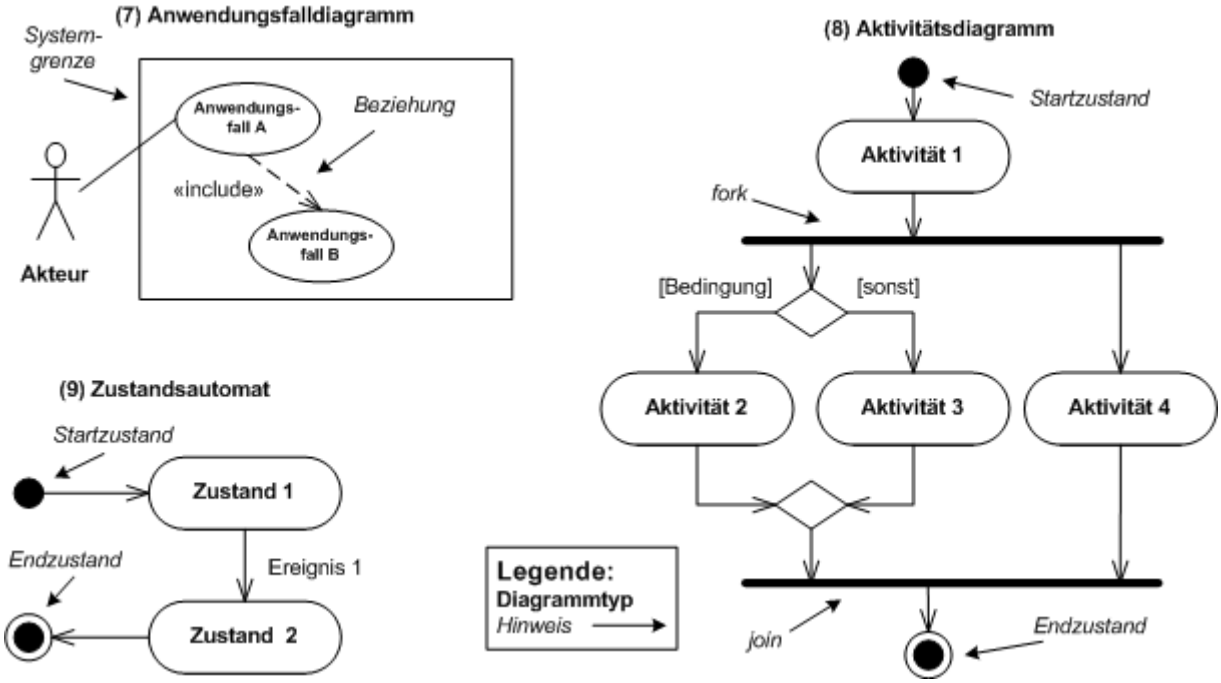


Abb. 2: Verhaltensdiagramme

Verhaltensdiagramme beschreiben die Dynamik zwischen Objekten eines Systems mithilfe von Interaktionen, Objektkonfigurationen und Zustandshistorien (Abbildung 2).

- Anwendungsfalldiagramm: Ein Anwendungsfall spezifiziert eine für einen Anwender wahrnembare Leistung eines Systems. Das Anwendungsfalldiagramm enthält eine Menge von Anwendungsfällen,

Aktoren und Beziehungen zwischen den Elementen. Ein Aktor kann sowohl einen Menschen als auch eine Maschine repräsentieren.
- Aktivitätsdiagramm: Aktivitätsdiagramme beschreiben Verhaltensaspekte eines Systems, an dem mehrere Objekte beteiligt sind. Sie werden häufig zur Geschäftsprozessbeschreibung benutzt. Spezielle Konstrukte erlauben eine Parallelisierung und Synchronisation von Aktivitäten. Bedingte Verzweigungen werden über sogenannte Wächter realisiert.
- Zustandsmaschine: Das Verhalten einzelner Objekte wird über Zustandsmaschinen spezifiziert. Transitionen beschreiben mögliche Wechsel von Zuständen eines Objekts. Ein Zustandswechsel wird durch ein bestimmtes Ereignis ausgelöst. Zwei ausgezeichnete Zustände kennzeichnen den Beginn und das Ende eines Verhaltensablaufs.

### Interaktionsdiagramme

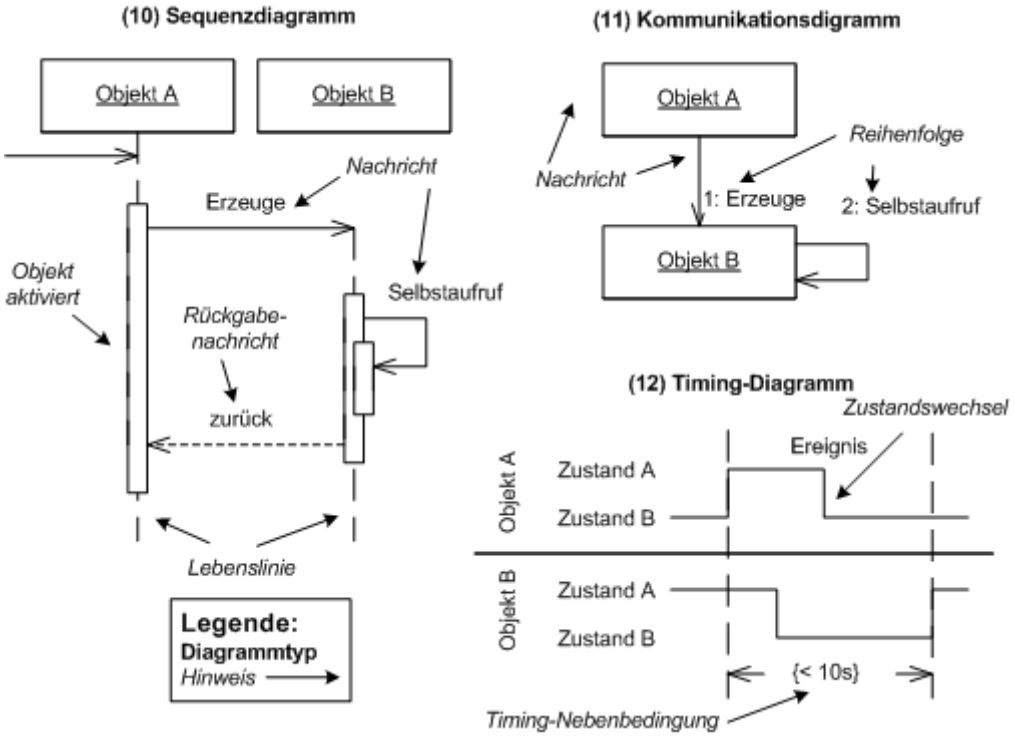


Abb. 3: Interaktionsdiagramme

Bei den Interaktionsdiagrammen handelt es sich um spezielle Verhaltensdiagramme, die für besondere Zwecke der Modellierung eingesetzt werden (Abbildung 3).

- Sequenzdiagramm: Sequenzdiagramme beschreiben Interaktionen zwischen verschiedenen Objekten. Eine Interaktion besteht aus einer partiell geordneten Menge von Nachrichten, die zwischen den Teilnehmern der Interaktion ausgetauscht werden.
- Kommunikationsdiagramm: Während Sequenzdiagramme den zeitlichen Ablauf der Interaktion hervorheben, betonen Kommunikationsdiagramme die Rollen der an der Interaktion beteiligten Teilnehmer. Die Reihenfolge der Nachrichten wird durch eine Sequenznummer beschrieben.
- Timing-Diagramm: Spezielle zeitliche Restriktionen zwischen Zustandsübergängen werden über Timing-Diagramme beschrieben. Mögliche Restriktionen umfassen minimale und maximale Zeitabläufe, die zwischen zwei Ereignissen eingehalten werden müssen.
- Interaktionsdiagramm: Interaktionsdiagramme verbinden Elemente von Aktivitäts- und Sequenzdiagrammen. Eine Möglichkeit zur Nutzung dieser Diagrammart besteht darin, die in einem Aktivitätsdiagramm enthaltenen Aktivitäten durch zusätzliche Sequenzdiagramme feiner zu untergliedern.

## Aktuelle Entwicklungen

Die UML entwickelt sich in unterschiedliche Richtungen weiter. Die wichtigsten Trends umfassen:

- Model Driven Architecture (MDA): MDA zielt darauf ab, den kompletten Lebenszyklus eines Softwaresystems durch Modelle zu unterstützen [Fettke, Loos 2003, S. 555]. Die UML ist ein wichtiger Basisstandard, mit dem dieses Ziel erreicht werden soll.

- Executable UML: Executable UML reichert die Modellierungskonzepte um eine ausführbare Semantik an [Mellor, Balcer 2002]. Hier wurde mit der UML 2.0 schon ein wesentlicher Fortschritt im Bereich der Verhaltensmodellierung erzielt, der auch in Zukunft weiter auszubauen ist.
- Ontologische Analyse und Semantik: Diese Forschungsanstrengungen bewerten die UML aus einer ontologischen Perspektive und reichern die UML mit einer realweltlichen Semantik an [Evermann, Wand 2006]. Das Ziel der Analyse ist es zu überprüfen, ob sämtliche Konstrukte der UML auf einer Ontologie abgebildet werden können.
- Anpassung der UML und domänenspezifische Modellierungssprachen: UML ist eine Modellierungssprache, die nicht auf bestimmte Zwecke ausgerichtet ist. Daher spielt die Anpassung der Sprache an besondere Einsatzgebiete eine wichtige Rolle. Hierfür werden unterschiedliche Anpassungskonzepte diskutiert [France et al. 2006, S. 61].
- Modellbibliotheken: Die UML wird zunehmend benutzt, um Referenzmodelle standardisiert zu beschreiben [Fettke, Loos, Zwicker 2007].
- Empirische Fundierung: Bisher war die Forschung zur UML hauptsächlich konzeptionell ausgerichtet. Zukünftige Forschung muss unter anderem folgende Fragen beantworten: Wie wird die UML in der Praxis genutzt? Sind sämtliche Konstrukte der UML relevant für die Modellierung in der Praxis? Erhöht die UML die Qualität von Informationssystemen und die Produktivität von Entwicklern?

# Vor- und Nachteile

Die Standardisierung von Modellierungssprachen ist mit verschiedenen Vorteilen verbunden. Im Wesentlichen sind hier zu nennen [Frank 1997, S. 13]:

- Investitionsschutz für Modellierungswerkzeuge,
- einfacher Modellaustausch,
- bessere Modellwiederverwendung,
- höhere Professionalisierung und
- besseres Training.

Diesen Vorteilen stehen auch Nachteile gegenüber [Fettke 2005, S. 2926]:

- Die UML wird hinsichtlich semantischer Inkonsistenzen, Konstruktmehrdeutigkeiten, inadäquate Notationen und kognitiven Unzugänglichkeiten kritisiert.
- Die hohe Komplexität der Sprache (beispielsweise umfasst die Sprachspezifikation aktuell rund 800 Seiten) verursacht Schwierigkeiten bei ihrer Benutzung.
- Bisher sind keine Werkzeuge verfügbar, die den Sprachstandard vollständig unterstützen.
- Die Pflege des Standards ist schwerfällig und leicht fehleranfällig.

Dennoch bildet die UML einen wichtigen Meilenstein für die Systementwicklung. Wie empirische Untersuchungen belegen, nimmt die UML trotz vorliegender Nachteile zunehmend die Rolle einer Lingua franca der Informationsmodellierung ein [Fettke 2008].

## Literatur


Evermann, Joerg ; Wand, Yair: Ontological Modelling Rules for UML: An Empirical Assessment. In: Journal of Computer Information Systems (2006), Nr. 5, S. 14-29.

Fettke, Peter ; Loos, Peter: Model Driven Architecture (MDA). In: Wirtschaftsinformatik (2003), Nr. 5, S. 555-559.

Fettke, Peter (2005): Unified Modeling Language. In: Mehdi Khosrow-Pour (Hrsg.): Encyclopedia of Information Science and Technology, Volume I-V. Idea: Hershey, PA, USA, et al., 2005, S. 2921-2928.

Fettke, Peter: Unified Modeling Language 2.0. In: Wirtschaftsinformatik (2007), Nr. 1, S. 55-58.

Fettke, Peter ; Loos, Peter ; Zwicker, Jörg (2007): Using UML for Reference Modeling. In: Peter Rittgen (Hrsg.): Enterprise Modeling and Computing with UML. Idea: Hershey, PA, USA, et al. 2007, S. 174-205.

Fettke, Peter: Ansätze der Informationsmodellierung und ihre betriebswirtschaftliche Bedeutung: Eine Untersuchung der Modellierungspraxis in Deutschland. In: Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung (2008), Nr., S. im Druck.

France, Robert B. ; Ghosh, Sudipto ; Dinh-Trong, Trung ; Solberg, Arnor: Model-Driven Development Using UML 2.0: Promises and Pitfalls. In: Computer (2006), Nr. 2, S. 59-66.

Frank, Ulrich: Towards a Standardization of Object-Oriented Modelling Languages? Institut für Wirtschaftsinformatik der Universität Koblenz Landau, 3. Koblenz, Germany, 1997.

Mellor, Stephen J. ; Balcer, Marc J.: Executable UML: A Foundation for Model-Driven Architecture. Addison Wesley : Boston et al., 2002.

OMG: Unified Modeling Language: Diagram Interchange, Version 2.0, ptc/05-06-04. Needham 2005.

OMG: Unified Modeling Language, Version 2.5, formal/2015-03-01. Needham 2015.

OMG: Object Constraint Language, Version 2.3.1, formal/2012-01-01. Needham 2012.

## Autor

Prof. Dr. Peter Fettke, Institut für Wirtschaftsinformatik (IWi) im Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI), Campus D3 2, 66123 Saarbrücken

Autoreninfo

---

Zuletzt bearbeitet: 22.02.2021 15:10
Letzter Abruf: 11.08.2021 13:23