

2 Modellierungssprachen

2.1 Modellierungssprachen für Daten

2.1.1 Entity-Relationship-Diagramme (ERM)

Das Entity-Relationship-Modell (ERM) wurde von CHEN entwickelt und erstmalig im Jahre 1976 veröffentlicht.¹ Seitdem ist es sowohl in der Informatik, insbesondere im Datenbankentwurf, als auch in anderen Disziplinen, in denen die Modellierung von Zusammenhängen der realen (oder gedachten) Welt auf einer abstrakten Ebene eine Rolle spielt, die wohl am weitesten verbreitete Modellierungssprache zur Datenmodellierung. Die Popularität des ER-Modells ist im Wesentlichen auf zwei Gründe zurückzuführen:² Zum einen kommt die Sprache mit nur zwei Grundkonstrukten aus, die von den meisten Nutzern als sehr natürliche Ausdrucksmittel empfunden werden. Zum anderen erlaubt sie die Darstellung mitunter komplexer konzeptioneller Zusammenhänge mittels einer leicht verständlichen, graphischen Notation.

Bei den zwei Grundkonstrukten der Sprache handelt es sich, wie schon aus ihrem Namen erkenntlich wird, um *Entitäten (Entities)*, konkrete Objekte der realen Welt, und *Beziehungen (Relationships)* zwischen Objekten. Im Kontext eines Handelsunternehmens können beispielsweise die Entitäten Müller GmbH, Ertel AG und Schmidt & Lehmann GbR existieren. Gleichartige Entitäten werden zu sogenannten Entitätstypen zusammengefasst. So werden die im Beispiel genannten Entitäten zum Entitätstyp *Lieferant* zusammengefasst. Analog enthält der Entitätstyp *Artikel* als Entitäten alle von einem Handelsunternehmen gehandelten Artikel. Weitere Beispiele für Entitätstypen sind Personen, Orte oder auch immaterielle Objekte wie die Zeit.

Beziehungen stellen semantische Zusammenhänge zwischen Entitäten dar, zum Beispiel: Lieferant Müller GmbH liefert Artikel Rennrad Le Tour. Analog zu den Entitäten werden gleichartige Beziehungen zu Beziehungstypen zusammengefasst. Die obige Lieferant-Artikel-Zuordnung wird so zu einem Beziehungstyp *Bezugsnachweis* verallgemeinert, der einen Zusammenhang zwischen den Entitätstypen Lieferant und Artikel herstellt. Die Verbindung von Entitätstypen mit Beziehungs-

¹ Vgl. Chen (1976).

² Vgl. im Folgenden Vossen (2008), S. 80 f.

typen wird durch die sogenannte *Kardinalität* weiter charakterisiert. Diese gibt an, wie oft eine Entität eines Entitätstypen in einen Beziehungstypen eingehen kann. Zur Darstellung der Kardinalität wird die (min, max)-Notation genutzt.³ Der min-Wert gibt dabei an, wie oft eine Entität mindestens eine Beziehung eingeht und der max-Wert gibt an, wie oft eine Entität maximal eine Beziehung eingeht. Dementsprechend kann in dem Beispiel in Abbildung 2 ein Lieferant keinen oder mehrere Artikel liefern, und ein Artikel kann von keinem (z. B. bei selbsterstellten Produkten) oder mehreren Lieferanten geliefert werden. Es handelt sich um eine (0,n:0,m)-Beziehung. Im Gegensatz dazu handelt es sich bei der Zuordnung von Artikeln zu Warengruppen um eine (1,1:0,m)-Beziehung oder Existenzabhängigkeit. Ein Artikel ist genau einer Warengruppe zugeordnet, eine Warengruppe kann jedoch mehrere Artikel umfassen. In einer (1,1:1,1)-Beziehung steht ein Objekt der ersten Entitätsgruppe mit genau einem Objekt der zweiten Entitätsgruppe in Verbindung.

Objekte und Beziehungen in der realen Welt besitzen weitere beschreibende Eigenschaften. Diese können im ERM durch *Attribute* spezifiziert werden. Betrachtet man den Entitätstyp Artikel, so sind mögliche Attribute Artikelnummer, Bezeichnung, Maße, Gewicht oder Farbe. Um ein konkretes Objekt der realen Welt eindeutig identifizieren zu können, muss aus der Menge der Attribute ein sogenanntes Schlüsselattribut bestimmt werden. Artikel sind beispielsweise eindeutig durch ihre Artikelnummer identifizierbar. Auch Beziehungen können Attribute besitzen, die die Verbindung zwischen Objekten genauer beschreiben. Ein Beispiel hierfür stellen Preise dar. So besitzt ein und derselbe Artikel bei unterschiedlichen Lieferanten in der Regel unterschiedliche Einkaufspreise. Deshalb kann die Preisinformation nicht als Attribut des Entitätstypen Artikel modelliert werden, sondern muss als Attribut des Beziehungstypen Bezugsnachweis abgebildet werden. Auch Beziehungen müssen eindeutig identifizierbar sein. Dazu wird automatisch ein zusammengesetztes Schlüsselattribut aus den Schlüsselattributen der an der Beziehung beteiligten Entitätstypen gebildet.

³ Zur (min, max)-Notation vgl. Schlageter, Stucky (1983), S. 50 f.



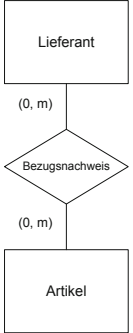


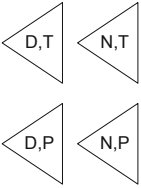
Element	Erläuterung	Symbol
Entitätstyp	Ein Entitätstyp repräsentiert eine homogene Gruppe von Entitäten (Objekten).	
Beziehungstyp	Ein Beziehungstyp repräsentiert eine Beziehung zwischen mindestens zwei Entity-Typen, wobei es sich bei den zwei Entity-Typen auch um denselben Entity-Typen handeln kann.	
Beziehung mit Kardinalität	Es existieren drei Arten von Beziehungen: „Eins-zu-eins“-Beziehungen (jedes Objekt der ersten Menge steht in Beziehung zu genau einem Objekt der zweiten Menge), „Eins-zu-viele“-Beziehungen (jedes Objekt der ersten Menge geht mit genau einem Objekt der zweiten Menge eine Beziehung ein, während jedes Objekt der zweiten Menge mit mehreren Objekt der ersten Menge eine Beziehung eingeht) und „Viele-zu-viele“-Beziehungen (jedes Objekt der ersten Menge geht mit mehreren Objekten der zweiten Menge eine Beziehung ein und umgekehrt). Die Kardinalitäten werden in der (min, max)-Notation dargestellt, die spezifiziert, wie oft ein Objekt einer Menge mindestens und höchstens mit einem Objekt der anderen Menge eine Beziehung eingehen muss/kann.	
Attribut	Ein Attribut beschreibt die Eigenschaften eines Entitäts- oder Beziehungstyps. Schlüsselattribute (unterstrichen) ermöglichen die eindeutige Identifikation einer Entität (Objektinstanz) eines Entitätstypen. Der Wertebereich eines Attributs wird Domäne genannt.	
Uminterpretierter Beziehungstyp	Ein uminterpretierter Beziehungstyp ist ein Beziehungstyp, der als Entitätstyp zu interpretieren ist und somit mit anderen Beziehungstypen eine Beziehung eingehen kann.	
Generalisierung bzw. Spezialisierung	Eine Generalisierung fasst zwei Entitätstypen zu einem verallgemeinerten Entitätstypen zusammen. Gemeinsame Attribute der spezifischen Entitätstypen werden dem verallgemeinerten Entitätstypen zugeordnet und spezifische Attribute verbleiben bei den spezifischen Entitätstypen. Eine Spezialisierung repräsentiert den umgekehrten Sachverhalt. Die Eigenschaften einer Generalisierung bzw. Spezialisierung lassen sich nach den Entitätstypen in disjunkte (D) und nicht disjunkte (N) Mengen, sowie totale (T) und partielle (P) Mengen aufteilen.	

Tabelle 1: Elemente des Entity-Relationship-Modells

Bei der Modellierung kommt der Abstraktion von Sachverhalten eine bedeutende Rolle zu. Im Rahmen der Datenmodellierung haben sich mit der Klassifikation, Generalisierung bzw. Spezialisierung und Aggregation allgemein anerkannte Abstraktionskonzepte herausgebildet.⁴

Die Klassifikation wird im ERM durch die Zuordnung von gleichartigen Entitäten zu übergeordneten Entitätstypen ermöglicht. Dabei wird von den Eigenschaften des konkreten Objektes der Realwelt abstrahiert, und es werden nur solche Eigenschaften beachtet, die allen Objekten eines Entitätstypen gemein sind. Demnach ist bei der Bildung von Attributen darauf zu achten, dass die Attribute eines Entitätstypen so allgemeingültig gewählt werden, dass sie auch auf alle untergeordneten Entitäten zutreffen. So wäre das Attribut Rahmengröße ein ungeeignetes Attribut für den Entitätstypen Artikel, da nicht alle Entitäten (z. B. Fußbälle) diese Eigenschaft teilen.

Ein weiteres Mittel zur Abstraktion ist die Generalisierung bzw. Spezialisierung. Sie wird durch das ERM seit der Erweiterung des ursprünglichen Konzeptes durch Smith und Smith unterstützt.⁵ Die Generalisierung bzw. Spezialisierung vollzieht sich, indem zu zwei oder mehreren Entitätstypen ein übergeordneter Entitätstyp gebildet wird bzw. vice versa. So können beispielsweise die Entitätstypen Lieferant und Kunde zum Entitätstyp Geschäftspartner generalisiert werden. Die Spezialisierung bezeichnet den komplementären Fall, ein Entitätstyp wird in zwei oder mehr untergeordnete Entitätstyp unterteilt. So kann bei Kunden beispielsweise zwischen Groß- und Kleinkunden oder A-, B- und C-Kunden unterschieden werden. In beiden Fällen werden die Attribute des übergeordneten Entitätstypen automatisch an die untergeordneten Entitätstypen übertragen. Man spricht in diesem Zusammenhang auch von Vererbung. Ferner kann zwischen disjunkter (d. h. eine Entität eines übergeordneten Entitätstypen kann genau einem untergeordneten Entitätstypen zugeordnet werden) und nicht-disjunkter (d. h. eine Entität eines übergeordneten Entitätstypen kann mehreren untergeordneten Entitätstypen zugeordnet werden) sowie vollständiger (d. h. eine Entität eines übergeordneten Entitätstypen muss einem untergeordneten Entitätstypen zugeordnet werden) und partieller (d. h. eine Entität eines übergeordneten Entitätstypen muss nicht zwangsweise einem untergeordneten Entitätstypen zugeordnet werden) Generalisierung unterschieden werden.

Unter der Aggregation wird die Kombination von bestehenden Entitätstypen verstanden, welche im ERM durch Beziehungstypen abgebildet wird. Eine besondere Form der Aggregation stellt die Uminterpretation von Beziehungstypen dar.⁶ Da Beziehungstypen zueinander nicht in Beziehung gesetzt werden dürfen, bedarf es in manchen Fällen der Uminterpretation von Beziehungstypen zu Entitätstypen.

⁴ Vgl. Vossen (2008), S. 79 f.

⁵ Vgl. Smith, Smith (1977), S. 107 ff.

⁶ Vgl. Scheer (1997), S. 38 f.

Ein Beispiel hierfür ist die Modellierung des Aktionsgeschäfts im Handel. Eine Aktion betrifft in der Regel nur ausgewählte Artikel. Diese Aktionsartikel werden als Beziehungstyp zwischen den Entitätstypen Artikel und Aktion abgebildet. Möchte man nun Aktionsartikel einer bestimmten Filiale zuordnen, so ist der Beziehungstyp zu einem Entitätstypen umzuinterpretieren, um mit dem Entitätstyp Filiale in Beziehung gesetzt werden zu können (Abbildung 3). Das Beispiel verdeutlicht die teilweise auftretenden Schwierigkeiten bei der Unterscheidung zwischen Entitäts- und Beziehungstypen.

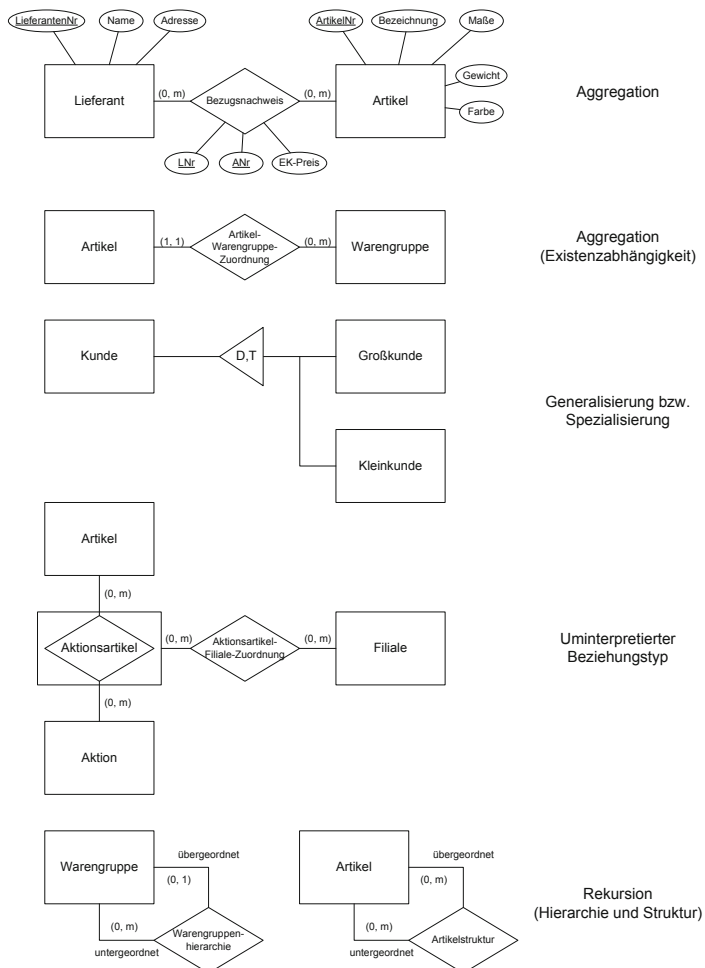


Abbildung 2: Beispiele für die Anwendung der Konstrukte des ERM

Neben diesen grundlegenden Konzepten zur Abstraktion spielt die Rekursion eine wichtige Rolle. Sie bezeichnet Beziehungen, die ein Entitätstyp mit sich selbst eingeht. Es kann zwischen der Hierarchie und Struktur unterschieden werden.⁷ Hat jedes übergeordnete Objekt der realen Welt mehrere untergeordnete Objekte und jedes untergeordnete Objekt nur ein übergeordnetes Objekt, so liegt eine Hierarchie vor. Dies ist beispielsweise bei einer Warengruppenhierarchie der Fall. Sind mehrere übergeordnete Objekte möglich, so liegt ein Netz oder eine Struktur vor. So können Artikel zum Beispiel für Zwecke des Category Managements in Strukturen (z. B. Lots, Sets, Displays) organisiert sein.

2.1.2 Klassendiagramme (UML)

Klassendiagramme sind ein Modelltyp der Unified Modelling Language (UML), die von der Object Management Group (OMG) im Jahr 1997 entwickelt und durch die International Organization for Standardization (ISO) im Jahr 2005 in der zweiten Version standardisiert wurde.⁸ Das zentrale Konstrukt dieser Modellierungssprache ist, wie schon aus ihrem Namen erkenntlich wird, die Klasse. Klassen beschreiben, ähnlich wie Entitätstypen aus den ER-Modellen, Typen realweltlicher Objekte oder Objekte der Vorstellungswelt. Klassen besitzen dabei eine Menge von Attributen und Operationen, die ein korrespondierendes realweltliches Objekt charakterisieren. Attribute beschreiben Eigenschaften einer Klasse. Operationen bilden ihr Verhalten ab. Eine Klasse wird durch eine dreiteilige Box dargestellt (Abbildung 3). Der obere Abschnitt enthält den Namen einer Klasse, der mittlere Teil die Attribute und der untere Abschnitt die Operationen. Anders als ein ER-Modell enthält ein UML-Klassendiagramm implementierungsnahe Angaben über den Daten- bzw. Rückgabebetyp eines Attributes oder einer Operation sowie über deren Sichtbarkeit. UML-Klassendiagramme können daher umgehend in Quellcode übersetzt werden.

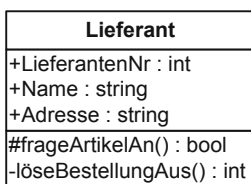


Abbildung 3: Eine Klasse in UML

Das erste Element einer Attributbeschreibung stellt seine Sichtbarkeit dar. Ein Pluszeichen (+) gibt dabei an, dass das entsprechende Attribut der Klasse public

⁷ Vgl. Becker, Schütte (2004), S. 90 f.

⁸ Vgl. im Folgenden ISO/IEC 19501 (2005).

ist. Dies bedeutet, dass nicht nur Objekte dieser Klasse, sondern auch Objekte jeder anderen Klasse dieses Attribut sehen, d. h. den Wert des Attributes auslesen und verändern können. Public-Attribute werden darüber hinaus an eventuelle Subklassen weitervererbt. Ein Minuszeichen (-) hingegen steht für die Sichtbarkeit private und drückt aus, dass dieses Attribut nur von Objekten der eigenen Klasse ausgelesen und verändert werden kann. Private-Attribute werden daher auch nicht weitervererbt. Je nach eingesetzter Programmiersprache sind ferner noch die Sichtbarkeitstypen protected und package zu unterscheiden. Ist ein Attribut protected, so bedeutet dies, dass es nur von Objekten der eigenen Klasse sowie von Subklassen gesehen werden kann. Objekte anderer Klassen können auf dieses Attribut folglich nicht zugreifen. Dieser Sichtbarkeitstyp wird in UML durch eine Raute (#) dargestellt. Besitzt ein Attribut hingegen den Sichtbarkeitstyp package, so ist es innerhalb eines Paketes sichtbar. Ein Paket beschreibt eine Menge zusammengehörender Klassen, die einen einheitlichen Namensraum bilden. Die Package-Sichtbarkeit wird in UML durch eine Tilde (~) dargestellt.

Das zweite Element einer Attributbeschreibung beschreibt den Namen des Attributes. Anschließend folgen ein Doppelpunkt und die Spezifikation des Datentyps. Ein Integer (int)-Attribut stellt beispielsweise eine natürliche Zahl dar, während ein String-Attribut aus einer Zeichenkette besteht.

Die Beschreibung einer Operation besteht aus ähnlichen Elementen. Der Sichtbarkeitstyp gibt an, welche Objekte welcher Klassen auf diese Operation zugreifen dürfen. Anschließend folgt der Name der Operation. In Klammern können eventuelle Parameter übergeben werden. Auf den Doppelpunkt folgt die Spezifikation des Rückgabewertes. So gibt die Operation `frageArtikelAn()` beispielsweise einen booleschen Wert zurück (Abbildung 3), der besagt, ob der jeweilige Lieferant einen bestimmten Artikel liefern kann oder nicht.

Einzelne Klassen können über Assoziationen miteinander in Beziehung gesetzt werden. UML unterscheidet zwischen unären und binären Assoziationen. Eine unäre Assoziation besteht aus einem Pfeil, der von der Quell- zur Zielklasse führt (Abbildung 4). Die Zielklasse erhält ein Attribut, welches die Assoziation beschreibt. So beinhaltet die Klasse *Warengruppe* in Abbildung 4 ein Attribut, das eine Liste aller Artikel dieser Warengruppe enthält.

Bei einer binären Assoziation hingegen erhalten beide Klassen ein zusätzliches Attribut. So erhält sowohl die Klasse *Filiale* ein Attribut, das auf die Liste aller Mitarbeiter dieser Filiale verweist, als auch die Klasse *Mitarbeiter* (Abbildung 4), dessen Attribut auf die jeweilige Filiale verweist, bei der der Mitarbeiter angestellt ist.

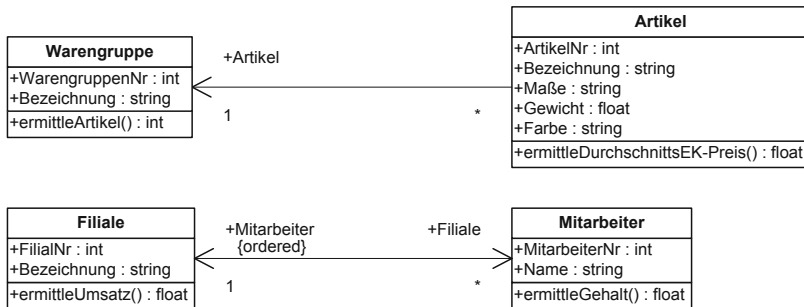


Abbildung 4: Unäre und binäre Assoziation

Assoziationen können bestimmte Eigenschaften aufweisen. Diese Eigenschaften werden in UML mit Hilfe von geschweiften Klammern annotiert. Beispiele für solche Eigenschaften sind `{ordered}` und `{readOnly}`. Die Eigenschaft `{ordered}` beschreibt, dass ein Attribut als Liste zu implementieren ist, deren Einträge beispielsweise alphabetisch sortiert sind (Abbildung 4). Die Eigenschaft `{readOnly}` hingegen besagt, dass ein Attribut nur ausgelesen und nicht verändert werden kann.

Gehen mehr als zwei Klassen eine Beziehung ein, so bietet UML die n-äre Assoziation an. Ähnlich wie in ER-Modellen wird zur Darstellung dieses Assoziations-typs eine Raute verwendet.

Möchte man die Assoziation zweier Klassen näher beschreiben, so stellt ein UML-Klassendiagramm das Konstrukt der Assoziationsklasse zur Verfügung. Damit können einer Assoziation eigene Attribute und Operationen zugewiesen werden. Mit ihrer Hilfe lassen sich many-to-many Beziehungen sinnvoll abbilden. So erhält die Assoziationsklasse *Bezugsnachweis* in Abbildung 5 beispielsweise das Attribut *EK-Preis*, welches festlegt, zu welchem Einkaufspreis ein bestimmter Artikel bei einem Lieferanten bezogen werden kann. Diese Art der Modellierung drückt aus, dass es nur genau einen Einkaufspreis für eine Kombination aus Artikel und Lieferant gibt.

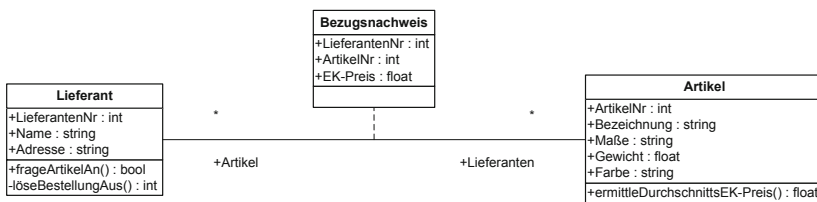


Abbildung 5: Assoziationsklasse

Um eine bestimmte Stärke einer Assoziation zu betonen, führt ein UML-Klassendiagramm das Konstrukt der Aggregation bzw. der Komposition ein. Es beschreibt eine „besteht aus“-Beziehung zweier Klassen. Eine Komposition stellt

dabei eine stärkere und eine Aggregation eine schwächere Assoziation dar. So besteht eine Bestellung unter anderem aus mehreren Bestellpositionen. Die Komposition drückt dabei aus, dass eine Bestellposition nicht ohne die zugehörige Bestellung existieren kann (Abbildung 6). Wird ein Objekt der Bestellklasse gelöscht, so werden auch alle entsprechenden Positionsobjekte mit gelöscht.



Abbildung 6: Komposition

Demgegenüber beschreibt eine Aggregation zwischen zwei Klassen eine Assoziation, bei der Objekte der einen Klasse auch ohne die Objekte der jeweiligen anderen Klasse existieren können. Möchte der Modellierer beispielsweise die Stärke der Beziehung zwischen Filiale und Mitarbeiter betonen, so kann er sie wie in Abbildung 7 darstellen. Im Gegensatz zur Modellierung als Komposition kann eine Filiale aber auch ohne Mitarbeiter existieren und umgekehrt.



Abbildung 7: Aggregation

Analog zu ER-Modellen können in UML-Klassendiagrammen darüber hinaus auch Kardinalitäten modelliert werden. Diese geben an, wie oft ein Objekt einer Klasse in eine Assoziation eingehen kann. In Abbildung 4 gehört ein Artikel zu genau einer Warengruppe, während eine Warengruppe über null bis viele Artikel verfügen kann. Anders als in ER-Modellen wird die Kardinalität einer Klasse dabei nicht direkt an der Klasse selbst annotiert, sondern bei der gegenüberliegenden Klasse. Nur bei n-ären Assoziationen werden Kardinalitäten wie in ER-Modellen annotiert, da nur so erkennbar ist, wie oft ein Objekt einer Klasse in die Assoziation eingeht. Auch das Konzept der (min, max)-Notation existiert in UML-Klassendiagrammen. Möchte man ausdrücken, dass ein Objekt einmal oder höchstens einmal in eine Assoziation eingehen kann, so wählt man die Kardinalität 0..1. Ein Stern (*) hingegen bedeutet, dass ein Objekt einmal oder unbegrenzt oft eine Assoziation eingehen kann. Demgegenüber bedeutet die Kardinalität 1..*, dass ein Objekt mindestens einmal in eine Assoziation eingehen muss.

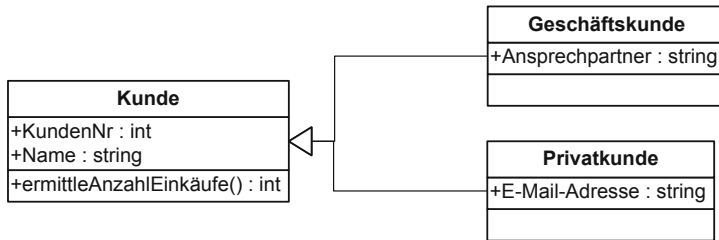


Abbildung 8: Generalisierung/Spezialisierung

Vererbungsmechanismen können in UML-Klassendiagrammen mit Hilfe des auch schon aus ER-Modellen bekannten Konstrukts der Generalisierung/Spezialisierung abgebildet werden (Abbildung 8). Eine Subklasse erbt dabei alle als *public* deklarierten Attribute und Operationen der Superklasse. Sie kann darüber hinaus eigene Attribute und Operationen definieren, die sie von ihrer Superklasse unterscheidet.

Können bestimmte Sachverhalte nicht in grafischer Notation ausgedrückt werden, so verfügt das UML-Klassendiagramm über das Konstrukt des Kommentars. Ein Kommentar kann frei in einem Modell erscheinen oder aber an eine konkrete Klasse annotiert sein. Oftmals beinhalten Kommentare in der Object Constraint Language (OCL) definierte formale Ausdrücke, die beispielsweise den Wertebereich eines Attributes genau spezifizieren.

Element	Erläuterung	Symbol
Klasse	Die Klasse stellt das zentrale Konstrukt zur Beschreibung realweltlicher Objekte in UML-Klassendiagrammen dar. Klassen besitzen Attribute, die Eigenschaften der Klasse repräsentieren, und Operationen, die ihr Verhalten festlegen.	<pre> classDiagram class Artikel { +ArtikelNr : int +Bezeichnung : string +Maße : string +Gewicht : float +Farbe : string +ermittleDurchschnittsEK-Preis() : float } </pre>
Unäre Assoziation	Eine unäre Assoziation beschreibt eine einseitig gerichtete Beziehung zweier Klassen. Nur die Zielklasse erhält ein Attribut, das die Assoziation beschreibt.	<pre> classDiagram class Artikel { +Artikel } </pre>

Element	Erläuterung	Symbol
Binäre Assoziation	Eine binäre Assoziation beschreibt eine beidseitig gerichtete Beziehung zweier Klassen. Beide Klassen erhalten ein Attribut. Assoziationen können mit Kardinalitäten versehen werden und Eigenschaften aufweisen.	
N-äre Assoziation	Eine n-äre Assoziation beschreibt eine Beziehung zwischen mehr als zwei Klassen.	
Assoziationsklasse	Eine Assoziationsklasse stellt eine Klasse dar, die Attribute und Operationen einer Assoziation definieren kann.	
Komposition	Eine Komposition stellt eine starke „besteht aus“-Beziehung dar. Beide Klassen können nicht ohne die jeweils andere Klasse existieren.	
Aggregation	Eine Aggregation stellt eine schwache „besteht aus“-Beziehung dar. Jede Klasse kann auch ohne die jeweils andere Klasse existieren.	
Generalisierung/Spezialisierung	Eine Generalisierung bzw. Spezialisierung bildet Vererbungsmechanismen ab. Je nach Sichtbarkeitstyp werden Attribute und Operationen an Subklassen vererbt.	
Kommentar	Ein Kommentar erlaubt die Annotation von syntaktischen oder semantischen Eigenschaften bestimmter Modellelemente, die nicht über eine grafische Notation abgebildet werden können.	

Tabelle 2: Elemente des UML-Klassendiagramms

2.2 Modellierungssprachen für Prozesse

2.2.1 Ereignisgesteuerte Prozessketten (EPK)

Die Ereignisgesteuerte Prozesskette (EPK) wurde erstmalig von Keller, Nüttgens und Scheer im Jahr 1992 vorgestellt und wird vor allem im Bereich des industriellen Geschäftsprozessmanagements eingesetzt.⁹ Die EPK ist eine der Hauptkomponenten der Architektur Integrierter Informationssysteme (ARIS) und beschreibt dort die Steuerungs- bzw. Prozesssicht.¹⁰ Die langjährige Verfügbarkeit und die Unterstützung durch das ARIS-Modellierungswerkzeug haben dazu beigetragen, dass die EPK zu den verbreitetsten Prozessmodellierungstechniken in der Praxis zu rechnen ist.

Bei Prozessmodellen, die mit der Modellierungstechnik EPK erstellt worden sind, handelt es sich um gerichtete Grafen, die im Kern die Elemente Funktion, Ereignis, Kontrollfluss (gerichtete Kante) und Konnektor beinhalten. In Tabelle 3 werden diese Elemente einzeln erläutert. Mit dem Element der Funktion werden Tätigkeiten bzw. Aktivitäten in einem Prozess dargestellt, daher besitzt es einen aktiven Charakter. Ereignisse hingegen beschreiben Zustände in einem Prozess und haben daher passiven Charakter. Ein Prozessmodell in Form der EPK beschreibt, welche Ereignisse welche Funktionen auslösen und welche Ereignisse von welchen Funktionen erzeugt werden. Ein EPK-Prozessmodell ist somit eine wechselnde Folge von Ereignissen und Funktionen. Die Verbindung zwischen Ereignissen und Funktionen wird mit dem Kontrollfluss vorgenommen.

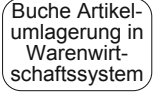
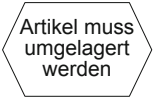

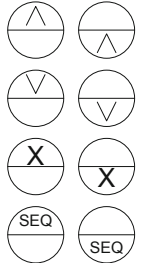

Durch den passiven Charakter der Ereignisse besitzen Ereignisse keine Entscheidungskompetenz. Sie stellen Zustände dar, auf die mit bestimmten Funktionen reagiert werden kann, sind aber nicht verantwortlich für weitere Zustandswechsel eines Prozesses. Funktionen führen die Transformation der Inputs zu den Outputs durch und haben auch Entscheidungskompetenz, d. h., nach einer Funktion kann entschieden werden, welches Ereignis als Resultat eintritt und wie sich dies auf den Zustand des Prozesses auswirkt.

Einfache lineare Prozesse können mithilfe von Funktionen, Ereignissen und dem Kontrollfluss dargestellt werden. Hierbei besitzt jede Funktion genau ein Vorgänger- und ein Nachfolgeereignis und diese sind jeweils mit einer gerichteten Kante verbunden. Wenn eine lineare Repräsentation eines Prozesses nicht hinreichend ist, so kommen die Konnektoren der EPK zum Einsatz. Diese können sowohl parallele Teilabläufe aufspalten und diese wieder zusammenführen als auch alterna-

⁹ Vgl. Keller et al. (1992). Zur Anwendung im Bereich des Geschäftsprozessmanagements vgl. z. B. Rump (1999) und Becker et al. (2008).

¹⁰ Zu ARIS vgl. z. B. Scheer (1997), Scheer (1998) und Scheer (2001).

tive Teilabläufe darstellen. Bei der Verwendung dieser logischen Konnektoren ist eine Reihe von Regeln zu beachten, welche die formale Richtigkeit des Prozessmodells sicherstellen sollen.¹¹

Element	Erläuterung	Symbol/Beispiel
Funktion	Eine Funktion steht für eine Aktivität bzw. Tätigkeit, welche einen Input in einen Output transformiert. Mit Funktionen werden die Bearbeitungsschritte eines Prozesses abgebildet. Funktionen werden aufgrund von eintretenden Ereignissen durchgeführt und lösen neue Ereignisse aus.	
Ereignis	Ein Ereignis beschreibt einen eingetretenen Systemzustand (Bereitstellungscharakter), der zur Folge eine oder mehrere Aktivitäten (Funktionen) haben kann (Auslösecharakter). Ein Ereignis stellt das passive Element einer EPK dar und hat ein Zeitintervall von null. Jeder Prozess beginnt und endet mit mindestens einem Ereignis.	
Kontrollfluss	Der Kontrollfluss setzt die Funktionen und Ereignisse miteinander in Verbindung. Hierdurch können sachlogische und zeitliche Abhängigkeiten abgebildet werden. Durch Verbindung mit den logischen Konnektoren können komplexere, nicht-lineare Prozesse dargestellt werden.	
Konnektoren	Konnektoren sind logische Operatoren, die dazu dienen Prozessverzweigungen modellieren zu können. Es werden vier Prozessverzweigungsoperatoren unterschieden: das logische UND, das INKLUSIVE ODER, das EXKLUSIVE ODER (XOR) und die wahlfreie SEQUENZ. Diese lassen sich jeweils in Eingangs- und Ausgansoperatoren unterscheiden. Der logische AND-Operator verknüpft mehrere parallel ablaufende Prozessstränge, während der OR- und XOR-Operator mehrere alternativ ablaufende Prozessstränge separiert. Der SEQUENCE-Operator hingegen separiert mehrere alternative sequentielle Abläufe desselben Prozessstrangs.	
Organisatorische Rolle	Eine organisatorische Rolle beschreibt einen Tätigkeitsbereich, der von einer von mehreren Personen, die diesem Bereich zugeordnet sind, ausgeführt wird.	

¹¹ Die Regelmenge wird an dieser Stelle aufgrund ihrer Komplexität nicht dargestellt, kann aber in der angegebenen Literatur zur EPK nachgeschlagen werden.


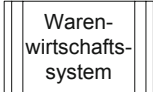
Organisatorische Stelle	Eine organisatorische Stelle beschreibt einen Tätigkeitsbereich der von einer konkreten Person ausgeführt wird.	
Anwendungssystem	Ein Anwendungssystem ist ein für Softwaresystem, das für eine bestimmte Funktion oder Anwendungsdomäne erstellt wurde.	

Tabelle 3: Grundlegende Elemente der eEPK-Modellierungssprache

Im Prozessmodell in der Abbildung 9 befinden sich noch weitere Elemente, die nicht zum Kern der EPK gehören. So werden Verantwortlichkeiten oder Durchführungen von Funktionen durch die Verbindung mit der Organisationssicht dargestellt. Die abgebildeten Organisationseinheiten und Stellen sind demnach Elemente, die aus der Organisationssicht stammen und in die EPK integriert werden. Daneben werden noch eingehende und ausgehende Dokumente und Informationen einer Funktion dargestellt. Hiermit werden die bekannten Inputs und Outputs eines Prozessmoduls abgebildet. Ein Prozessmodul ist ein „sinnvoll und logisch eindeutig abgegrenzter Funktionsbereich eines Geschäftsprozesses“¹² Zusätzlich kann eine Reihe von weiteren Ressourcen, die zur Funktionsdurchführung benötigt werden, dargestellt werden. Dies sind z. B. verwendete Anwendungssysteme. Eine EPK, in deren Rahmen die genannten Elemente verwendet werden, wird auch als erweiterte EPK (eEPK) bezeichnet (vgl. Tabelle 3).

Mit den dargestellten Elementen kann ein Prozess auf einer Prozessebene, also einem festen Abstraktionsgrad, dargestellt werden. Da die Funktionen der EPK mit den Prozessmodulen korrespondieren, findet die Verfeinerung direkt auf dem Element der Funktion statt. Hierfür bietet die EPK die Funktionsverfeinerung an. Wenn eine Funktion verfeinert dargestellt werden soll, so wird an das Funktionsymbol ein kleines Verfeinerungssymbol hinzugefügt. Hinter eine solche Funktion kann nun ein weiteres eigenständiges Prozessmodell gelegt werden, das die reine textuelle Bezeichnung der Funktion als detaillierten Ablauf darstellt. Um die Konsistenz zwischen einer Funktion und ihrer Verfeinerung sicherzustellen, müssen die Vorgänger- und Nachfolgeereignisse von Funktion und ihrer Verfeinerung übereinstimmen. In Abbildung 10 wird dies beispielhaft dargestellt.

Neben der vertikalen Auflösung können vollständige Prozessmodelle miteinander über Schnittstellen verknüpft werden. Dies entspricht der horizontalen Verkettung von Prozessmodulen. Hierfür wird das Element der Prozessschnittstelle bereitgestellt. Auch hier müssen die Endereignisse des Vorgängerprozesses als Anfangsereignisse im Nachfolgeprozess eingeblendet werden, damit die formale Konsistenz sichergestellt werden kann.

¹² Klein (2007), S. 177.

In Abbildung 9 wird die Anwendung der logischen Konnektoren anhand eines Verwaltungsprozesses beispielhaft illustriert.

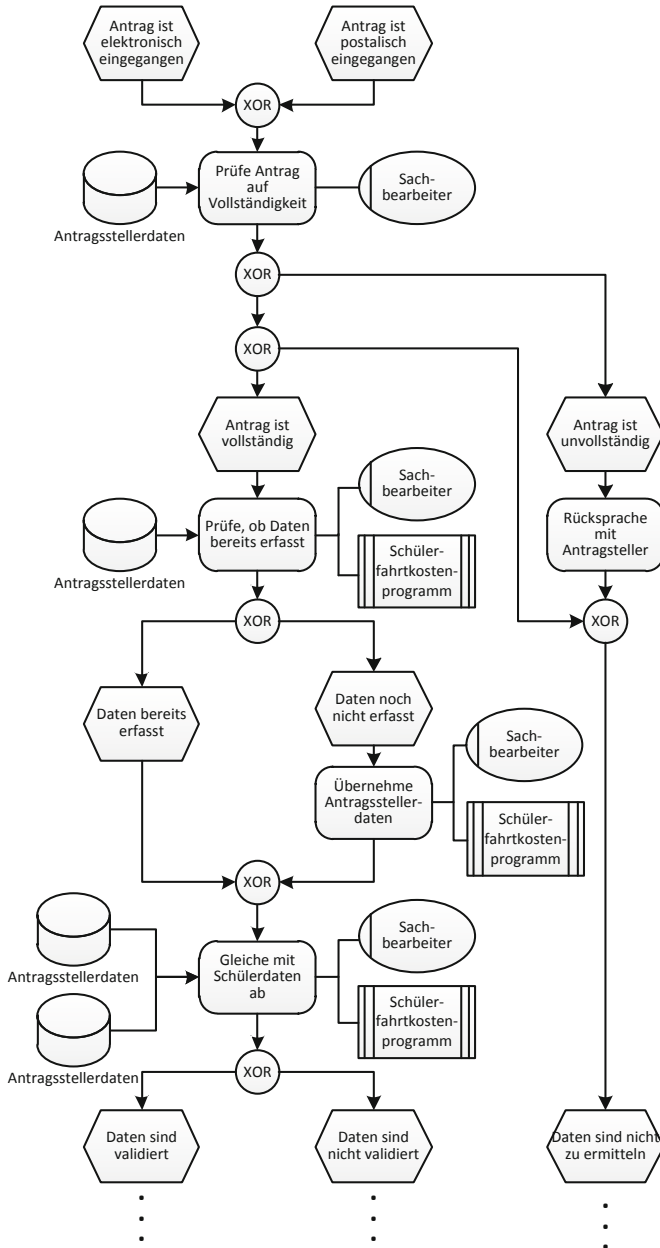


Abbildung 9: EPK-Modellauszug aus dem Prozess der Schülerfahrtkostenerstattung

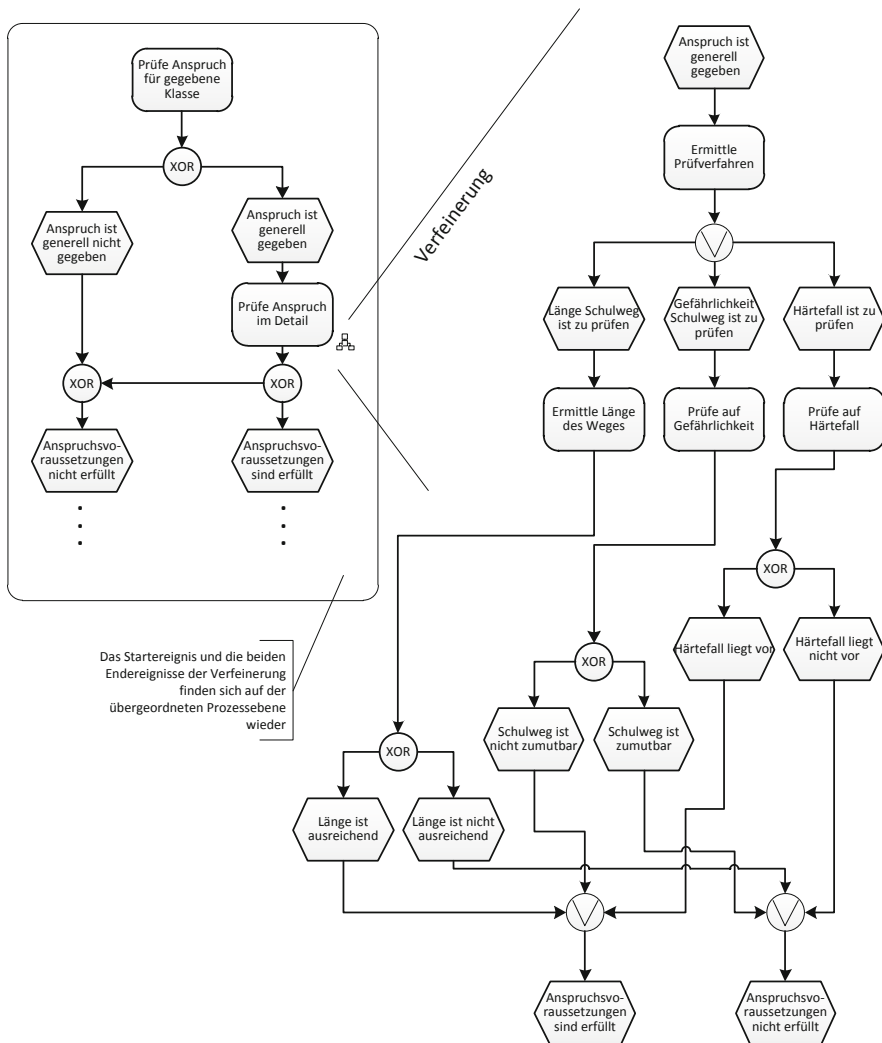


Abbildung 10: Beispiel einer Funktionsverfeinerung in EPK-Notation

Insgesamt kann die Modellierungstechnik EPK als sehr flexibles und universelles Instrument eingestuft werden. Die Kernelemente „Funktion“ und „Ereignis“ sind so abstrakt gewählt, dass bei der Darstellung von Abläufen kaum Restriktionen gesetzt werden. Die EPK ist somit weder auf eine konkrete Domäne noch einen speziellen Problembereich begrenzt.

Bezüglich der Darstellungstiefe bzw. -genauigkeit lässt sich festhalten, dass die Tätigkeiten bzw. Aktivitäten und die auslösenden und eintretenden Ereignisse auf einem beliebigen Abstraktionsniveau dargestellt werden können, da hier keinerlei Vorschriften existieren. So wird zwar empfohlen, dass sich die Bezeichnungen

von Funktionen, die auf einer Prozessebene dargestellt werden, möglichst auf einem Abstraktionsniveau befinden sollten, die Realisierung dieser Empfehlung obliegt allerdings vollständig dem Modellierer, und die EPK bietet hierfür keine expliziten Elemente oder Hilfestellungen an.

Die logischen Konnektoren sind ebenfalls sehr flexibel bezüglich der Abbildungsmöglichkeiten zur Darstellung komplexer, nicht-linearer Abläufen. Durch geschickte Kombination der Konnektoren lassen sich auch sehr verzweigte Prozessabläufe darstellen. Diese Flexibilität erfordert allerdings auch ein tief gehendes methodisches Verständnis, da sowohl das formale Regelwerk zur korrekten Anwendung der Konnektoren als auch die Interpretation der resultierenden Prozessmodelle nicht trivial ist.

Die EPK integriert weitere Sichten in die dargestellte Prozesssicht. Dazu gibt die Architektur Integrierter Informationssysteme (ARIS) den Integrationsrahmen vor. Der Fokus liegt dabei vor allem auf der Integration der Organisations-, Daten- und Funktionssicht, so dass hier klare Schnittstellen vorliegen und die einzelnen Sichten für sich genommen gut abgegrenzt sind. Für die vorgestellten Elemente zur Darstellung von Eingangs- und Ausgangsdokumenten und weiteren Prozessressourcen, wie z. B. Anwendungssystemen, existieren allerdings keine gesonderten Sichten, wie z. B. eine „Ressourcensicht“ oder „Geschäftsobjektsicht“. Dies hat zur Folge, dass die hier angebotenen Elemente nicht im Sinne des Sichtenkonzeptes über Schnittstellen wiederverwendet werden können.

2.2.2 Petri-Netze

Petri-Netze, auch bekannt als Stelle/Transitions-Netze bzw. S/T-Netze, wurden erstmals von Carl Adam Petri im Rahmen seiner Dissertation im Jahr 1962 vorgestellt.¹³ Sie sind gerichtete, zustandsbasierte Graphen und eignen sich zur Modellierung von Systemen und Transformationsprozessen. Bis zum Beginn der 1980er Jahre ließ sich, in Relation zur heutigen Bedeutung, eine eher mäßige Beachtung der Petri-Netze verzeichnen. Ab diesem Zeitpunkt stieg die Anzahl an praktischen und theoretischen Arbeiten im Bezug auf Petri-Netze stark an. Ergebnis waren zahlreiche konzeptionelle Erweiterungen und gänzlich neue, auf den Grundlagen von Petri-Netzen basierende Konzepte für die unterschiedlichsten Anwendungsgebiete. Diese Entwicklung wurde weiterhin durch die höhere Verfügbarkeit von Softwaretools (Editoren und Simulationstools) begünstigt. Das Einsatzgebiet von Petri-Netzen reicht von der Steuerungstechnik über die Softwareentwicklung bis hin zur Prozessmodellierung oder der Modellierung von Workflowmanagement-Prozessen. Ihr Vorteil, im Vergleich zu anderen Prozessmodellierungssprachen,

¹³ Vgl. Petri (1962).

liegt insbesondere in ihrer sehr guten mathematischen Fundierung und den daraus resultierenden Analysemethoden.¹⁴

Im Wesentlichen besteht ein Petri-Netz aus Stellen und Transitionen (Übergängen), welche durch gerichtete Kanten miteinander verbunden sind. Dabei dürfen niemals zwei Stellen oder zwei Transitionen direkt miteinander verbunden sein. In Tabelle 4 sind die Elemente der Modellierungssprache für Petri-Netze dargestellt und beschrieben. Stellen werden durch Kreise dargestellt. Jede Stelle besitzt eine bestimmte Kapazität, welche an die jeweilige Stelle annotiert wird. Stellen, bei denen die Kapazität nicht explizit angegeben ist, haben in der Regel eine Kapazität von eins oder unendlich. Der Zustand einer Stelle wird durch sogenannte Marken dargestellt, welche in Form von schwarzen Punkten innerhalb des Kreises einer Stelle modelliert werden. Eine Stelle kann dabei keine, eine oder mehrere Marken beinhalten. Die Kapazität einer Stelle gibt dabei die maximale Anzahl an Marken an, die eine Stelle aufnehmen kann. Die Stellen sind mit den Transitionen über gewichtete Kanten verbunden. Die Gewichte der gerichteten Kanten sind in der Regel an diese annotiert. Wenn keine expliziten Gewichte angegeben sind, dann hat das Gewicht den Wert eins.

Transitionen werden durch beschriftete Rechtecke dargestellt und bilden die Zustandsübergänge ab. Eine Transition wird immer dann als aktiv oder schaltbereit bezeichnet, wenn alle Stellen, von denen aus eine Kante in diese Transition führt, mindestens so viele Marken besitzen, wie die Kantengewichte es erfordern, und alle der Transition nachgelagerten Stellen den Kantengewichten entsprechend freie Kapazitäten aufweisen können. Wenn eine Transition schaltbereit ist, dann *kann* sie schalten. Sind mehrere Transition gleichzeitig schaltbereit, dann ist die Reihenfolge, in der sie schalten, nicht determiniert. Aus diesem Grund wird ein Petri-Netz auch als ein Modell von nebenläufigen Systemen angesehen. Der Begriff Nebenläufigkeit bezeichnet hierbei nicht die „Gleichzeitigkeit“, sondern vielmehr die „Unabhängigkeit“ der einzelnen Netz-Elemente. Schaltet eine Transition, dann „verbraucht“ sie Marken der vorgelagerten Stellen und „erzeugt“ Marken für die nachgelagerten Stellen. Marken werden also durch Transitionen nicht „bewegt“, sondern verbraucht und erzeugt, was dazu führt, dass sich in der Regel die Gesamtzahl der Marken in einem Petri-Netz ändern kann. Wenn alle Kantengewichte und Stellenkapazitäten den Wert eins besitzen, dann spricht man auch von einem Bedingungs-, Prädikat- oder Ereignis-Netz.

¹⁴ Vgl. Langner et al. (1997), S. 480.




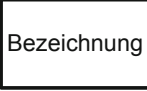

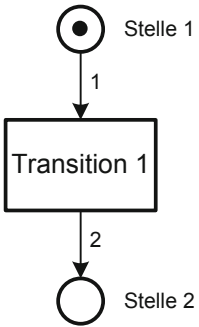
Element	Erläuterung	Symbol
Marke	Eine Marke (engl. Token) dient zur Beschreibung des Zustands einer Stelle.	•
Stelle	Eine Stelle (engl. Place) stellt einen Zustand in dem modellierten System dar. Sie besitzt entweder keine Marke, eine Marke oder N Marken. Eine Kapazität von >1 wird explizit an die Stelle annotiert.	 Bezeichnung  Bezeichnung Kapazität N  Bezeichnung
Transition	Eine Transition stellt den Übergang zwischen zwei Zuständen dar. Ein solcher Übergang kann beispielsweise durch ein Ereignis oder eine Aktivität ausgelöst werden. Die Beschriftung einer Transition kann sich entweder innerhalb oder außerhalb des Modellelementes befinden.	 Bezeichnung
Kante	Die Kanten in Petri-Netzen sind gerichtet und gewichtet. Wenn kein explizites Gewicht angegeben wird, dann ist das Gewicht eins.	 Gewicht
Kontrollfluss	Der Kontrollfluss setzt die Stellen und die Transitionen in sachlogischer Reihenfolge miteinander in Verbindung. Die Kanten, die von einer Stelle ausgehen und in eine Transition eingehen, stellen den Input der Transition dar. Die aus der Transition ausgehenden Kanten stellen deren Output dar. Die Größe des jeweiligen In- und Outputs wird in Form von Gewichten an die Kanten annotiert. Eine Transition ist aktiviert bzw. schaltbereit, wenn alle Inputbedingungen erfüllt und genügend Outputkapazitäten frei sind.	

Tabelle 4: Elemente von Petri-Netzen

Im Folgenden soll anhand eines Beispiels die Funktionsweise eines Petri-Netzes verdeutlicht werden (vgl. Abbildung 11). Ein solches Petri-Netz könnte beispielsweise zur Beschreibung eines bestimmten Programmablaufs einer Software oder zur Beschreibung eines bestimmten Prozesses in einem Unternehmen dienen. Je nachdem wären die Marken dann eventuell Datenobjekte oder Fertigungsmaterial. Die Stellen wären dann Datenspeicher bzw. Zwischenlager und die Transitionen datenverarbeitende Programmfunktionen oder Material-Bearbeitungsschritte.

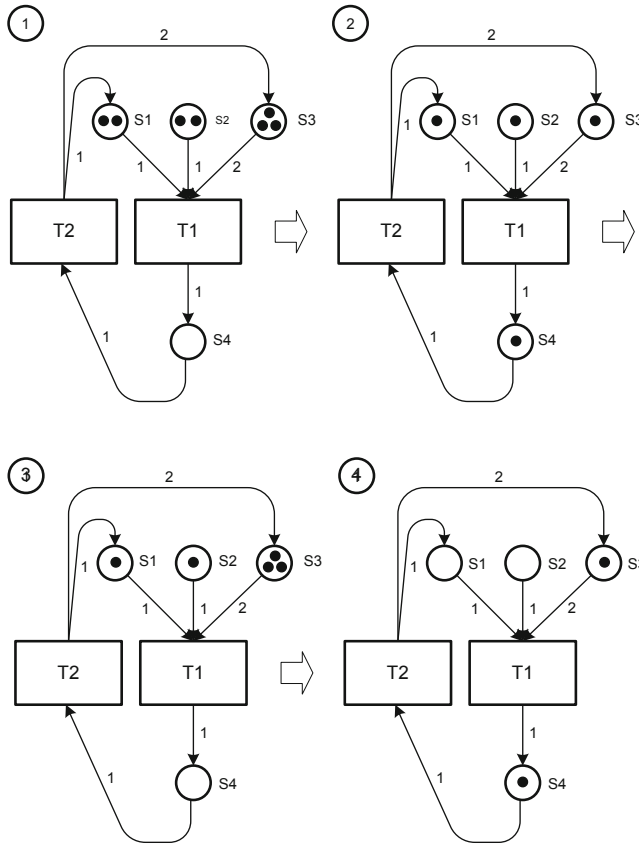


Abbildung 11: Beispiel für ein Petri-Netz

Das in Abbildung 11 zu sehende Petri-Netz besteht aus vier Stellen S1 - S4 und zwei Transitionen T1 und T2. Zunächst befindet sich das Netz in einem Zustand, in dem die Transition T1 aktiv bzw. schaltbar ist (Modell 1). Die drei vorgelagerten Stellen S1 - S3 erfüllen jeweils die Anforderungen für T1, welche durch die Gewichte der Kanten dargestellt sind. Außerdem besitzt die nachgelagerte Stelle S4 freie Kapazitäten. Im nächsten Schritt wird T1 schalten. Den Stellen S1 - S3 werden entsprechend den Kantengewichten Marken entnommen. Dafür erhält die Stelle S4 eine Marke dazu. In diesem neuen Zustand kann T1 nun nicht mehr schalten, da die Stelle S3 lediglich eine Marke besitzt (Modell 2). Dafür kann nun jedoch die Transition T2 schalten, denn S4 enthält die dafür erforderliche Marke, und die nachgelagerten Stellen S1 und S3 haben ausreichend freie Kapazitäten. (In diesem Beispiel wurden die Kapazitäten aus Gründen der Übersichtlichkeit nicht explizit an die Stellen annotiert). Als nächstes schaltet die Transition T2. Die Marke aus S4 wird entnommen, und S1 und S3 erhalten entsprechend Marken dazu. Jetzt kann T1 wieder schalten, da die erforderlichen Marken in den vorgelager-

ten Stellen verfügbar sind und die Kapazität in S4 wieder frei ist (Modell 3). Nach einem erneuten Schalten von T1 befindet sich das Netz in dem in Modell 4 dargestellten Zustand.

An Modell 4 lässt sich bereits erkennen, dass dieses Petri-Netz nicht unendlich oft schalten kann, denn S2 besitzt bereits keine Marken mehr und erhält auch keine neuen dazu. Ein solches Netz nennt man auch todesgefährdet, da es nach einer weiteren Schaltung tot sein wird. Lebendigkeit und Sicherheit sind zwei wesentliche Eigenschaften von Petri-Netzen. Sicherheit bedeutet in diesem Zusammenhang grob gesagt, dass nichts Verbotenes im System geschieht, und Lebendigkeit, dass das System nicht stecken bleibt.¹⁵ In Bezug auf die Lebendigkeit lassen sich folgende Eigenschaften identifizieren:

Ein Petri Netz ist...

- ...*tot*, wenn alle Transitionen tot sind, d. h. wenn keine Transition mehr schalten kann.
- ...*todesgefährdet*, wenn das Netz nach dem Schalten einer bestimmten Transition tot ist.
- ...*schwach lebendig*, wenn das Netz nach keinem direkt folgenden Schaltvorgang tot ist.
- ...*(stark) lebendig*, wenn alle Transitionen lebendig sind, d. h. wenn alle Transitionen nach jedem Schaltvorgang wieder schaltbar sind.

Ein Petri-Netz heißt sicher (oder auch b-sicher), wenn zu jeder Zeit in jeder Stelle maximal b Marken liegen. Im Gegensatz zur Kapazität einer Stelle, die eine Begrenzung bereits im Vorhinein festlegt, stellt die Sicherheit eines Petri-Netzes eine Begrenzung dar, die im Nachhinein durch Beobachtung festgestellt wird.¹⁶ Petri-Netze können, wie in Abbildung 11 zu sehen, Schleifen beinhalten oder aber einen festgelegten Start- und Endpunkt haben.

In der Praxis werden Petri-Netze oft sehr groß und komplex und dadurch schwer lesbar. Außerdem wird der Modellierungsvorgang kompliziert und nimmt daher viel Zeit in Anspruch. Aus diesem Grund sind sogenannte *high-level*-Petri-Netze entstanden, welche auf unterschiedliche Weise anhand von Erweiterungen versuchen, diesen Nachteilen entgegenzuwirken.

Mit farbbasierten Petri-Netzen versucht man, durch unterschiedliche Färbung der Marken die Möglichkeit zur Unterscheidung zu schaffen. Auf diese Weise lassen sich den einzelnen Marken spezifische Eigenschaften zuordnen, die in den Schaltvorgang einer Transition einfließen können.

¹⁵ Vgl. Baumgarten (1990), S. 130.

¹⁶ Vgl. Baumgarten (1990), S. 131.

Damit sich auch die zeitliche Abfolge in einem Petri-Netz darstellen lässt, können Petri-Netze um Zeitangaben erweitert werden, die den einzelnen Marken, Stellen und Transitionen zugeordnet werden können. So kann man beispielsweise festlegen, dass immer diejenigen Marken zuerst verarbeitet werden sollen, die schon am längsten in einer Stelle liegen.

Eine weitere Erweiterung der klassischen Petri-Netze ist die Einbeziehung von Modell-Hierarchien in den Modellierungsprozess. Auf diese Weise lassen sich Petri-Netze auf verschiedenen Komplexitätsstufen modellieren. So können komplexe Modellstrukturen integriert abgebildet werden.

2.2.3 Business Process Modeling Notation (BPMN)

Die Business Process Modeling Notation (BPMN) wurde von der Business Process Management Initiative (BPMI) entwickelt und im Mai 2004 in der Version 1.0 veröffentlicht. Heute (2012) liegt BPMN in der Version 2.0 vor. Mit der Entwicklung dieser Modellierungstechnik wurden folgende Zielsetzungen verfolgt:¹⁷

Die Modellierungstechnik sollte für die verschiedenen Akteure im Rahmen der Prozessmodellierung gleichermaßen verständlich und einfach zu lesen sein. Zu diesen Akteuren gehören nach Auffassung der BPMI Systemanalytiker, welche die fachliche Modellierung vornehmen, Technologieentwickler, welche die erstellten Modelle verwenden und ggf. verfeinern, und die Mitarbeiter der untersuchten Unternehmung, welche die dahinterliegenden Prozesse gestalten, ausführen und überwachen.

Neben der Verständlichkeit der Modelle sollten diese vor allem für technische Fragestellungen wiederverwendet werden können. So wurde das allgemeine Problem aufgegriffen, dass Prozessmodelle, die den fachlichen Teil eines Prozesses wiedergeben, häufig nicht für technische Anwendungszwecke, wie z. B. die Entwicklung und Konfiguration von Anwendungssystemen, direkt herangezogen wurden bzw. herangezogen werden konnten. Mit der Entwicklung und Bereitstellung von verteilten serviceorientierten Architekturen und deren Realisierungsmöglichkeit durch Web-Services wurde die Forderung aufgegriffen, dass Prozessmodelle möglichst für den Aufbau und die Konfiguration solcher Systeme genutzt werden können. Die Vision dahinter ist, dass fachliche, leicht verständliche Prozessmodelle direkt genutzt werden können, um in diese Systeme „gefüttert“ werden zu können. Damit soll praktisch „auf Knopfdruck“ ein lauffähiges Informationssystem konfiguriert werden können, welches die Prozessausführung steuert und überwacht. Aus diesen Sichtweisen lässt sich der eher technische Fokus der Entwicklungsbestrebungen erkennen. Die umfassende Demonstration dieses Ansatzes steht allerdings sowohl in Industrie als auch Verwaltung noch immer aus.

¹⁷ Vgl. im Folgenden White (2004).

Im Kern besteht die BPMN aus Geschäftsprozessdiagrammen, die jeweils einen oder mehrere Gesamtprozesse abbilden und die verschiedenen Modellierungselemente beinhalten. Diese Elemente sind in vier unterschiedliche Klassen unterteilt, die jeweils einen spezifischen Teil der Prozessinformationen abbilden können. Es sind die Klassen der Ablaufelemente, der Verbindungselemente, der Schwimmbahnen und der Artefakte.

In Abbildung 12 ist ein Ausschnitt aus einem Verwaltungsprozess mit der BPMN abgebildet worden. Die Erläuterungen der Elemente befinden sich in den nachfolgenden Tabellen.

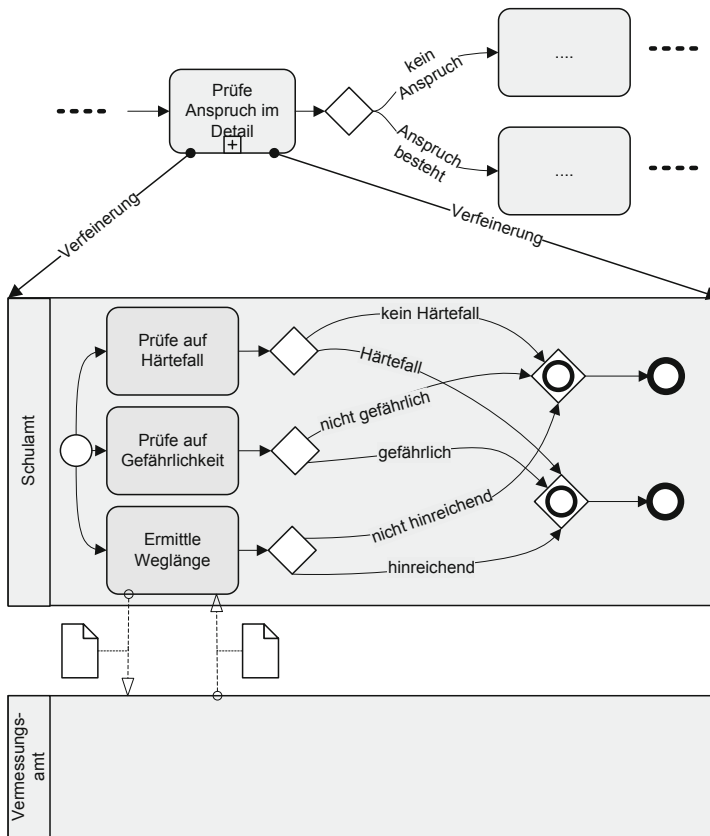


Abbildung 12: Ausschnitt der Schülerfahrtkostenerstattung in BPMN

In der Klasse der Ablaufelemente (vgl. Tabelle 5) befinden sich die Elemente, die die Aktivitäten und den zeitlichen und sachlogischen Ablauf der Aktivitäten darstellen.

Die Ablaufelemente der BPMN ähneln den Elementen der EPK. So werden hier ebenfalls Aktivitäten und Ereignisse dargestellt, und Aktivitäten können verfeinert werden. Auf diese Weise sind auch in der BPMN vertikale Auflösungen von Prozessmodulen möglich. Der Entscheidungspunkt übernimmt die Funktion aller logischen Konnektoren der EPK.

Die nachfolgende Klasse der Verbindungselemente ermöglicht es, verschiedene Elemente miteinander zu verbinden. Es werden verschiedene Verbinder bereitgestellt, je nachdem, welche Elemente miteinander verbunden werden sollen. Tabelle 6 fasst diese zusammen.





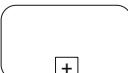



Element	Erläuterung	Symbol
Ereignis	Durch Ereignisse werden Zustandsänderungen repräsentiert. Sie beeinflussen den Ablauf von Prozessen. Es wird zwischen Start-, Zwischen- und Endereignissen eines Prozesses unterschieden.	 Startereignis  Zwischenereignis  Endereignis
Aktivität	Eine Aktivität repräsentiert das eigentliche Handeln. Es wird zwischen atomaren und zusammengesetzten Aktivitäten unterschieden. Zusammengesetzte Aktivitäten werden auch als Unterprozesse bezeichnet und mit einem „+“ gekennzeichnet.	 Aktivität  Unterprozess
Entscheidungspunkt	Mit Entscheidungspunkten kann der Ablauf in Prozessen aufgespalten und zusammengeführt werden, um z. B. Parallelität abbilden zu können. Weiterhin können Entscheidungen dargestellt werden, welche in unterschiedliche Prozessalternativen resultieren.	 XOR  UND  ODER

Tabelle 5: Ablaufelemente der Business Process Modeling Notation

Die Sequenzverbindung übernimmt eine zentrale Rolle, da hierdurch die eigentlichen Abläufe und damit die Reihenfolgebeziehungen abgebildet werden. Der Nachrichtenfluss dient der Abbildung von Kommunikationsbeziehungen zwischen verschiedenen Prozessbeteiligten. Dabei ist festzuhalten, dass der Nachrichtenfluss nur für die Koordination von entkoppelten Prozessen verwendet wird. Entkoppelte Prozesse zeichnen sich dadurch aus, dass sie nur wenige Interaktionspunkte besitzen, inhaltlich somit größtenteils unabhängig ablaufen. Dies ist häufig bei örtlich oder organisatorisch getrennten Prozessen der Fall. Aus diesem Grund

werden in der BPMN eben nur Nachrichten zwischen „Pools“ ausgetauscht, also Prozessen, die von verschiedenen, unabhängigen Prozessbeteiligten durchgeführt werden.


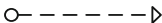

Element	Erläuterung	Symbol
Sequenzverbindung	Durch die Sequenzverbindung wird die Abfolge von Ablaufelementen dargestellt, indem diese mit dem Verbinder verknüpft werden.	
Nachrichtenfluss	Durch den Nachrichtenfluss wird der Austausch von Nachrichten zwischen unterschiedlichen Prozessbeteiligten dargestellt. Die unterschiedlichen Akteure werden durch verschiedene „Pools“ dargestellt (vgl. Pool Element).	
Verbindung	Mithilfe dieser allgemeinen Verbindung können z. B. Inputs und Outputs von Aktivitäten dargestellt werden. Allgemein können hiermit beliebige Artefakte (vgl. Artefakte) an Ablaufelemente an-notiert werden.	

Tabelle 6: Verbindungselemente der Business Process Modeling Notation

Die in Tabelle 7 dargestellten Schwimmbahnelemente dienen eben dieser Klassifikation von inhaltlich abhängigen und unabhängigen Prozessbereichen. Die Aktivitäten eines Prozessbeteiligten werden innerhalb eines Pools dargestellt und dort mit Sequenzverbindern verbunden. Ein Pool kann dabei noch in Bahnen unterteilt werden, wenn es noch weitere inhaltliche und sinnvolle Unterteilungen gibt.


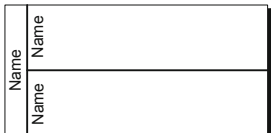
Element	Erläuterung	Symbol
Pool	Mithilfe von Pools werden Prozessbeteiligte visualisiert. Alle Aktivitäten eines Beteiligten werden in dem entsprechenden Pool dargestellt und können als eigenständiger Prozess aufgefasst werden. Prozessbeteiligte ergeben sich häufig aus den Organisationen oder Rollen, die an einem Gesamtprozess beteiligt sind.	
Bahn	Bahnen unterteilen Pools, um so Untergliederungen von Aktivitäten eines Prozessbeteiligten abbilden zu können. Bahnen werden z. B. genutzt, um interne Organisationseinheiten darzustellen.	

Tabelle 7: Schwimmbahnelemente der BPMN

Zwischen Pools, also zwischen Prozessbeteiligten, kann die Koordination nur mit dem Nachrichtenfluss vorgenommen werden. Innerhalb eines Pools, also z. B. auch zwischen Bahnen eines Pools, dürfen keine Nachrichtenflüsse dargestellt werden, sondern es findet eine direkte Koordination über die Benutzung der Sequenzverbinder statt.

Die Klasse der Artefaktelemente beinhaltet die Elemente, die zusätzlich für die Durchführung der Aktivitäten benötigt werden oder sonstige Modellinformationen darstellen können. In Tabelle 8 sind diese zusammengefasst.

Zentrales Element ist hier das Datenobjekt. Dieses wird für die Abbildung von Dokument- oder allgemein Informationsaustauschen verwendet. Die notwendigen Inputs und die resultierenden Outputs der Aktivitäten können hiermit abgebildet werden.

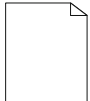


Element	Erläuterung	Symbol
Datenobjekt	Datenobjekte werden genutzt, um Inputs und Outputs von Aktivitäten darzustellen. Sie werden mithilfe der „Verbindung“ an die Aktivität geknüpft.	 Name [Zustand]
Gruppierung	Die Gruppierung kann beliebige Modellelemente zusammenfassen. Sie dient somit reinen Dokumentations- oder Analysezwecken und hat keine Auswirkung auf den Prozessablauf.	
Anmerkung	Mithilfe von Anmerkungen kann freier Text an beliebige Modellierungselemente annotiert werden, damit das Modell besser dokumentiert werden kann.	 Textuelle Anmerkung für weitere Informationen

Tabelle 8: Artefaktelemente der Business Process Modeling Notation

Die Artefaktklasse ist ebenfalls dafür vorgesehen, dass die Anwender weitere für sie wichtige Elemente der Modellierungstechnik hinzufügen können, wie z. B. physische Produkte oder weitere Ressourcen, wie z. B. Informationssysteme oder sonstige Arbeitsmaterialien. Weiterhin ist festzuhalten, dass die BPMN durch so genannte „Markierungen“, die vielen der vorgestellten Elemente hinzugefügt werden können, noch mehr Möglichkeiten bietet, die Bedeutung der einzelnen Elemente weiter zu konkretisieren. So kann z. B. bei Ereignissen annotiert werden, ob es sich um ein „zeitlich getriebenes“ Ereignis handelt, welches z. B. immer nach 30 Minuten nach Prozessbeginn eintritt.

Wie auch die EPK ist die BPMN sehr flexibel bezüglich der Darstellungstiefe und -genauigkeit von Prozessen. Auch hier werden keinerlei Vorgaben für die Einhaltung eines einheitlichen Abstraktionsgrades getroffen. Die Möglichkeit der Darstellung von Ablaufvarianten wird ebenfalls gegeben. Die Verknüpfung der Aktivitäten zur Organisation wird über die Schwimmbahnen realisiert, bietet allerdings nicht die inhaltliche Qualität der EPK, da kein explizites Organisationsmodell angelegt wird.

Letztendlich handelt es sich bei den beiden vorgestellten Modellierungstechniken um sehr freie Techniken, die bezüglich ihrer Abbildungsmöglichkeiten kaum Einschränkungen mit sich bringen und sehr flexibel eingesetzt werden können. Die grundlegenden Elemente, die zur Prozessbeschreibung notwendig sind, also Aktivitäten, Ereignisse und die Darstellung von Ablaufregeln, werden gleichermaßen unterstützt. Bei der Abbildung von Input- und Outputdokumenten und notwendigen Ressourcen zur Bearbeitung, wie z. B. Informationssystemen oder Sekundärinformationen, bieten beide Ansätze zwar Konzepte an, diese sind aber nicht standardisiert, so dass Abstraktionsebenen und Darstellungen stark variieren können.

Grundsätze ordnungsmäßiger Modellierung
Konzeption und Praxisbeispiel für ein effizientes
Prozessmanagement

Becker, J.; Probandt, W.; Vering, O.

2012, XVI, 238 S. 91 Abb., Hardcover

ISBN: 978-3-642-30411-8