technische universität
dortmund

Technische Universität Dortmund

IT in Produktion
und Logistik

Fachgebiet
IT in Produktion und Logistik
Prof. Dr.-Ing. Markus Rabe

Konzeptionierung eines erweiterbaren Objektmodells für die ereignisdiskrete Simulation von Produktionsprozessen

Masterthesis zur Erlangung des Grades

Master of Science M. Sc.

Eingereicht am Fachgebiet
IT in Produktion und Logistik
Studiengang
Wirtschaftsingenieurwesen
Vertiefungsrichtung Industrial Management

Prüfer: Prof. Dr.-Ing. Markus Rabe

Betreuer: Dipl.-Geoinf. Maik Deininger

Ruven Baumgart
In der Esmecke 39 / 59846 Sundern
ruven.baumgart@tu-dortmund.de
Matrikelnr.: 164298

Fachsemester 4

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ARIS Architektur integrierter Informationssysteme 20

DES Discrete Event Simulation 12

EPK Ereignisgesteuerte Prozessketten 20

ITK Informations- und Telekommunikationstechnologie 9

MDD Model Driven Development 93

OMG Object Management Group 32

UML Unified Modelling Language 31

1. Einleitung

Die moderne industrielle Produktion unterliegt einem ständigen Druck, der aus steigenden Anforderungen an Flexibilität und Wirtschaftlichkeit resultiert. In vielen Branchen muss sich die Produktion überdies am Kunden ausrichten. Die steigenden Ansprüche der Kunden nach individuelleren Produkten, die zugleich auf dem neuesten Stand der Technik sein sollen und eine hohe Qualität besitzen müssen, verursachen zusätzliche Anforderungen an Unternehmen [Westkämper u. a., 2013]. Erstgenanntes hat zur Folge, dass sich Firmen mit der Aufgabe konfrontiert sehen, immer größere Produktindividualität zu gewährleisten. Gemäß dem zweiten Anspruch nehmen die Lebenszyklen erstellter Produkte stetig ab. Das erhöht den Druck auf produzierende Unternehmen, in immer kürzeren Zyklen neue Produkte zu erstellen oder ursprüngliche Produkte weiterzuentwickeln [Westkämper, 2006].

Innovationen sowie Produkterweiterungen und Verbesserungen verlangen dabei stets die Anpassung aktueller Produktionssysteme, auf denen die neuen oder zusätzlichen Arbeitsschritte implementiert werden müssen. Innovationen und technischer Fortschritt bedeuten immer auch Investitionen von monetären Mitteln. Je besser eine Planung a priori der realen Umsetzung und Inbetriebnahme eines neuen Produktionsprozesses, desto besser lassen sich in Betracht kommende Handlungsvorhaben und strategische Entscheidungen absichern, um in einem volatilen Markt mit hohem Konkurrenzdruck Fehlentscheidungen mit möglicherweise sogar prekären Folgen zu vermeiden. Die Organisation der Produktion hin zu effizienten, wertschöpfenden Prozessen nimmt somit einen großen Einfluss auf die Konkurrenzfähigkeit von Unternehmen [Westkämper, 2006].

Darüber hinaus ist die sich stetig verändernde Wettbewerbssituation zu berücksichtigen. Unternehmen müssen trotz eines Vorsprungs durch laufende Produkt- und Prozessinnovationen auf die Konkurrenz diese Stellung behaupten können [Nyhuis, 2010]. Letztendlich ist es unabdingbar, kontinuierlich in *neue, von Kunden und Markt* akzeptierte Produkte sowie in die Modifikation und Entwicklung von Produktionsprozessen zu investieren, um den Fortschritt des gesamten Unternehmens und damit die *langfristige* Wettbewerbsfähigkeit aufrechtzuerhalten [Batz u. a.,

1994].

Aufgrund dieser Umstände spielt eine hohe Variabilität und Flexibilität der Unternehmen eine große, wenn nicht entscheidende Rolle [Westkämper, 2006]. Im Zeitalter der Informations- und Kommunikationstechnologie bietet sich die Verwendung von rechnergestützten Hilfsmitteln an. Birta u. Arbez sehen in der Anwendung von Informationssystemen zur Unterstützung der Planung, Realisierung und Kontrolle von Produktionsprozessen einen Schlüsselfaktor für Produktionsunternehmen und schreiben der computergestützten Modellierung und Simulation eine essenzielle Funktion zu. Durch die Analyse mehrerer Handlungs- und Investitionsalternativen im Hinblick auf die Organisation der Produktion wird ein Unternehmen bei der Entscheidungsfindung unterstützt. Die Modellierung der Produktionsprozesse erlaubt es durch die Verwendung des Modells in einer Simulationsanwendung, verschiedene Alternativen zu prüfen, um schließlich die technisch sinnvollste und wirtschaftlichste umzusetzen [Birta u. Arbez, 2007].

Batz u. a. vertreten in ihrer Arbeit ebenfalls die Ansicht, dass für eine ständige Verbesserung der Wirtschaftlichkeit von Produktionsprozessen eine kontinuierliche Anpassung der Prozesse und des Produktionssystems unabdingbar sei. Maßnahmen können zumeist ausschließlich in rechnergestützten Modellen entwickelt und erprobt werden, da eine Unterbrechung der realen Produktion häufig unvereinbar mit möglichen ökonomischen Konsequenzen ist [Batz u. a., 1994]. Die einzige praktikable Lösung ist infolgedessen, den Produktionsprozess mit all seinen Komponenten und Eigenschaften rechnergestützt zu modellieren. Zukünftige Änderungen oder Modifikationen eines Produktionsprozesses lassen sich auf diese Weise am virtuell erstellten Modell simulieren und erproben. Die Modellierung und die Simulation sind somit hilfreiche Werkzeuge zur Unterstützung der Organisation und Bewertung von geplanten oder real existierenden Produktionssystemen und -anlagen, die vermehrt in Ingenieursdisziplinen zum Einsatz kommen [Birta u. Arbez, 2007]. Einige der häufigsten Anwendungsfelder sind:

- die Darstellung von Ist- und Soll-Zustand
- die Bewertung künftiger Entscheidungsmöglichkeiten und Alternativlösungen
- die Bewertung von Strategien einer geplanten Transformation oder Veränderung
- die Erstellung von Prognosen und Bewertung von Konzepten

Das Ziel dieser Arbeit ist, dem soeben beschriebenen Sachverhalt durch die Entwicklung eines Modellierungskonzeptes für die ereignisdiskrete Simulation von Produktionsprozessen zu begegnen. Aufgrund der Mannigfaltigkeit moderner Produktionssysteme scheint es evident, dass nicht für jedes Produktionssystem ein angemessenes Modell zur Abbildung vorhanden sein kann. Ein zweckmäßiges und konstruktives Modell zu erstellen, das sich für die jeweilige Zielset-

zung als tauglich und sachdienlich erweist und ermöglicht, die Logik von Produktionsprozesses abzubilden und dazu ebenfalls leicht erweiterbar ist, stellt den Kern dieser Arbeit dar und ist ein entscheidender Aspekt bei der Simulation von Produktionsprozessen. Es soll ein gefestigtes und umfassendes Modell entstehen, das als Referenzmodell bei der Modellierung vielfältiger Produktionssysteme herangezogen werden kann. Zudem soll sich das Modell durch eine ausgeprägte Flexibilität auszeichnen, um dem ständigen Evolutionsprozess von Produktionsprozessen berücksichtigen zu können [Westkämper u. a., 2013]. Aus diesem Grund kann das in dieser Arbeit entwickelte Modell nicht als vollständig gelten, sondern nur die Methodik zur Erstellung eines derartigen Modells verdeutlichen.

Um ein solches Modell zu konzeptionieren, liegt der Betrachtungsschwerpunkt der Arbeit auf der Beschreibung und der Modellierung von Produktionssystemen und -prozessen unter Verwendung einer grafischen Modellierungssprache. Grundsätzlich steht die Transformation von Produktionsprozessen in ein rechnergestütztes Modell im Vordergrund, das eine Beschreibung der Struktur und des dynamischen Verhaltens eines Produktionssystems zulässt. Verglichen mit anderen Bereichen der Modellierung ist die Prozessmodellierung noch nicht vollkommen ausgereift und bietet Stoff für intensive Diskussionen, unter anderem über die Gegensätze zwischen *klassischer* und *objektorientierter* Modellierung, die bei der Modellierung von Produktionsprozessen in der Praxis Anwendung finden. Im Anschluss an eine kurze Einführung in das Gebiet der Produktion und Simulation wird diese Diskussion aufgegriffen und auf Unterschiede beider Modellierungsmethoden kurz eingegangen. Gleichwohl liegt der Schwerpunkt der Arbeit auf dem Paradigma der objektorientierten Modellierung, da sich zeigen wird, dass sich das objektorientierte Paradigma im Hinblick auf die Aufgabenstellung besser eignet. Ferner gilt es, die Entscheidung für die Verwendung des objektorientierten Konzeptes als Grundlage für die Modellierung von Produktionsprozessen im Verlauf der Arbeit zu bekräftigen. Aufbauend auf der anschließenden Beschreibung von Grundlagen der Modellierung wird infolgedessen die Modellierung auf Basis der Objektorientierung konkretisiert. Hierfür wird das objektorientierte Konzept einer ausgewählten Modellierungssprache weiter ausgeführt, um neben weiteren Einblicken in die Objektorientierung die grafischen Symbole, die Syntax und Semantik einer solchen Sprache zu erfahren. Die vorgestellte Methode des objektorientierten Modellierens rückt nachfolgend in den Fokus der Arbeit und wird mittels der vorgestellten Modellierungssprache zur Modellierung eines Produktionsprozesses angewendet. Schlussendlich erfolgt nach der praxisorientierten Anwendung eine abschließende Bewertung und Analyse des objektorientierten Modells hinsichtlich Erweiterbarkeit und Eignung zur Abbildung von Produktionsprozessen.

2. Produktion und Simulation

2.1. Produktion

2.1.1. Produktionssysteme und Produktionsprozesse

Die Begriffe Produktionsprozess und -system entstammen dem Bereich der industriellen Fertigung von Produkten. Im Allgemeinen ist der Zweck eines jeden Produktionssystems die Erfüllung eines Kundenauftrags durch die Ausführung von zusammenhängenden Prozessen [Fischer, 1995]. In diesem Bezugsrahmen sind Produktionsprozesse technische Prozesse und stellen Transformationsprozesse dar. Solche Prozesse transformieren durch eine wertschöpfende Veränderung der Beschaffenheit Rohstoffe und Halbzeuge in industrielle Güter mit einem höheren Geldwert. Der Sinn eines Produktionssystems liegt folglich darin, in Bezug auf den Input einen Mehrwert zu generieren [Nyhuis, 2010]. Zudem lassen sich technische Prozesse weiter in technologische, logistische und unterstützende Prozessen untergliedern. Ein technologischer Prozess ist der Transformationsprozess im eigentlichen Sinne, der alle Prozessschritte zur Fertigung und Herstellung von Bauteilen und Produkten beinhaltet. Die Prozessschritte lassen sich wiederum in Fertigungs- und Montageprozessen differenzieren. Logistische Prozesse umfassen die materialfluss- und informationsflussorientierten Prozesse, die zur Sicherstellung eines reibungslosen Ablaufs der Produktion dienen. Als unterstützender Prozess ist die Bereitstellung von Hilfs- und Betriebsstoffen zu nennen [Westkämper u. a., 2013].

Im Allgemeinen lassen sich Prozesse innerhalb eines bestimmten Betrachtungsrahmens beobachten. Beziehen sich die Prozesse nicht auf den ganzen Betrachtungsrahmen oder möchte man bestimmte Einflüsse ignorieren, betrachtet man für gewöhnlich ein Segment. Das Segment, auf das man sich konzentriert, lässt sich als System auffassen [Kiess, 1995]. In dem Betrachtungsrahmen eines Unternehmens lassen sich Produktionsprozesse folgerichtig dem System Produktion zuordnen, und dementsprechend lässt sich auch von einem Produktionssystem sprechen. Der Systembegriff spielt eine bedeutende Rolle bei der Beschreibung von

Produktionsprozessen und nimmt Einfluss auf eine spätere Modellierung. Im Folgenden soll der Systembegriff als Rahmen von Produktionsprozessen näher beleuchtet werden.

Ein System lässt sich als eine Ansammlung von Komponenten verstehen, die in gegenseitiger, organisierter Wechselbeziehung stehen und im Hinblick auf die Erreichung eines logischen und zielgerichteten Resultats zusammenarbeiten. Ein System ist demnach dadurch definiert, dass es aus einer Menge von Elementen und einer Menge von Beziehungen unter diesen Elementen besteht, die gemeinsam dargestellt werden können und eine Struktur bilden. Ein System lässt sich ferner in Unter- oder Teilsysteme gliedern, in denen wiederum verschiedene Elemente interagieren [Freitag, 2005].

Objekte stellen gegenständliche (z. B. Arbeitsmaschinen), abstrakte oder organisatorische Bausteine des Systems dar und sind über eine definierte Systemgrenze von der Umgebung separiert. Die Systemgrenze kann durch Input (Informationen oder Ressourcen) und Output (ebenfalls Informationen und bspw. ein Produkt) durchstoßen werden, sodass ein System in diesem Fall als ein offenes System bezeichnet wird. Hat das System weder Eingangs- noch Ausgangsgrößen, spricht man von einem geschlossenen System [Freitag, 2005]. Die Systemelemente können zudem durch Relationen miteinander verbunden sein, deren Inhalt sich in einem Produktionsprozess z. B. auf den Austausch von Materialien oder Informationen bezieht [Böge, 2013; Mertins u. a., 1994]. Informationen können verschiedener Natur sein, wie zum Beispiel Einzelheiten über einen Auftrag oder quantitative Angaben über Materialien.

Im Rahmen der industriellen Produktion, also der Herstellung von Produkten und Dienstleistungen, unterscheidet man darüber hinaus folgende drei Arten von Systemen: technische (Maschinen-Systeme), soziale (Mensch-Systeme) und soziotechnische Systeme (Mensch-Maschine-Material-Mitwelt-Systeme). Arbeitssysteme und somit Produktionssysteme sind soziotechnische Systeme, in denen Menschen und Betriebsmittel zusammenwirken [Böge, 2013]. Bei Produktionssystemen handelt es sich darüber hinaus um offene und dynamische Systeme [Westkämper u. a., 2013].

Westkämper postuliert die Segmentierung eines Produktionsprozesses in zwei Bereiche, die der zu Beginn dieses Abschnittes beschrieben Klassifizierung ähnlich ist. Entsprechend dieser Einteilung lässt sich ein Produktionsprozess in einen technologischen und einen produktorientierten Bereich gliedern. Der technologische Bereich umfasst den primären Herstellungsprozess von Teilen, im produktorientierten Bereich konzentrieren sich Montageprozesse von Baugruppen und zuvor hergestellten Teilen. Infolgedessen setzt sich ein Produktionssystem in Anlehnung an REFA [Böge, 2013] und die im vorausgehenden Abschnitt ausgeführten Beschreibungen aus

Systemelemente eines Produktionssystem
· Arbeitsaufgabe – Auftrag · Material – Rohstoffe – Fertigteile – Halbzeuge · Betriebsmittel – Fertigungsmaschine – Montagemaschine

Tabelle 2.1.: Elemente eines Produktionssystems

den aufgeführten Elementen in **Tabelle 2.1** zusammen.

Es lässt sich somit konstatieren, dass in einem Produktionssystem Menschen, Informationen, Betriebsmittel und Materialien zusammenwirken, um eine bestimmte Arbeitsaufgabe zu erfüllen. Eine solche Arbeitsaufgabe könnte etwa die Herstellung von Halbzeugen oder Endprodukten sein, die aus verschiedenen Ressourcen wie Rohstoffen und Fertigteilen gefertigt werden. Auf ihrem Weg zum Endprodukt durchlaufen die Materialien mehrere Produktions- oder Herstellungsschritte, die zusammengenommen den Produktionsprozess kennzeichnen [Böge, 2013; Westkämper, 2006]. In **Abbildung 2.1** ist ein System gemäß der erörterten Definition modellhaft dargestellt.

2.1.2. Ereignisdiskrete Produktionsprozesse

Produktionssysteme lassen sich über die zuvor beschriebenen Eigenheiten hinaus noch durch weitere Aspekte beschreiben, die ein zugrundeliegendes System charakterisieren. Die weiteren Aspekte sind sog. Zustandsgrößen und Zustandsänderungen, die eine entscheidende Rolle bei der Klassifizierung von Systemen spielen.

Zustandsgrößen sind messbare Eigenschaften der Objekte innerhalb eines Systems. Die Eigenschaften beschreiben Verhaltensgrößen, die quantitativ und qualitativ erfasst werden können, sodass sich zu einem bestimmten, diskreten Zeitpunkt der Zustand eines Produktionssystems durch die Verhaltensgrößen beschreiben lässt. **Zustandsänderungen** werden als die Folge

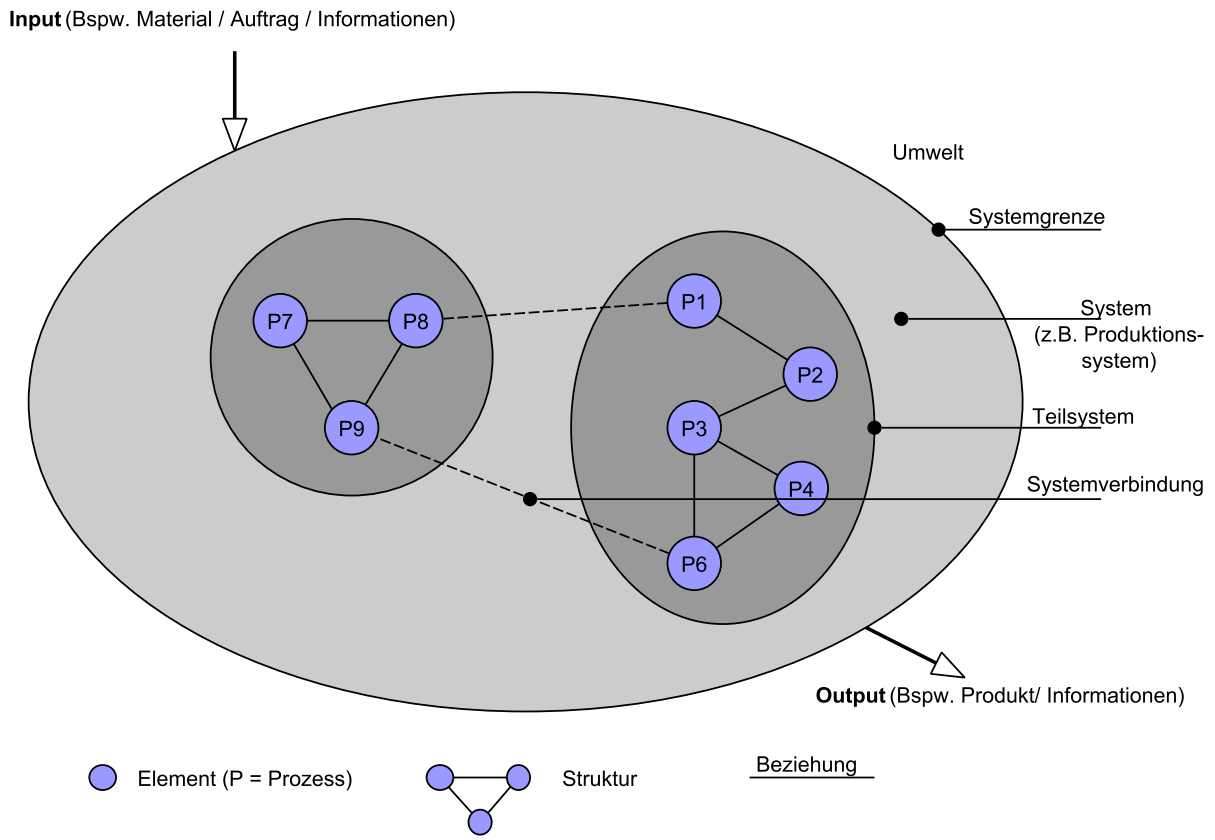


Abbildung 2.1.: Modellhafte Darstellung eines Systems nach [Freitag, 2005]

von Ereignissen verstanden, die durch ihr Eintreten Verhaltensgrößen von Objekten verändern. Zustände und Charakteristika der Zustandsänderungen erlauben eine Einteilung von Produktionssystemen in *diskrete* und *kontinuierliche* Systeme. Diskrete Systeme sind solche, bei denen sich die Zustandsgrößen nur zu bestimmten, diskreten Zeitpunkten aufgrund von eintretenden Ereignissen oder aufgrund von Prozessen innerhalb des Systems ändern. Demnach scheint es offensichtlich – analog zum Namen dieser Systeme –, dass in kontinuierlichen Systemen Ereignisse kontinuierlich auftreten und sich Zustände somit stetig verändern können. Dieser Aspekt der Art von Zustandsänderungen muss bei der Modellierung berücksichtigt werden. Verhaltensgrößen verändernde Ereignisse, die somit Auslöser einer Zustandsänderungen sind, können in einem solchen Szenario interne oder externe Ursachen haben [Fischer u. Ahrens, 1996].

Zustandsgrößen beschreiben in einem System quantitativ und qualitativ die Eigenschaften von Gegenständen, wie zum Beispiel die Anzahl von Halbzeugen in einem Puffer, die Verfügbarkeit von Materialien oder den Status einer Maschine. All diese Eigenschaften kommen durch entsprechende Ereignisse zustande und werden durch jene auch wieder geändert. Da Gegenstände

durch Objekte repräsentiert werden, lässt sich in diesem Sinne auch von Objektzuständen sprechen. Wie im späteren Verlauf der Arbeit noch deutlich wird, liegt in der letzten Aussage eine erste Analogie zur Modellierung vor. Auch hier werden Gegenstände aus einem realen System als Objekte verstanden und definiert [Byoung u. Donghun, 2013].

Ereignisdiskrete Produktionsprozesse sind die Art von Produktionssystemen, deren grundlegendes System auf den Charakteristika diskreter Systeme aufbaut, die in diesem Abschnitt beschrieben wurden. Da Zustandsänderungen durch das Eintreten von Ereignissen initiiert werden, die zudem ausschließlich zu diskreten, diskontinuierlichen Zeitpunkten auftreten, spricht man auch von ereignisdiskreten Systemen. Ein Produktionsprozess, der auf einem ereignisdiskreten Produktionssystem basiert, ist in **Abbildung 2.2** dargestellt. Entsprechend der Ereignisse *Arrive, Load* und *Unload* ändern sich die Zustände des abgebildeten Puffers und der Maschine. Beide sind Objekte des Produktionssystems [Byoung u. Donghun, 2013].

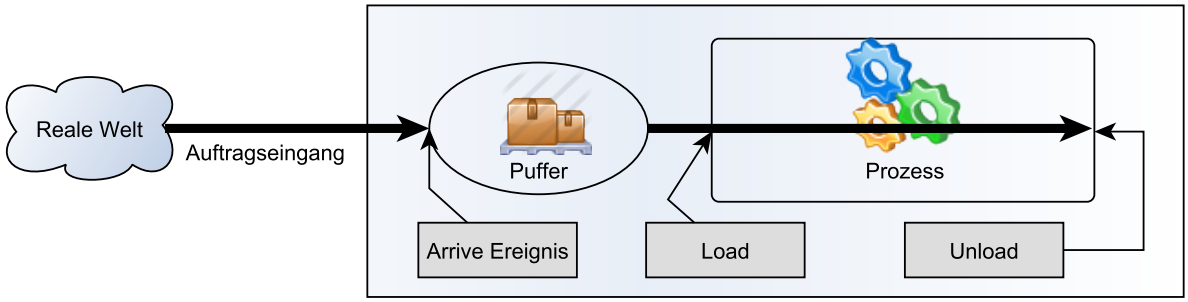


Abbildung 2.2.: Beispiel eines ereignisdiskreten Produktionsprozesses nach [Byoung u. Donghun, 2013]

Eine Eigenschaft ereignisdiskreter Systeme ist nun, dass sich der Systemzustand zu diskreten Zeitpunkten verändert. Ausgelöst werden diese Veränderungen durch Ereignisse, welche die aktuell vorhandenen Zustände von Objekten in einem System beeinflussen. Es handelt sich demnach um ein dynamisches System, charakterisiert durch Zustände und gesteuert durch Ereignisse [Birta u. Arbez, 2007; Byoung u. Donghun, 2013].

Ein dynamisches System bildet, wie in **Abschnitt 2.1.1** dargelegt, eine Einheit aus Identitäten, die sich gegenseitig beeinflussen. Über die Zeit lässt zwischen diesen Identitäten ein bestimmtes Verhalten beobachten. Es gibt in Bezug auf das Verhalten durchaus sehr komplexe Systeme, was dazu führen kann, dass man abhängig vom zu untersuchenden System und dessen Komplexität die Dynamik dieses Systems nicht vollständig erfassen kann [Hedtstück, 2013]. Das zugrunde gelegte Produktionssystem in dieser Arbeit soll sich jedoch nahezu vollständig beschreiben lassen und ein System geringer Komplexität darstellen. Ein bestimmter Input wird in einem

solchen einfachen Produktionssystem in einen definierten Output transformiert. Die Einfachheit des Systems ist eine Voraussetzung, um das Verhalten zu analysieren und nicht nur qualitative, sondern auch quantitative Aussagen über das System zu erhalten [Freitag, 2005; Birta u. Arbez, 2007].

2.1.3. Informationstechnologie in der Produktion

Anfang der 1950er-Jahre gewann die Informationstechnologie zunehmend an Bedeutung für die Entwicklung der industriellen Produktion [Mertins u. a., 1994]. Anfangs konzentrierte sich die Anwendung lediglich auf Teilaufgaben innerhalb eines breiten Spektrums an typischen Aufgaben für Produktionsunternehmen. Zu nennen sind hier das Computer Aided Design zur Unterstützung der konstruktiven Produktentwicklung sowie Numerical Control für eine rechnerunterstütze Bearbeitung von Bauteilen auf Basis programmierter Fertigungsschritte. Die rasante Entwicklung und die ansteigende Verfügbarkeit von Rechenleistung führten zu einer weiten Verbreitung der Informations- und Telekommunikationstechnologie (ITK). Die evolutionär vollzogene Entwicklung ließ aufgrund der Verfügbarkeit von fortgeschrittenen Datenverarbeitungssystemen eine umfangreiche Rechnerunterstützung in der gesamten Produktion entstehen, das sogenannte Computer Integrated Manufacturing [Mertins u. a., 1994].

Dank dieser Fortschritte in der Informationstechnik haben im Rahmen der Unternehmensmodellierung vor allem grafisch orientierte Beschreibungstechniken, wie sie zur Modellierung Verwendung finden, an Beliebtheit zugenommen. Ein stark prosperierendes Gebiet stellt die Modellierung von Produktionssystemen und -prozessen dar. Heutzutage existiert eine ganze Reihe von Methoden und Modellen für eine rechnerunterstütze Abbildung und Darstellung eines Sachverhalts im Bereich der Produktion. Gleichwohl umfasst die Produktionsprozessmodellierung nur einen Teil der Unternehmensmodellierung [März u. a., 2011].

Für die Modellierung von Produktionsprozessen eignen sich die verschiedenen Methoden unterschiedlich gut. Eine nicht neue, aber jüngst vermehrt an Beachtung gewinnende Methodik stellt in diesem Themengebiet die objektorientierte Modellierung dar [Mertins u. a., 1994]. Die meisten großen kommerziellen Lösungen zur Geschäftsprozessmodellierung sind heute dennoch klassisch orientiert, zumal noch einiges an Überzeugungsarbeit von erforderlich ist, ob sich denn die Theorie der Objektorientierung zur Modellierung von Produktionssystemen und -prozessen als tauglich erweist [Staud, 2010]. Durch die vorliegende Arbeit soll ein Teil der Überzeugungsarbeit geleistet werden.

Des Weiteren impliziert Staud, dass die Objektorientierung in der Prozessmodellierung noch am Anfang steht. In der Programmierung ist die Objektorientierung jedoch bereits fester Bestandteil. Als Beispiel hierfür wären Programmiersprachen wie C++ und Java zu nennen, also Sprachen, die sich zur formalen Beschreibung von zuvor erstellten, objektorientierten Modellen eignen und dabei helfen, das Modell in den Rechner zu überführen [Staud, 2010; Fischer u. Ahrens, 1996].

Modellierungsmethoden – unabhängig davon, ob klassisch oder objektorientiert – finden trotz alldem, getrieben durch die Verbreitung der ITK, vermehrt Verwendung in industriellen Unternehmen. Beide werden in dieser Arbeit kurz gegenübergestellt und anhand einzelner Kriterien bewertet. Ziel ist die Auswahl und Rechtfertigung der bestgeeigneten Methode zur Modellierung von Produktionsprozessen.

2.2. Simulation

2.2.1. Allgemeines über die Simulation

Unter Simulation wird eine experimentelle Methode zur Untersuchung funktioneller Eigenschaften eines Systems mithilfe von Modellen verstanden. Unterscheiden kann man in Simulationen mit und ohne Computer; gleichwohl bezeichnet die Simulation heutzutage fast immer die Computersimulation, wie auch in dieser Arbeit. Spielt das dynamische Verhalten eines Systems für die Simulationsanwendung eine Rolle, so spricht man von dynamischer Simulation [Birta u. Arbez, 2007]. Bspw. ist ein ereignisorientiertes Produktionssystem (s. **Abschnitt 2.1.2**) ein dynamisches System, das durch dynamische Simulation analysiert werden kann.

Der erste Schritt einer Simulation ist die Modellfindung, wobei das System, das als Grundlage eines zu entwickelnden und anschließend zu untersuchenden Modells dient, entweder bereits existieren oder zukünftig noch erstellt werden kann. Es wird also nicht ein originales Produktionssystem für eine Untersuchung herangezogen, sondern dessen Abbildung in der Form eines Modells. Die Simulation stellt auf diese Weise eine effiziente, kostengünstige und zumeist die einzige Möglichkeit dar, Produktionssysteme zu planen und deren Verhalten und Eigenschaften zu untersuchen [Staud, 2010; Fischer u. Ahrens, 1996].

Mithilfe der Simulation an einem Modell geht man davon aus, dass die im Laufe der Simulationsanwendung gewonnenen Erkenntnisse auf das reale System übertragen werden können. Grundlegend werden Simulationen durchgeführt, um Entscheidungsprozesse und Lösungsfindung zu unterstützen. Weiter bietet die Modellierung und Simulation neben der Möglichkeit der prospekti-

ven Planung von Produktionsprozessen eine alternative Herangehensweise zur Identifizierung von produktivitätssteigernden Möglichkeiten und Maßnahmen an. Die Simulation ermöglicht demnach einen Erkenntnisgewinn für die Steigerung der Wertschöpfung von Produktionsprozessen [März u. a., 2011].

Aufgrund einer zunehmenden Nutzung der ITK im Bereich der Produktion von Unternehmen, der Verfügbarkeit von Simulationssprachen wie auch des Fortschritts hinsichtlich der rechentechnischen Möglichkeiten basieren mittlerweile viele Entscheidungen bezüglich umfangreicher Produktionsprozesse auf Simulationen [Byoung u. Donghun, 2013]. Ferner liefern Simulationen fundierte Erkenntnisse über die Planung und die Verbesserungsmöglichkeiten eines Produktionssystems. So können mithilfe von Simulationen binnen kürzester Zeit endlich viele Prozessablaufanalysen durchgeführt werden und durch Variation der Steuerungsstrategie die effizienteste und effektivste selektiert werden [Birta u. Arbez, 2007; Byoung u. Donghun, 2013].

Ein in Planung befindlicher Produktionsprozess oder eine neue Produktionsstraße können vorab auf deren Realisierbarkeit und auf mögliche Erschwernisse hin geprüft werden. So lassen sich mögliche Fehlerquellen bereits im Voraus aufdecken und Lösungsalternativen erarbeiten. Summa summarum erleichtert die Simulation signifikant die Planung und führt in vielen praktischen Anwendungen zu Systemverbesserungen [Byoung u. Donghun, 2013].

Aufgrund der durch das Mooresche Gesetz prognostizierten zunehmenden Erhöhung der Integrationsdichte auf integrierten Schaltkreisen, profitieren softwaretechnische Anwendungen der Modellierung und Simulation durch die damit einhergehende Steigerung der Rechenleistung von Computern. Infolgedessen wird der Nutzen künftiger Simulationsmodelle aufgrund der höheren Datenmenge, mit der ein Softwareprogramm gefüttert werden kann, qualitativ immer hochwertiger und bietet zu den bereits beschriebenen Aspekten in vielerlei Hinsicht enorme Vorteile [Gadatsch, 2012].

Man unterscheidet die kontinuierliche und die ereignisdiskrete Simulation. Der ereignisdiskreten Simulation liegt ein ereignisdiskretes System (s. **Abschnitt 2.1.2**) zugrunde. Nachfolgend soll dieser Typ Simulation kurz erläutert werden.

2.2.2. Die ereignisorientierte Simulation

Im Bereich der Produktion werden hauptsächlich Simulationen von Computermodellen durchgeführt, die das Verhalten eines dynamischen Systems durch die Verwendung von Ereignissen und damit verbundenen Zustandsänderungen in diskreten Zeitpunkten beschreiben (vgl. **Ab-**

schnitt 2.1.2). Man spricht bei einem solchen Szenario neben der bereits bekannten Bezeichnung ereignisdiskrete Simulation auch von ereignisorientierter Simulation oder Discrete Event Simulation (DES) [März u. a., 2011]. Ereignisse sind in diesem Kontext Vorgänge, die durch ihr Auftreten die Verhaltensgrößen von Objekten, auch Zustandsgrößen genannt, im zeitlichen Verlauf eines Simulationsmodells ändern [Fischer u. Ahrens, 1996]. Basierend auf einem für diese Form der Simulation geeigneten Modell, werden in einem solchen Modell alle Ereignisse des Produktionsprozesses berücksichtigt [Hedtstück, 2013].

Als Beispiel soll ein einfaches Modell nach [Hedtstück, 2013] die ereignisorientierte Simulation verdeutlichen. In **Abbildung 2.3** ist die Fahrt eines Autos über zwei Kreuzungen grafisch und textuell ergänzend dargestellt. Die aufgeführten Ereignisse wie *Ankunft Warteschlange* charakterisieren den Zustand des Objektes Auto, der neben den anderen Zuständen innerhalb des blau hervorgehobenen Balkens abzulesen ist. Das Objekt Auto befindet sich nach Eintreten des soeben genannten Ereignisses im Zustand *Warten*. Nachdem das Folgeereignis *Beginn Überqueren* eintritt, ändert sich der Zustand des Autos in *Überqueren*. In Anlehnung an dieses

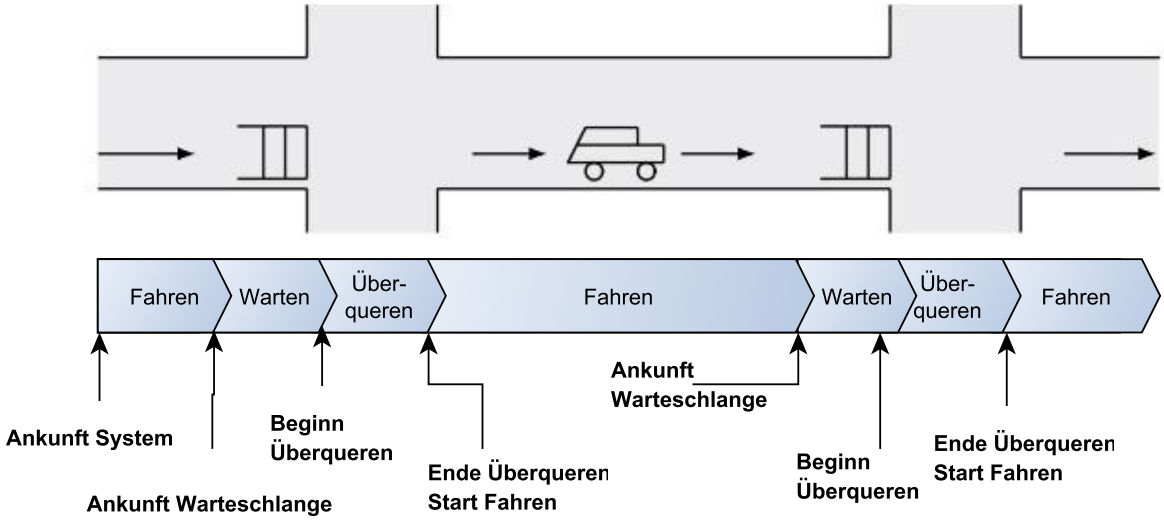


Abbildung 2.3.: Beispiel eines geeigneten Modells für die ereignisorientierte Simulation in Anlehnung an [Hedtstück, 2013]

Beispiel können in einem ereignisdiskreten Produktionsprozess Ereignisse wie *Teile eingetroffen* den Zustand einer Maschine von *Warten* in *Aktiv* überführen. Es ist nachvollziehbar, dass die Zustandsgrößen und Zustände von Objekten bei der ereignisorientierten Simulation vollständig von den Ereignissen abhängig sind und aufgrund des Auftretens von Ereignissen zu diskreten Zeitpunkten über einen bestimmten Zeitverlauf geändert werden.

Die Simulation gliedert sich im Verlauf eines Analysevorhabens eines Produktionssystems an die Modellerstellung an. Die Beschreibung der Entwicklung eines Modells mit einem geeigneten Werkzeug stellt den Schwerpunkt dieser Arbeit dar. Die Verwendung des entwickelten Modells in einer Simulationsanwendung ist eine relevante Anforderung, aufgrund deren ein geeignetes und adäquates Modell benötigt wird, bei dessen Entwicklung neben der verwendeten Modellierungsmethode vor allem der richtige und sinnvolle Einsatz der Modellierung eine fundamentale Rolle einnimmt. Der Prozess der Modellbildung liegt in den folgenden Kapiteln dieser Arbeit im Fokus der Betrachtung und dient dazu, ein Verständnis über die Vorgehensweise zur Erstellung eines solchen Modells zu erlangen.

3. Modellierung

3.1. Grundlagen der Modellierung

3.1.1. Allgemeines über die Modellierung

Die Modellierung wird immer dann angewendet, wenn es darum geht, ein Original für die Zwecke eines Subjekts auf vereinfachende Weise materiell oder immateriell zu repräsentieren [Becker, 2012]. Neben physischen Objekten handelt es sich bei solchen Originalen um Systeme. Das Vorhaben der Modellierung zeichnet sich demnach auch dadurch aus, ein System und dessen Eigenschaften in simplifizierender Darstellung abzubilden, sodass eine Beschreibung der Abbildung des Systems der Beschreibung des realen Systems sehr nahekommt. Mithilfe der Modellierung soll das System so dargestellt werden, dass möglichst einfach die Lösung einer Fragestellung bezüglich des Systems erreicht werden kann. Aus diesem Grund kann das Ergebnis einer Systemmodellierung nicht so komplex sein wie das reale System selbst. Entschließt man sich dazu, ein System zu modellieren, so richtet sich der Fokus der Modellierung auf einen bestimmten Realitätsausschnitt oder Gegenstandsbereich, der für ein beabsichtigtes Untersuchungsziel relevant ist [Byoung u. Donghun, 2013].

Innerhalb eines Gegenstandsbereichs lassen sich bestimmte Phänomene beobachten, für die sich ein Modellbildner interessiert. Phänomene werden in einem Modell durch sog. Objekte repräsentiert, die sich durch messbare Verhaltensgrößen (Eigenschaften) und durch eine Veränderung von Verhaltensgrößen (dynamisches Verhalten) beschreiben lassen. Verhaltensgrößen sind durch deren Typ und Wert bestimmt, wobei die Größe eines Wertes eine Dimension besitzen kann oder aber auch nicht. Beispiele sind die Geometriegrößen eines Bauteils oder die (dimensionslose) Anzahl von Kapazitäten in einem Puffer. Verhaltensgrößen, die auch als Zustandsgrößen bezeichnet werden, können sich über die Zeit verändern. Angeregt werden die Zustandsänderungen durch das Eintreten von Ereignissen [Fischer u. Ahrens, 1996].

Der Schlüsselbeitrag der Modellierung besteht darin, die Phänomene der realen Welt durch

Abstraktions- und Strukturierungsmechanismen strukturgetreu abzubilden. Die Abbildung von Phänomenen eines Gegenstandbereichs der realen Welt in ein Modell geschieht meist auf Basis einer grafischen oder textuellen Modellierungssprache. Erstgenannte manifestierten sich in der Verwendung von grafischen Symbolen und Zeichen zur Darstellung eines Sachverhalts. Die Verwendung der Darstellungselemente unterliegt dabei Regeln, die in einer sog. Syntax der jeweiligen Modellierungssprache formal definiert sind. Verwendet man eine textuelle Darstellungsform, so erfolgt die Abbildung durch ein System von Schriftzeichen. Generell sind Modellierungssprachen künstliche Sprachen und dienen zur Beschreibung von Sachverhalten, zählen also zu den deskriptiven Sprachen [Patig, 2006; Fischer u. Ahrens, 1996]. Zur besseren Einordnung soll **Abbildung 3.1** dienen.

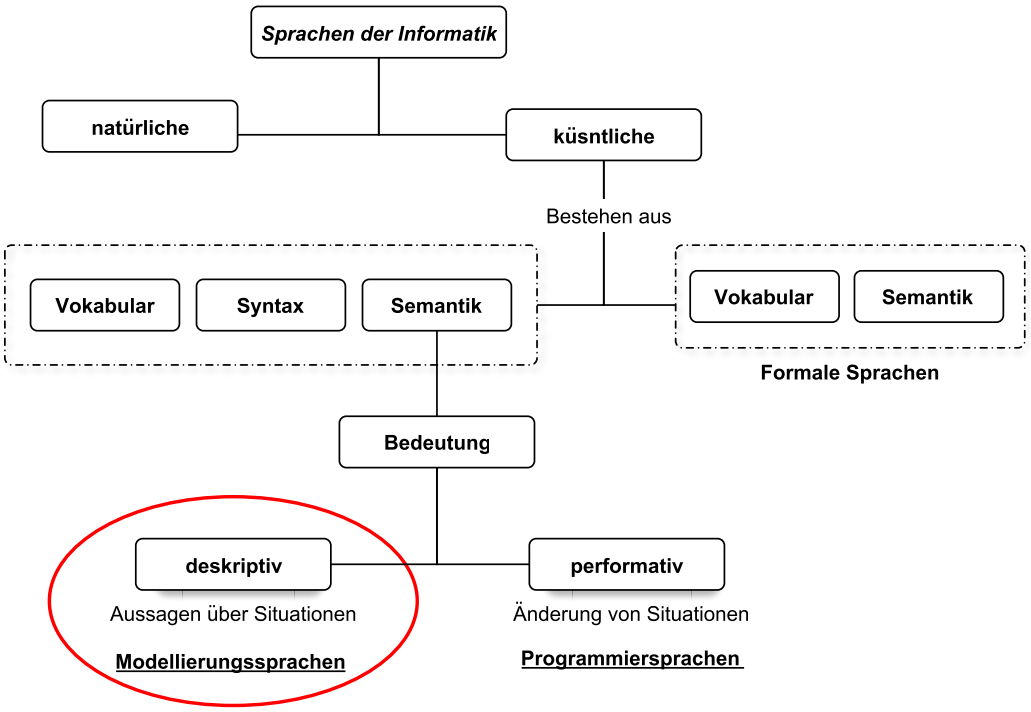


Abbildung 3.1.: Taxonomie der Sprachen der Informatik nach [Patig, 2006]

Es gibt eine Fülle sinnvoller Verwendungszwecke für die Modellierung, wobei je nach Intention des Vorhabens und der Art des Sachverhaltes unterschiedliche Anforderungen relevant sind. Beispielsweise sind die Anforderungen an die Modellierung und an ein resultierendes Modell im konstruktiven Bereich von Ingenieuren andere als in der Konzeptionierung von Produktionsprozessen. Das Modell des Ingenieurs ist meist ein um ein bestimmtes Verhältnis kleineres materielles Abbild eines Gegenstandes. Mithin ist ein Automodell ein geeignetes Beispiel für ein solches Modell. Im Gegensatz dazu besteht das Modell eines Produktionssystems aus immateriellen Symbolen wie Rauten, Sechsecken und Verbindungslinien, die sich von einem

Papier oder Bildschirm abheben [Westkämper u. a., 2013].

Es gibt verschiedene Gründe für die Verwendung von Modellen, wie bspw. die Nichtexistenz eines Produktionssystems und dessen Produktionsprozesse. In diesem Fall befindet sich das Vorhaben der Planung eines Produktionssystems bzw. einzelner Produktionsprozesse entweder im konzeptionellen Entwurf oder bereits in frühen Phasen der Umsetzung. Hier ist es prinzipiell nicht möglich, Untersuchungen und Experimente am nicht vorhandenen Produktionssystem durchzuführen. Ergo ist die Modellierung die einzige Möglichkeit, Zugang zu den gewünschten Informationen zu erlangen, auf die mit einer Untersuchung des Systems abgezielt wird. Bei existierenden Produktionssystemen können Gründe wie hohe Kosten, Gefährdung der Produktion, ein zu hoher Zeitaufwand oder ein zu hohes Risiko gegen die Durchführung von Experimenten am bestehenden (realen) System sprechen. Ferner bieten die gewonnenen Ergebnisse einer Modellierungsanwendung die Möglichkeit, als Endscheidungshilfe bei diversen Planungsvorhaben herangezogen zu werden, und leisten Hilfestellung bei der Lösung vielerlei Probleme [Westkämper u. a., 2013; Birta u. Arbez, 2007].

Die Modellierung dient also der Entwicklung von Modellen, wobei der Modellbegriff nicht nur materielle, sondern auch immaterielle Modelle beschreibt. In dieser Arbeit werden Modelle ausschließlich durch grafische Modellierungssprachen erstellt, was aufgrund des zuvor formulierten Sachverhalts Resultaten in Form von immateriellen Modellen entspricht. Unter den Modellbegriff in dieser Arbeit fallen demnach grafische Gebilde wie Diagramme. Was genau Modelle sind und wie Modelle charakterisiert werden, wird im Folgenden detailliert ausgeführt.

3.1.2. Merkmale von Modellen

Im Laufe dieser Arbeit wurden bereits Modelle zum Zweck der Demonstration eines Sachverhalts verwendet, wie z. B. in **Abbildung 2.3** aus **Kapitel 2.1.1**, in der das Beispiel eines Modells für die Abbildung eines ereignisorientierten Produktionsprozesses dargestellt wurde. Hinsichtlich der Systemmodellierung sind Modelle abstrakte Abbildungen eines Ausschnittes der Realität, mit deren Hilfe in der Regel auf vereinfachte Art und Weise ein Sachverhalt abstrahierend dargestellt werden kann. Nicht selten werden Modelle dann eingesetzt, wenn es darum geht, einen komplexen Ausschnitt der realen Welt zu primitivieren [Mertins u. a., 1994]. Modelle können in dieser Hinsicht Zeichnungen wie Diagramme, eine maßstäblich veränderte Darstellung eines Originals, abstrakte mathematische Beschreibungen oder auch Computerprogramme sein [Fischer u. Ahrens, 1996].

Immaterielle Modelle entstehen durch einen Konstruktionsprozess, in dem ein Modellierer seine subjektive Wahrnehmung eines abzubildenden Sachverhaltes mit all den für ihn relevanten Einflussfaktoren durch die Verwendung einer Modellierungssprache beschreibt [Staud, 2010]. Daraus folgt, dass nicht jedes Modell über denselben Ausschnitt der Realität einem anderen Modell gleicht. Aus Sicht eines Modellbildners bieten sich scheinbar unendlich viele Möglichkeiten, einen Sachverhalt mit signifikanten Merkmalen und Einflüssen durch ein Modell zu reproduzieren. Dennoch müssen Konventionen über die Merkmale eines Modells eingehalten werden, die sich in *Abbildungsmerkmal, Verkürzungsmerkmal* und *pragmatisches Merkmal* unterscheiden lassen [Becker, 2012; Martin u. Odell, 1999]. Zum umfassenderen Verständnis über den Modellbegriff soll die Beschreibung dieser Merkmale dienen.

Das ***Abbildungsmerkmal*** bringt zum Ausdruck, dass das Modell stets ein Abbild eines Originals darstellt [Becker, 2012]. Ein Modell muss jedoch nicht grundsätzlich Bezug zu einem Vorbild haben und wird häufig auch vor der Entstehung eines Originals entwickelt [Hedtstück, 2013]. Es lässt sich bei der Abbildung eines Originals auf ein Modell ebenso wie bei im Voraus entwickelten Modellen anhand der Ähnlichkeit des Modells in Bezug auf das Original bzw. des späteren Originals zwischen *isomorphen* und *homomorphen* Abbildungen differenzieren. Als Indikator wird das Ausmaß der Ähnlichkeit herangezogen [Mertins u. a., 1994].

Spricht man von isomorphen Abbildungen, so entspricht ein jedes Element aus der realen Welt einem Modellelement und umgekehrt. Elemente und Relationen des abgebildeten Systems werden originalgetreu reproduziert. Im zweiten Fall, der homomorphen Abbildung, lässt sich nur eine ausreichende Ähnlichkeit erkennen [Gab, 2014]. In dieser Arbeit wird sich, wie in **Abbildung 3.2** dargestellten, auf homomorphe, ein System abstrahierende Modelle beschränkt.

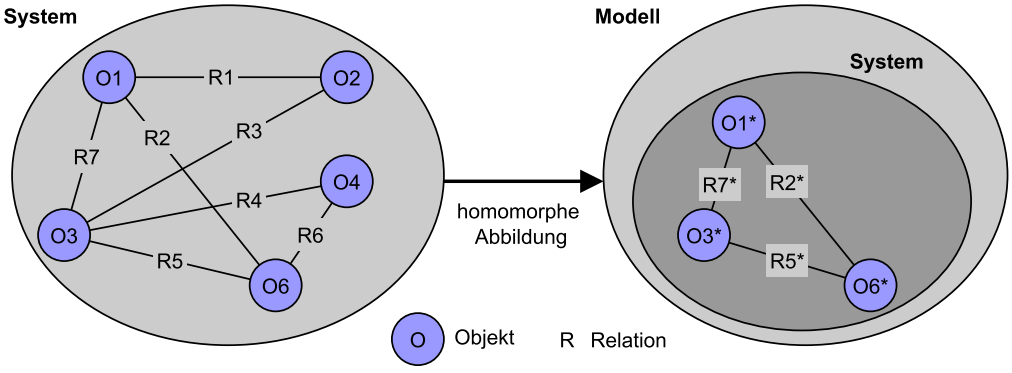


Abbildung 3.2.: Darstellung der Eigenschaften eines homomorphen Modells nach [Mertins u. a., 1994]

Das ***Verkürzungsmerkmal*** resultiert aus der Tatsache, dass ein Modell nicht die Gesamtheit

aller Elemente und Relationen der Realität erfassen kann und auch meist nicht soll. Ein Modell ist auf eine Auswahl von Gegenständen und Dingen beschränkt, die durch den Modellierer beeinflusst wird, und stellt eine Abstraktion eines beobachteten Sachverhalts dar. Abstraktion bedeutet in der Modellierung, dass die Struktur und das Verhalten eines Original-Systems mit einer geringeren Detailgenauigkeit im Modell beschrieben werden. Es wird bewusst auf Dinge verzichtet, die nicht dem Anspruch gerecht werden, für den Zweck der Repräsentation durch das Modell relevant zu sein [März u. a., 2011; Gab, 2014].

Jedes Modell ist aufgrund der Abstraktion folglich mit Informationsverlust behaftet. Die Folge dieser Tatsache ist, dass Modelle nur Dinge aus der realen Welt einbeziehen, die dem Zweck des Modells dienlich sind. Somit scheint klar, dass in einem Modell nicht alles Niederschlag findet, was bspw. in einem Produktionssystem tatsächlich beobachtet werden kann. In Bezug auf den Zweck, für den ein bestimmtes Modell jedoch erstellt wurde, ist es als vollständig anzusehen [Becker, 2012; Hitz u. a., 2005]. Ein Modell wird sinngemäß folgendermaßen definiert.

> Gabler Wirtschaftslexikon: „Auf der Basis von Funktions-, Struktur- oder Verhaltensähnlichkeiten bzw. -analogien zu einem Original, werden Modelle zum Zwecke speziell solcher Problemlösungen benutzt, deren Durchführung am Original nicht möglich oder zu aufwendig wäre".

Zuletzt bleibt noch die Beschreibung des ***pragmatischen Merkmals***, das die Sachdienlichkeit eines Modells beschreibt. Im Hinblick auf eine Problemstellung vereinfachen Modelle ein komplexes System auf pragmatische Weise und können in diesem Sinne wie folgt definiert werden.

> „Modelle stellen den Bezugsrahmen dar, der die Funktion beinhaltet, die wichtig für die Beantwortung der Fragen sind, für die das Modell geschaffen wurde [Mertins u. a., 1994, S. 2]."

Modelle erfüllen somit die beiden zuvor genannten Merkmale der Verkürzung und Abbildung. Unter Berücksichtigung der soeben beschriebenen Merkmale eines Modelles, die als Konvention der Modellbildung gelten, kann ein Modell in folgender Weise entwickelt werden und so aussehen, wie es in **Abbildung 3.3** dargestellt ist.

Neben den in diesem Abschnitt herausgearbeiteten Konventionen der Modellbildung werden weitere Aspekte zur Charakterisierung von Modellen herangezogen. Diese Aspekte beschreiben die Angemessenheit eines Modells und stellen Kriterien für die Bewertung der Modellierung bereit. Denn wie bereits erarbeitet wurde, stellen Modelle immer einen Rekonstruktionsprozess

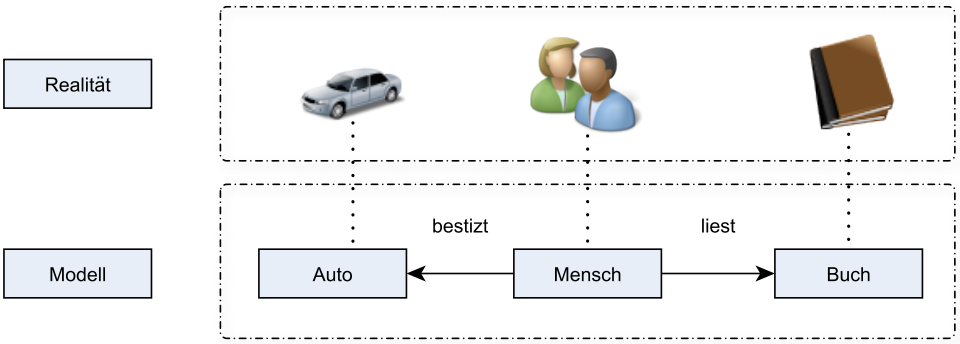


Abbildung 3.3.: Darstellung eines einfachen Modells

aus etwas Beobachtetem oder Gedachtem dar, und das resultierende Modell divergiert aufgrund der Subjektivität des Modellerstellers. Welches Modell eine höhere Qualität aufweist und seinen Zweck besser erfüllt, kann empirisch durch die Einhaltung folgender Grundsätze bestimmt werden, die in **Tabelle 3.1** zusammengefasst sind [Becker, 2012].

Grundsätze ordnungsgemäßer Modellierung nach [Becker, 2012]
1. *Richtigkeit:* Das Modell wurde syntaktisch korrekt erstellt und bildet das System in Struktur und Verhalten korrekt ab.
2. *Relevanz:* Nur die Teile des Systems wurden modelliert, die zur Lösung erforderlich sind. Das System wurde in passender Granularität abgebildet.
3. *Klarheit:* Das Modell ist anschaulich, strukturiert, übersichtlich und lesbar.
4. *Vergleichbarkeit:* Die Logik im System und in dem das System abbildenden Modell sind identisch. Unterschiedliche Modelle führen zum gleichen Ergebnis.
5. *Systematischer Aufbau:* Die Modellerstellung erfolgt nach nachvollziehbaren systematischen Grundsätzen.

Tabelle 3.1.: Zu berücksichtigende Grundsätze einer Modellierungssprache

Ein Modell erfüllt den Grundsatz der *Richtigkeit*, wenn es syntaktisch einwandfrei aufgebaut ist, die Modellierungsregeln der angewendeten Sprache demnach befolgt und eingehalten wurden. Auf diese Weise wird die Korrektheit des Modells gewährleistet; ob das Modell für den jeweiligen Zweck von Nutzen ist, wird durch die semantische Richtigkeit sichergestellt. Der Ausschnitt des

Produktionssystem, für den sich der Modellbildner interessiert, muss infolgedessen in Struktur und Verhalten korrekt abgebildet werden, sodass die Funktionsstruktur und die Dynamik, sc. das Verhalten, eines Systems durch das Modell in adäquater Weise repräsentiert werden. Die *Relevanz* besagt, dass nur Aspekte im Modell Niederschlag finden, die für den Modellierungszweck relevant sind. Die Repräsentation der relevanten Aspekte muss in der Weise erfolgen, dass das resultierende Modell den Grundsatz der *Klarheit* erfüllt und demgemäß einen Grad der Verständlichkeit aufweist, der durch die Ausgeprägtheit von Lesbarkeit, Anschaulichkeit und Strukturiertheit bemessen wird. Ein weiterer Grundsatz definiert die *Vergleichbarkeit* von Modellen. Damit ist zum einen die Vergleichbarkeit mit dem System selbst zu verstehen. Das bedeutet, dass identische Abläufe im System auch in dem System repräsentierenden Modell als identisch verstanden werden können und zum anderen, dass eine Vergleichbarkeit zwischen unterschiedlichen Modellen, die den gleichen Sachverhalt beschreiben gegeben ist. Der fünfte und letzte Grundsatz zielt auf den *systematischen Aufbau* des Modells ab [Becker, 2012].

Es lässt sich resümieren, dass ein Modell gemäß allgemeingültiger Konventionen und einzuhaltender Grundsätze beschrieben werden kann. Der Modellierer besitzt dennoch immer eine eigene Betrachtungsweise darüber, welche wesentlichen Eigenschaften eines Produktionssystems von Bedeutung sind. Je nachdem, was mit einer Untersuchung intendiert wird, muss ein Modell die Struktur, das Verhalten und die entsprechend relevanten Teile eines Systems abbilden, die für die angestrebte Lösung als unabdingbar gelten [Staud, 2010].

3.1.3. Die klassische Modellierung

Die Modellierung von Geschäftsprozessen oder speziell – wie in dieser Arbeit behandelt – von Produktionsprozessen kann durch verschiedene Herangehensweisen durchgeführt werden. Zum einen gebe es da die Alternative, die klassische Modellierungsmethode zu verwenden. Diese Art der Modellierung stützt sich unter anderem auf den Architektur integrierter Informationssysteme (ARIS)-Ansatz oder auf die Verwendungen nicht-objektorientierter Methoden, wie etwa Ereignisgesteuerte Prozessketten (EPK) [Staud, 2010].

Ein fundamentaler Baustein der klassischen Modellierung ist die strikte Trennung von Daten und Funktionen [Forbig, 2007]. So werden Datenflussdiagramme zur Modellierung von Prozessen und Entity-Relationship-Diagramme zur Darstellung der Informationsstruktur verwendet [Balzert u. a., 2011].

Ein Nachteil der klassischen Modellierung ist der, dass die erstellten Modelle weniger flexibel

gegenüber zukünftigen Änderungen und Erweiterungen sind. Dieser Nachteil spricht gegen eine Verwendung der klassischen Modellierung, da das Ziel der vorliegenden Arbeit ein Modell sein soll, das Flexibilität und Erweiterbarkeit seine Eigenschaften nennt. Überdies ist der Einsatz bei umfangreichen Modellierungsaufgaben kritisch zu bewerten, da eine gewisse Schwierigkeit darin besteht, Konsistenz zwischen Datenflussdiagramm und Entity-Relationship-Diagramm zu erreichen. Die Durchgängigkeit zwischen den verschiedenen Modellierungsphasen wird durch diesen Aspekt deutlich erschwert [Balzert u. a., 2011].

3.1.4. Die objektorientierte Modellierung

Neben der klassischen Modellierung gibt es einen weiteren Ansatz für die Modellerstellung; die objektorientierte Modellierungsmethode. Diese Entwicklungsphilosophie setzt sich mehr und mehr durch, da dieser Ansatz der Modellierung einige Vorteile gegenüber dem herkömmlichen Ansätzen bietet und als äußerst fruchtbar in der Modellierung von Produktionsprozessen gilt [März u. a., 2011; Vetter, 1998].

Die Objektorientierung ist ursprünglich ein Paradigma aus der Softwareentwicklung und fand zu Anfang Einzug in das Metier der Programmiersprachen. Beginnend bei Programmiersprachen wie SIMULA, über C++ bis hin zu Java, findet die Objektorientierung bis heute vermehrt Verwendung in diesem Bereich. Aber auch in der Modellierung stößt das objektorientierte Konzept auf immer mehr Beliebtheit [Hitz u. a., 2005].

Generell dient die Modellierung, basierend auf einer rechnergestützten Anwendung, der strukturgetreuen Abbildung von Phänomenen der realen Welt in der digitale Welt. Ein Phänomen in der Wirklichkeit, repräsentiert durch Gegenstände und Begriffe, entspricht im Modell einem Objekt [Fischer u. Ahrens, 1996]. Bei der objektorientierten Modellierung ist die Quintessenz – wie der Name dieser Modellierungsmethode vermuten lässt –, einen Gegenstand oder Gegenstände des Interesses sowohl in der Realität als auch in der Modellierung als Objekt zu betrachten. Objekte können alle physischen Gegenstände wie etwa ein Auto, ein Kundenauftrag oder aber auch immaterielle Dinge darstellen. Objekte lassen sich durch eine Menge von spezifischen Merkmalen charakterisieren, die in der Objektorientierung Attribute genannt werden. Fähigkeiten und Aktivitäten, die die Objekte ausführen können und die das Verhalten des Objekttyps beschreiben, werden durch Operationen dargestellt. Das Verhalten beschreibt mögliche Aktionsfolgen, die von Objekten ausgeführt werden können [Fischer u. Ahrens, 1996]. Operationen dienen überdies zum Nachrichtenaustausch zwischen verschiedenen Objekten, die auf diese Weise miteinander

interagieren können [Wöllhof, 1995].

Die objektorientierte Entwicklung und Darstellung von Modellen mittels Objekten lässt eine Klassifizierung der Objekte in Klassen zu. Klassifikationsmerkmale stellen gemeinsame Attribute und Operationen dar. Die objektorientierte Modellierung beruht demnach auf dem generischen Konzept, das durch das Zusammenfassen von nicht unterscheidbaren, spezifischen Eigenheiten eines Objektes in eine gemeinsame Klasse charakterisiert wird und sich infolgedessen durch eine strikte Zuordnung von Eigenschaften und Methoden zu Objekten auszeichnet [Rumpe, 2011].

Eine bereits gebildete Klasse lässt sich durch die in ihr enthaltenden Objekte beschreiben. Alle Struktur- und Verhaltenseigenschaften der Objekte werden der Klasse zugeschrieben, sodass die Klasse stellvertretend für die Objekte stehen kann. Eine Klasse kann dann ihre Struktur- und Verhaltenseigenschaften an andere Klassen weiterreichen, die diese übernehmen, aber auch erweitern können. In der Objektorientierung spricht man in diesem Fall von Vererbung [Rumpe, 2011]. Fasst man das soeben Formulierte zusammen, so beschreibt die Objektorientierung einen Abstraktionsprozess von Dingen in Bezug auf gemeinsame Merkmale. Wesentliche Gemeinsamkeiten werden zusammengefasst und in einer generischen Klasse aggregiert. Unterschiede, die als unwesentlich angesehen werden, bleiben ungeachtet. Was nun wesentliche und unwesentliche Eigenschaften sind, bleibt subjektiv und ist vom Modellierer abhängig [Rumpe, 2011; Martin u. Odell, 1999; Fischer u. Ahrens, 1996].

3.1.5. Vorteile der Objektorientierung

Wie bereits in **Abschnitt 2.1.1** bemerkt, setzt die Modellerstellung immer die Abgrenzung eines Systems gegen die Umgebung voraus. Ist ein Ausschnitt der Realität als System erkannt, so gilt es, das System in seine konstituierenden Komponenten zu zerlegen und diese bezüglich ihrer Eigenschaften und Fähigkeiten zu klassifizieren. Ferner ist die Dynamik des Systems zu beschreiben, die Wechselwirkung zwischen den Komponenten also. Für diese Aufgabe ist die Objektorientierung besonders geeignet und erlaubt, durch die systematische Abbildung eines Produktionssystems und inhärente Prozesse, das System klar und prägnant zu verstehen. Dazu werden in einer ersten Phase, genannt objektorientierte Analyse (OOA), die wesentlichen Komponenten des System identifiziert und die für das Modell relevanten Objekte und deren Eigenschaften gebildet, die in angemessener Weise die Komponenten des Produktionssystems repräsentieren. Auf diese grundlegende Phase folgt der objektorientierte Entwurf (OOD). Hier

geht es um den konzeptuellen Entwurf des Modells, der das zu modellierende System in entsprechender Vollständigkeit abbildet. Das bedeutet: Es wird nicht nur die Struktur des Systems, die sich aus den Objekten und deren Relationen untereinander zusammensetzt, modelliert, sondern das dynamische Verhalten des Modells und der Modellelemente wird abgebildet. Ein prioritärer Vorteil der Objektorientierung ist nun der, dass durch die konsistente Verwendung eines objektorientierten Ansatzes ein problemloser Übergang von OOA zum OOD möglich ist [Fischer, 1995].

Darüber hinaus sind die Anpassungsfähigkeit, die Flexibilität an zukünftige Veränderungen und eine einfache Erweiterbarkeit eines Modells als wesentliche Vorteile der Objektorientierung zu nennen [Hedtstück, 2013]. So indizieren Batz u. a. diesen Aspekt als einen ausschlaggebenden Punkt, der für eine Orientierung am objektorientierten Konzept als Grundlage der Modellentwicklung spricht. Es liegt in der Natur der Objektorientierung, flexibel und erweiterbar zu sein. Und ist nicht zuletzt aus diesem Grund eines der prosperierenden Konzepte in der Modellierung. Die Möglichkeit der einfachen Integration von Veränderungen und Modifikationen von Produktionsprozessen in das objektorientierte Modell bietet eine enorme Flexibilität und ermöglicht eine gezielte und situationsspezifische Anpassungsfähigkeit [Balzert u. a., 2011].

Ein weiterer Schlüsselbeitrag liegt in den Abstraktions- und Strukturierungsmechanismen, die durch die Objektorientierung ermöglicht werden [Fischer u. Ahrens, 1996]. Umgesetzt durch die Konstrukte der Objektorientierung, wie Klassen, abstrakte Klassen (vgl. **Kapitel 4.2.5** Seite 38) und natürlich der Objekte an sich, können auf abstrahierende Weise gemeinsame Eigenschaften auf einer ersten Ebene (Klasse) erfasst und durch Spezialisierung über weitere Ebenen spezifiziert werden. Auf diese Weise gelangt man unter anderem zu der Objektdefinition nach Kiess, der ein Objekt als *" an abstraction of a real-world entity "* definiert [Kiess, 1995]. Abweichende Objekte, die im Hinblick auf ihre Eigenschaften nicht in extenso mit Eigenschaften von bereits erfassten Objekten übereinstimmen, können so leicht in ein Modell inkludiert werden. Unterschiede dieser Objekte zu bereits existierenden Objekten müssen nur in der die Objekte zusammenfassenden Klasse beschrieben werden. Änderungen müssen somit nur an einer Stelle erfolgen, und verschiedene Objekte müssen nicht ständig neu definiert werden [Wöllhof, 1995; Balzert u. a., 2011]. Anpassungen in einem Produktionssystem können zum Beispiel bei Neuanschaffungen von Betriebsmittel oder bei der Änderung des Produktangebots anfallen [Westkämper u. a., 2013].

Zum anderen bietet die Objektorientierung einen Rahmen für das Arbeiten in einem Verbund, da Veränderungen eines Mitarbeiters leicht von anderen nachvollzogen werden können. Bereits

erstellte Klassen in einem Modell können für weitere Modelle übernommen werden. Auch eine spätere Wiederverwendung ist durch eine eindeutige Verständlichkeit gegeben. [Batz u. a., 1994].

3.2. Modellierung in der Produktion

3.2.1. Modellierung von Produktionsprozessen

Die Erstellung eines Modells zur Prozessmodellierung bildet nur einen Teil eines breiten Spektrums an Modellierungsanwendungen innerhalb von Unternehmen und ist Bestandteil umfassender Unternehmensmodellierungen. Staud postuliert jedoch, dass die Prozessmodellierung im Vergleich zu den restlichen Modellierungsanwendungen den größten Modellierungsaufwand besitzt [Staud, 2010].

Den Ausgangspunkt der Prozessmodellierung stellt ein ereignisorientiertes System (vgl. **Abschnitt 2.1.2**) dar. Wie zuvor beschrieben, wird mit der Modellierung nicht immer beabsichtigt, ein isomorphes Modell zu entwickeln. Vielmehr ist es möglicherweise auch von Interesse, die wesentlichen Komponenten zu identifizieren, die für einen Produktionsprozess als elementar gelten. Bedeutsame Aspekte, die in einem ereignisorientierten System modelliert werden müssen und fundamentale Elemente für eine spätere Simulation darstellen, sowie die Hauptkomponenten eines Produktionssystems sind: *Aktivitäten, Ereignisse, Zustandsgrößen, Materialien und Betriebsmittel*. Die drei erstgenannten wurden vor dem Hintergrund der ereignisorientierten Systeme in **Abschnitt 2.1.2** bereits näher beschrieben. Alle genannten Komponenten bilden zusammen das Referenzmodell ereignisorientierter Produktionssysteme und tragen jeweils zum Aufbau des Referenzmodells bei, das sich wie folgt zusammensetzt: Die erste von insgesamt drei Ebenen bilden die Komponenten oder Entitäten (Objekte). Objekte charakterisieren gleichzeitig den Kern des Referenzmodells und stellen das statische Modell dar. Die zweite Ebene, bestehend aus den logischen Komponenten des Systems wie Aktivitäten, Ereignissen und Zustandsgrößen, entspricht dem funktionellen Teil eines Modells. Die dritte und damit letzte Ebene bildet die Beschreibung der Systemdynamik [Byoung u. Donghun, 2013].

Die Struktur des hier beschriebenen Referenzmodells offenbart eine ausgesprochene Ähnlichkeit in Bezug auf das Konzept der Objektorientierung, wie es aus objektorientierten Programmiersprachen und Modellierungsphilosophien bekannt ist. Aus diesem Grund liegt es nahe, für die Modellierung von ereignisorientierten Produktionsprozessen das Paradigma der Objektorientie-

rung zu verwenden.

3.2.2. Modellierung mit objektorientierten Programmiersprachen

Modelle dienen, wie bereits erarbeitet, der Abbildung eines Realweltausschnittes in der digitale Welt. Eine weitere außerordentlich wichtige Rolle spielt im Hinblick auf eine Simulationsanwendung eine Programmiersprache, mit der man das objektorientierte Modell adäquat auf den Rechner bringen kann, sodass ein lauffähiger Softwarecode für ein Simulationsprogramm entsteht. Ein Vorteil des objektorientierten Ansatzes ist, wie im zurückliegenden Abschnitt deutlich wurde, die strukturelle Äquivalenz zwischen einem objektorientierten Modell und dessen Implementierung. Diese Äquivalenz erweist sich als besonders nützlich bei der Transformation eines erstellten Modells in ein ausführbares Modell für Simulationsanwendungen, das durch den Programmcode beschrieben wird [Fischer u. Ahrens, 1996]. Durch die Verwendung der Objektorientierung und einer passenden Modellierungssprache ist ein gradliniger und stringenter Übergang vom Modellierungsvorgang zur Implementierung in ein lauffähiges Programm ohne Schwierigkeiten möglich, da der objektorientierte Ansatz konsistent in allen Phasen der Modellentwicklung wie auch in der nachgelagerten Phase der Implementierung angewendet wird [Martin u. Odell, 1999].

Nun lässt sich deduzieren, dass objektorientierte Modellierungssprachen nicht notwendigerweise auf grafischen Notionen basieren müssen. Denn für Überführung des grafischen Modelles in Programmcode muss es objektorientierte Modellierungssprachen geben, die auf textueller Darstellung beruhen. Beispiele für solche Sprachen sind beispielsweise die Programmiersprachen C++ und Java.

Um das objektorientierte Modell in den Programmcode zu transformieren, ist es durchaus möglich, die Transformation durch einen geeigneten Generator durchzuführen, der dann automatisch den Programmcode generiert. Auch wenn der Programmcode nicht mittels Generator automatisch generiert wird, führen die konsistente Verwendung des objektorientierten Ansatzes und die Verwendung einer geeigneten, unterstützenden grafischen Modellierungssprache zu einer wesentlichen Beschleunigung des Transformationsprozesses. Dies hilft, Fehlerquellen zu vermeiden [März u. a., 2011]. Können des Weiteren bereits erstellte Modellkomponenten für die Simulation wiederverwendet werden, so lassen sich mithilfe der Computersimulation schnell und flexibel entscheidungsrelevante Informationen über ein System gewinnen [Fischer u. Ahrens, 1996].

Die Programmiersprache als wesentliches Werkzeug zur Abbildung eines Modells in ein Programm muss die syntaktischen und semantischen Paradigmen unterstützen, die zur Modellbildung verwendet wurden. Nur dann ist die strukturerhaltende Programmierung des Modells möglich. Im Falle des in dieser Arbeit vorgestellten objektorientierten Ansatzes muss die Programmiersprache die Konzepte *Klasse, Objekte, Vererbung, Relationen* und *Kommunikation* unterstützen (vgl. **Abschnitt 3.1.4**).

Im weiteren Verlauf der vorliegenden Arbeit wird sich nun ausführlich der Thematik des Modellierens mit einer grafischen Modellierungssprache gewidmet. Es werden verschiedene Modellierungssprachen kurz beschrieben, um anschließend eine geeignete Sprache für den Modellierungszweck dieser Arbeit auszuwählen. Auf die ausgewählte Modellierungssprache wird dann näher eingegangen, und das Konzept der Sprache wird explizit beleuchtet.

4. Auswahl und Darstellung einer Modellierungssprache

4.1. Auswahl einer Modellierungssprache

4.1.1. Anforderungen an die Modellierungssprache

Zur Modellierung eines Modells, das geeignet ist, ein Produktionssystem hinreichend genau zu beschreiben, ist die Nutzung einer entsprechenden grafischen Modellierungssprache notwendig. Des Weiteren muss gewährleistet sein, dass das Modell die resultierenden Anforderungen an eine angemessene Repräsentation von Produktionsprozessen erfüllt, die sich aus der Intention des Modells und der Modellierungsaufgabe ergeben. Eine geeignete Modellierungssprache, die vor dem Hintergrund der Aufgabenstellung eine zweckentsprechende Entwicklung eines Modells ermöglicht, soll im Folgenden erarbeitet werden. In diesem Kontext sei erwähnt, dass mit Sprache nicht die wörtliche oder schriftliche Sprache gemeint ist, sondern im weiteren Sinne Sprachkonstrukte, mit deren Hilfe etwas Bestimmtes ausgedrückt werden soll. Sprachkonstrukte stellen grafische Symbole dar und haben an sich schon eine Bedeutung, die durch präzisierenden Text ergänzt werden kann [Becker, 2012]. Nachdem aus einem Pool geeigneter Modellierungssprachen eine geeignete Sprache ausgewählt worden ist, die die Anforderungen im Hinblick auf die Intention des mit der Sprache zu konzeptionierenden Modells erfüllt, wird eine Einarbeitung in die Bedeutung der Sprachkonstrukte dieser Modellierungssprache erfolgen, mit deren Hilfe schließlich das Modell modelliert wird.

Eine Anforderung an eine potenzielle Modellierungssprache ist wie bereits ausgearbeitet, die Fähigkeit auf dem Fundament der Objektorientierung aufzubauen. Zum einen ergibt sich diese Anforderung aus der Beschreibung des Referenzmodells von ereignisorientierten Produktionsprozessen in **Abschnitt 3.2.1**, das mit dem des objektorientierten Konzeptes korreliert. Zum anderen lässt sich die Anforderung an die Modellierungssprache, auf dem Konzept der Objek-

torientierung aufzubauen, aus der Aufgabenstellung erkennen. Ferner wurden in **Abschnitt 3.1.5** die Vorteile des objektorientierten Paradigmas erörtert, die zusätzlich für die Verwendung einer objektorientierten Modellierungssprache sprechen.

Es gibt viele Modellierungssprachen, die objektorientiert sind und demnach das genannte Kriterium, auf der Objektorientierung aufzubauen, erfüllen. Aber auch weitere Anforderung sind von Belang. Einige ergeben sich aus den in **Abschnitt 3.1.2** beschriebenen Merkmalen von Modellen, die für die Sicherstellung der Qualität von Modellen erfüllt werden müssen. Andere sondieren sich aus der Modellierungsaufgabe, die hier die Modellierung von Produktionsprozessen darstellt.

In erster Linie ist es wichtig, dass das Modell eine übersichtliche und anschauliche Gestaltung ermöglicht, sodass abgebildete Prozesse wie auch das gesamte Produktionssystem an sich nachvollziehbar visualisiert werden können. Hier bietet sich die Verwendung von Modellierungssprachen an, die eine unmissverständliche Symbolnation durch grafische Symbole bereitstellen. Um das Merkmal der Klarheit (s. **Abschnitt 3.1.2**) zu erfüllen, ist es obendrein wichtig, dass die verschiedenen Bereiche eines Modells, also die verschiedenen modellierten Realitätsausschnitte, grafisch prägnant dargestellt werden können [Balzert u. a., 2011]. Das Modell muss sich so strukturieren lassen, dass eine Beschreibungen auf unterschiedlichen Abstraktionsebenen möglich wird, um mit geeigneten Abstraktions- und Strukturierungsmechanismen das gesamte Produktionssystem sowie Teile des Produktionssystems in entsprechender Granularität abbilden zu können. Die Möglichkeit der Strukturierung schließt die Darstellung unabhängiger, d. h. nebenläufiger sowie paralleler Prozesse mit ein. Letztgenanntes scheint im Rahmen der Zielsetzung unabdingbar zu sein, da in einem industriellen Produktionsprozess nicht notwendigerweise alle Vorgänge sequenziell Ablaufen. Auch die Ablaufsteuerung, basierend auf Bedingungen, spielt eine wichtige Rolle [Fischer, 1995].

Die Modellierungssprache muss ferner die Möglichkeit bieten, die verschiedenen Komponenten eines Produktionssystems abzubilden. Da sich die Komponenten eines Produktionssystems weiter untergliedern lassen und ihrerseits wiederum aus Komponenten zusammengesetzt sind, muss deren Komposition in einem ausreichenden Detaillierungsgrad zu veranschaulichen sein. Das bedeutet, dass mithilfe der grafischen Darstellungsmöglichkeiten der Modellierungssprache die Konzepte der Generalisierung und Spezialisierung bzw. der Komposition und Dekomposition idealerweise visualisiert und angewendet werden können.

Wie in **Abschnitt 2.1.2** und **2.2.2** bereits formuliert wurde, ist ein essenzieller Aspekt von industriellen Produktionsprozessen – der eine entscheidende Bedeutung für das in dieser Arbeit

erstellte Modell darstellt – das ereignisorientierte System. In dieser Art System stellen Zustände und Zustandsänderungen von Systemobjekten sowie Ereignisse fundamentale Bestandteile dar. Infolgedessen muss die Modellierungssprache in der Lage sein, die sich aus den Umständen der Zustandsänderungen ergebende Dynamik in geeigneter Form und in entsprechender Logik darzustellen. Die Anforderungen an die Modellierungssprache zur Darstellung eines erweiterbaren Objektmodells sind zusammenfassend in **Tabelle 4.1** dargestellt.

Vorgeschriebene Anforderungen
· objektorientierte Modellierung
· Übersichtlichkeit und Anschaulichkeit des Modells
· strukturierte Darstellung von Teilmodellen
· Darstellung von Nebenläufigkeiten
· Verwendung von Generalisierung und Spezialisierung
· Darstellung von Zustandsänderungen durch Ereignisse
· Abbildung von Logik
· Abbildung von Beziehungen zwischen Systemelementen.

Tabelle 4.1.: Anforderungen an eine geeignete Modellierungssprache für die Entwicklung eines erweiterbaren Objektmodells

4.1.2. Wahl einer geeigneten Modellierungssprache

Um eine geeignete Modellierungssprache für den vorliegenden Modellierungszweck auszuwählen sollen verschiedene Modellierungssprachen verglichen werden, sodass diejenige, die bezüglich den Anforderungen aus dem zurückliegenden Abschnitt am geeignetsten erscheint, selektiert werden kann. Dazu sollen zunächst in Betracht kommende Modellierungssprachen übersichtlich dargestellt werden, um einen vorläufigen Überblick über die zur Verfügung stehenden Tools für die rechnerunterstützte Modellierung zu erhalten. Entsprechend dem Schwerpunkt dieser Arbeit und den genannten Anforderungen eignen sich die in **Tabelle 4.2** generalisierend aufgeführten Sprachen.

Tabelle 4.2 zeigt die vier meistgenutzten Modellierungssprachen im Bereich der Prozessmodellierung [Gadatsch, 2012; Oestereich u. a., 2003]. Durch das Symbol O wird auf das Erfüllen und durch das Symbol X auf das Nichterfüllen einer Anforderung hingewiesen. Es wird deutlich, dass die UML auf den ersten Blick allen Anforderungen genügt und demnach die geeignete

Modellierungssprache	Notation	Sichtenkonzept	Hierarchische Darstellung	Nebenläufigkeit	Struktur und Dynamik	Objektorientiert
EPK	Grafisch Textuell	O	X	O	O	X
BPMN	Grafisch Textuell	X	O	O	O	X
Petri Netze	Grafisch Mathematisch	X	O	O	O	O
UML	Grafisch Textuell	O	O	O	O	O

Tabelle 4.2.: Ausgewählte Modellierungssprachen zur Prozessmodellierung

Modellierungssprache zu sein scheint. Die Auswahl einer Modellierungssprache – tendierend zur UML – soll durch eine kurz Beschreibung der genannten Sprachen präzisiert werden.

EPK

Im Bereich der Geschäftsprozessmodellierung finden Ereignisgesteuerte Prozessketten (EPK) vermehrt Einsatz und würden sich diesbezüglich also durchaus für die Modellierung in dieser Arbeit anbieten. Ebenso erlaubt diese Methode die Darstellung von Zuständen und Ereignissen [Gadatsch, 2012]. Faktoren, die gegen die Verwendung der EPK sprechen, sind zum einen der Mangel an Möglichkeiten, verschiedene Sichten auf ein System darzustellen. Zum anderen basiert diese Methode auf dem ARIS-Ansatz und orientiert sich folglich eher an der klassischen Modellierungsmethode und weniger an der objektorientierten [Staud, 2010]. Objektorientierung ist jedoch zwingend für die Eignung einer Modellierungssprache vorausgesetzt worden, sodass die Nichterfüllung dieser Anforderung den ausschlaggebenden Punkt für die Entscheidung gegen die EPK als Modellierungssprache darstellt. Für weitere und detailliertere Informationen in Bezug auf EPK sei auf Gadatsch [2012] und Becker [2012] verwiesen. Im Rahmen dieser Arbeit soll die soeben kurz formulierte Beschreibung genügen.

BPMN

Die Business Process Model and Notation (BPMN) ist eine grafische Modellierungssprache, die zur Darstellung von Geschäftsprozessen entwickelt wurde. Die Sprache verwendet Darstel-

lungselemente der Business Process Diagramme (BPD), die sich wiederum auf Elemente der Swimlane-Diagramme stützen (Abbildung von Organisationseinheiten, die Schwimmbahnen ähnlich sein sollen) [Gadatsch, 2012]. Das Swimlane-Prinzip erlaubt zwar eine übersichtliche Darstellung von Geschäftsprozessen, bietet aber wenig Erweiterungsmöglichkeiten, sodass eine umfassende, auf verschiedenen Sichten beruhende Repräsentation von Produktionsprozessen erschwert wird. Entsprechend der EPK verzichtet die BPMN ebenfalls auf eine Orientierung am objektorientierten Konzept und kommt für eine Verwendung zur Modellierung von Produktionsprozessen auch aus diesem Grund nicht infrage. Weiteres Wissen über die BPMN ist über Gadatsch [2012] zu beziehen.

Petri-Netze

Petri-Netze, auch bekannt als Stelle/Transitions-Netze, sind gerichtete, zustandsbasierte Graphen zur Modellierung von Transformationsprozessen und Systemen. Sie eignen sich zur Modellierung von Produktionsprozessen und bauen auf einem mathematischen Fundament auf, was ihre Anwendung hinsichtlich einer Analysemethode und einer späteren Simulationsanwendung begünstigt. Diese Modellierungssprache eignet sich zur Darstellung von statischen und dynamischen Systemaspekten und ermöglicht die Umsetzung des objektorientierten Konzeptes sowie die Darstellung von Nebenläufigkeiten und bedingten Abläufen [Priese u. Wimmel, 2008].

Umfassendere Prozessmodelle von realen Produktionssystemen neigen jedoch zur Unübersichtlichkeit und bieten keine ausreichende Möglichkeit der Modellierung verschiedener Sichten auf das System. Zudem ist die Notation uneinheitlich und wird meist als unverständlich und komplex bewertet [Gadatsch, 2012]. Trotz praktikabler Eigenschaften fällt die Entscheidung daher gegen die Verwendung der Petri-Netze als Modellierungssprache. Sollte Interesse aufseiten des Lesers bestehen, sich einen umfassenderen Blick über die Petri-Netze zu verschaffen, so sei auf Priese u. Wimmel [2008] verwiesen.

UML

Die Unified Modelling Language (UML) ist eine grafische Modellierungssprache zur Visualisierung, Dokumentation und Spezifizierung komplexer Systeme und eignet sich darüber hinaus zur Modellierung von Geschäftsprozessen [Rupp u. a., 2012; Oestereich u. a., 2003]. Die UML basiert auf dem objektorientierten Konzept und verwendet eine Vielzahl an verständlichen grafische Notationselementen. Aufgrund einer Vielzahl an übersichtlichen Diagrammtypen

wird die Modellierung verschiedener Sichten auf ein Modell ermöglicht, was eine umfassende Abbildung statischer wie auch dynamischer Systemaspekte erlaubt. Für die Abbildung von Nebenläufigkeiten und bedingten Abläufen stehen ebenfalls Darstellungselemente zur Verfügung. Ferner ist eine hierarchische Darstellung basierend auf dem Prinzip der Spezialisierung und Generalisierung möglich und erlaubt eine strukturierte Abbildung umfangreicher Systeme [Rupp u. a., 2012; Hitz u. a., 2005].

Die UML bietet sich dementsprechend als potente Sprache für das Modellierungsvorhaben in dieser Arbeit an und soll in den folgenden Abschnitten dieses Kapitels näher beschrieben werden. Neben einer detaillierteren Beschreibung der Modellierungssprache werden auf diesem Wege bereits genannte Aspekte der Objektorientierung erneut aufgegriffen und weiter ausgeführt. Wie genau das Konzept der Objektorientierung, vorgestellt in **Abschnitt 3.1.4**, in Verbindung mit der UML im Einzelnen aussieht, wird dem Leser dieser Arbeit infolgedessen noch klarer werden.

4.2. Die Unified Modelling Language

4.2.1. Einführung in die UML

Der Entwicklungsprozess der Unified Modelling Language beginnt im Jahre 1996 mit der Forderung nach einer Spezifikation eines objektorientierten Modellierungsstandards. Der Aufruf zur Erfüllung dieser Forderung wurde vom damals wichtigsten Standardisierungsgremium erlassen, der Object Management Group (OMG), und 1997 mit der UML Version 0.9 beantwortet. Die erste Version wurde unter Federführung der OMG stetig weiterentwickelt, was zu regelmäßigen Veröffentlichungen neuer Versionen führte und die UML, dank des notwendigen Rückhalts aus der Industrie, als Standard der Objektorientierung etablierte. Die UML ist durch die OMG sowie der International Organization for Standardization (ISO) als Standard der Objektorientierung anerkannt [Rupp u. a., 2012; Oestereich u. a., 2003]. 2011 veröffentlichte die OMG schließlich die bis heute aktuellste Version 2.4.1. Korrekturen und Verbesserungen betrafen inhaltliche Aspekte auf der Metamodellebene wie Semantikdefinitionen. Elemente, grafische Symbole zur Modellierung, und Diagramme blieben überwiegend unberührt [Rupp u. a., 2012; Hitz u. a., 2005].

Unter den aktuellen Standardverfahren wird die UML als diejenige Modellierungssprache angesehen, die sich besonders für die Prozessmodellierung eignet [Oestereich u. a., 2003]. Die Fülle grafischer Notationselemente, die sich durch eine intuitive Verständlichkeit auszeichnen,

ermöglichen eine eindeutige Visualisierung komplexer Systeme und bieten die Möglichkeit der Modellierung von statischen und dynamischen Systemaspekten. Viele der in der UML verwendeten Elemente sind grundlegend an die Objektorientierung angelehnt und werden durch spezifische Symbole und Darstellungsmethoden ergänzt. An dieser Stelle sei darauf verwiesen, dass die Unified Modelling Language keine Methode darstellt, sondern lediglich eine Sammlung von Notationen spezifiziert und einen Rahmen für das Vorgehen der Modellierung definiert [Rupp u. a., 2012].

Die UML beschreibt demnach eine einheitliche Notation und Semantik. Sie zählt zu den semantischen Modellierungssprachen und nicht wie EPK und Petri-Netze zu den Methodensprachen. Semantischen Modellierungssprachen ist gemein, dass die Freiheitsgerade der Modellierung in einem gewissen Rahmen liegen und die Vergleichbarkeit, Auswertbarkeit und Handhabbarkeit erstellter Modelle deutlich zunehmen [Oestereich, 2009; Becker, 2012].

Weiter machen Rupp u. a. deutlich, dass die UML sowohl für die Modellierung von komplexen Objektbeziehungen als auch für die Modellierung von Abläufen mit Nebenläufigkeitsanforderungen geeignet ist [Rupp u. a., 2012]. Neben der Tatsache, dass die UML die an sie gestellten Anforderungen (vgl. **Abschnitt 4.1.1**) ohne Abstriche erfüllt, fällt die Wahl auf die UML zur Verwendung als Modellierungssprache des Weiteren aus dem Grund, dass die UML von Balzert u. a. als Notation der Zukunft gehandelt wird und eine langwierige Verwendungsmöglichkeit erstellter Modelle sichergestellt wird [Balzert u. a., 2011].

Überdies ist die UML eine der verbreitetsten Modellierungssprachen der Objektorientierung. Dadurch, dass die UML einen Standard in der Softwaremodellierung darstellt, wird durch die Verwendung für die Modellierung von Produktionsprozesse eine Lücke zwischen der Prozessmodellierung und der Softwareentwicklung zur Ausführung eines Simulationsmodells geschlossen. Der objektorientierte Ansatz bekräftigt in diesem Zusammenhang die Eignung als umfassende Modellierungssprache und als Instrumentarium zur Prozessmodellierung [Staud, 2010; Rupp u. a., 2012; Hitz u. a., 2005]. Aus diesem Grund bietet die UML ein großes Potenzial in Hinblick auf die Verwendung zur Prozessmodellierung.

Ein Modell wird in der UML in Form von Diagrammen dargestellt [Hitz u. a., 2005]. Wie in **Abschnitt 3.1.2** bereits festgelegt wurde, fällt unter den Modellbegriff ebenfalls die Repräsentation in dieser Form. Die UML bietet viele verschiedene Diagrammtypen an, mit denen unter Verwendung unterschiedlicher Notationselemente bestimmte Sachverhalte abstrahierend modelliert werden können. Die verschiedenen Diagrammtypen eignen sich unterschiedlich gut, um bestimmte Modellierungsaspekte herauszustellen, und bieten verschiedene Sichten zur Modellierung

eines Produktionsprozess an. Aufgrund dieser Fülle an Diagrammtypen bietet die UML einen gewissen Spielraum, um einen Produktionsprozesse adäquat zu modellieren und sehr flexibel eingesetzt zu werden. Bei der Auswahl eines der verschiedenen Modelle (Diagrammtypen) nehmen die in **Abschnitt 3.1.2** beschriebenen Merkmale Einfluss. Ist ein geeignetes Modell für einen spezifischen Anwendungszweck ausgewählt, so dient dieses eine Modell dem Ziel, den spezifischen Modellierungsaspekt zu betonen und zu beleuchten. Die verschiedenen Diagrammtypen lassen sich in Struktur- und Verhaltensdiagrammen differenzieren und erlauben etwa die Modellierung der Statik und der Dynamik eines Produktionssystems.

Aufgrund des großen Spektrums an zur Verfügung gestellten Notationselementen und verschiedenen Diagrammtypen können in dieser Arbeit unter Berücksichtigung der Aufgabenstellung nur die wesentlichen grafischen Darstellungsmöglichkeiten erläutert werden, die für die Entwicklung eines objektorientierten Modells eines Produktionsprozesses nützlich und erforderlich sind. Dazu werden die verschiedenen Notationselemente vorgestellt, was zugleich, wie zum Ende des letzten Abschnittes angedeutet, einen umfassenderen Einblick in die Objektorientierung ermöglicht. Abschließend werden die Notationselemente in einem Verbund ausgewählter Diagrammtypen dargestellt.

4.2.2. Beschreibung und Darstellung von Objekten

Wie bereits deutlich wurde, kennzeichnet das objektorientierte Paradigma – in Analogie zum Namen – die Reduzierung von Dingen der real beobachtbaren Welt (Realweltobjekte) zu Objekten in der digitalen Modellwelt. Objekte beschreiben Phänomene des jeweiligen Anwendungsbereich und repräsentieren einen Gegenstand des Interesses. Objekte charakterisieren dabei nicht nur physische, haptische Objekte, sondern können auch Begriffe und immaterielle Dinge repräsentieren [Balzert u. a., 2011].

Realweltobjekten können Verhaltensgrößen zugeschrieben werden, die bei der Überführung in ein Modellobjekt durch Eigenschaften und Funktionen repräsentiert werden. Eigenschaften, genannt Attribute, repräsentieren eine Menge bezeichnender Merkmale und beschreiben überdies den aktuellen Zustand von Objekten [Staud, 2010; Vetter, 1998]. Das Objektverhalten wird durch Funktionen erfasst, die im Rahmen der UML Operation genannt werden. Das Verhalten eines Objektes beschreibt die Menge von Aktionen, die es ausführen kann. Es lässt sich dabei in interne Aktionen und Interaktionen unterscheiden [Fischer u. Ahrens, 1996]. Operationen können überdies angewendet werden, um die Attribute von Objekten zu ändern, indem durch

den Aufruf einer Operation eine im Objekt spezifizierte Methode ausgeführt wird [Forbig, 2007].

Operationen sowie Attribute werden innerhalb des entsprechenden Objektes deklariert und definiert. Unter dem Begriff Deklaration versteht man in der Informatik und in diesem Zusammenhang das Bekanntmachen von Attributen und Operationen. Man informiert über den Namen und den Typ einer Operation oder eines Attributs, ohne weitere Informationen zu nennen. Das Nennen von Informationen und Details erfolgt anschließend durch die Definition. Bei Operationen umfasst der Vorgang des Definierens etwa die Implementation des Funktionsrumpfes und die Beschreibung des Algorithmus, der sich hinter dem Operationsnamen verbirgt. Die Definition von Attributen umfasst das Festlegen von spezifischen Attributwerten [Gebhardt, 2013]. Die Repräsentation von Realweltobjekten in der UML lässt sich entsprechend der so eben beschriebenen Ausführung wie folgt definieren.

> „In an object-oriented system, each real world object is represented by an object to which is associated a state and a behavior" [Staud, 2010, S. 31]

Ein Objekt wird auch Instanz oder Entity genannt. Entity ist ein Terminus aus der begrifflichen und philosophischen Umwelt der Objektorientierung, speziell aus dem Anwendungsbereich der UML, und beschreibt im Sinne eines Objektes einen Informationsträger, der existiert und Eigenschaften besitzen kann [Staud, 2010].

Ein Objekte wird in der UML durch ein Rechteck dargestellt. Der Objektname steht zentriert im Rechteck und beginnt laut Spezifikation mit einem Kleinbuchstaben. Um zu verdeutlichen, dass es sich um einen Objektnamen handelt, wird der Name unterstrichen [Staud, 2010]. **Abbildung 4.1** zeigt die Darstellung eines Objektes in der UML. Als Beispielobjekt ist *bauteilX* gewählt worden, das in einem verarbeitenden Betrieb durchaus ein reales Objekt darstellen kann.

Abbildung 4.1.: Darstellung eines Objektes in der UML

4.2.3. Eigenschaften von Objekten

Eigenschaften von Objekten sind ein Beschreibungsmittel, um Objekte zu charakterisieren. Eigenschaften werden Attribute genannt und sind inhärente Eigenschaften eines Objektes. Attribute beschreiben die Daten, die von Objekten angenommen werden können, und werden mit einem individuellen Attributwert belegt. Attributwerte spezifizieren die Attributausprägungen

und speichern den aktuellen Zustand eines Objektes [Oestereich, 2009; Rumpe, 2011]. Als Attributwert kann genau eine Ausprägung, wie eine konkrete Zahl oder aber eine Menge von Ausprägungen, in diesem Fall ein Wertebereich, festgelegt werden [Staud, 2010; Balzert u. a., 2011].

Attribute eines Objektes werden unterhalb des Rechteckes des betreffenden Objektes mit einer entsprechenden Bezeichnung eingetragen. Die Attribute werden durch ihren mit einem Kleinbuchstaben beginnenden Attributnamen und durch ihren Datentyp beschrieben. Der Datentyp beschreibt den Wertebereich von Attributen, mit denen Operationen arbeiten und die man definieren sowie manipulieren kann [ITW, 2014]. Um eine fehlerhafte Verwendung im Voraus zu unterbinden, dient der Typ eines Attributs dem Zweck, in Hinblick auf eine möglichst präzise Darstellung, den Wertebereich eines Attributes einzugrenzen [Balzert u. a., 2011; Rupp u. a., 2012; Schneeweiss, 2013]. Datentypen in der UML sind bspw. *integer*, für numerische Datentypen, und *string*, für alphanumerische Datentypen.

Bestehen Attribute aus zwei oder mehreren Wörtern, so wird das jeweils nächste Wort beginnend mit einem Großbuchstaben und ohne Leerzeichen an das vorherige gehängt. Das anfangs verwendete Beispielobjekt *bauteilX* soll zur besseren Verständlichkeit Attribute zugewiesen bekommen. Das Resultat dieser Zuweisung ist in **Abbildung 4.2** zu erkennen.

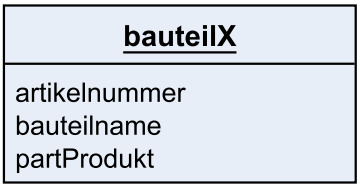


Abbildung 4.2.: Die UML Objektnotation mit Attributen des Objektes

4.2.4. Operationen

Ein Objekte in der UML übernimmt das Verhaltensmuster des Realweltobjekts, das es im Modell repräsentiert. Das Verhalten wird durch eine Menge von Aktionen definiert, die in der UML durch Operationen beschrieben werden. Operationen stellen demzufolge ein geeignetes Strukturierungsmittel für die Darstellung des Verhaltens von Objekten dar [Fischer u. Ahrens, 1996]. Durch Operationen lassen sich darüber hinaus die Attribute und Attributwerte – genauer die Daten und Zustände – von Objekten manipulieren [Balzert u. a., 2011; Rumpe, 2011]. Überdies lassen sich neben der Manipulation von Attributen neue Attribute durch entsprechende Operationen

ein- oder auslesen. Operationen können verschiedene Tätigkeiten ausführen und werden nach ihren Aufgaben klassifiziert. Man unterscheidet bspw. Operationen mit schreibendem Zugriff auf Daten, Operationen mit rein lesendem Zugriff, Berechnungsoperationen und Operationen zum Erzeugen und Löschen von Objekten [Balzert u. a., 2011].

Die Syntax einer Operation ähnelt stark der von Attributen. So beginnt der Name einer Operationen ebenfalls mit einem Kleinbuchstaben. Operationen werden analog zu den zuvor beschrieben Attributen ebenfalls unterhalb des Rechtecks zur Objektdarstellung eingetragen und mittels einer horizontalen Linie von den Attributen separiert. Gemäß der Attributsbezeichnung werden Operationen, sobald der Operationsname aus zwei oder mehreren Wörtern besteht, so dargestellt, dass das jeweils nächste Wort mit einem Großbuchstaben beginnt und ohne Leerzeichen an das vorherige gehängt wird. Zusätzlich lässt sich hinter dem Operationsnamen ein Übergabeargument in ovalen Klammer angeben. In **Abbildung 4.3** ist ein Objekt mit Operationen und vollständig deklarierten Attributen dargestellt. Zusätzlich wurde das Attribut *artikelnummer:int = 1253* mit den Attributwerten 1253 definiert.

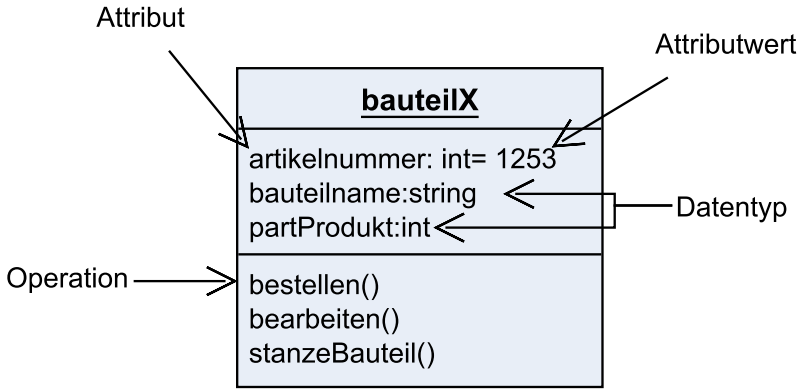


Abbildung 4.3.: Darstellung eines Objektes mit Operationen und Attributen

Es ist deutlich geworden, dass verschiedene Operationen aufgrund ihrer Tätigkeiten unterschieden werden können. Was jede einzelne Operation nun tatsächlich für Funktionen oder Tätigkeiten ausführt, wird durch ihre Implementierung bestimmt. Die Implementierung einer Operationen umfasst den Algorithmus bzw. den programmiersprachenspezifischen Rumpf einer Operation und wird Methode genannt [Rupp u. a., 2012; Oestereich, 2009]. Neben den aufgabenspezifischen Operationen gibt es noch weitere verschiedenartige Operationen, die durch ihre Methode und ihren Einflussbereich definiert sind. Balzert u. a. unterscheiden in diesem Sinne zwei grundlegende Arten von Operationen [Balzert u. a., 2011]:

- **Objektoperationen:** Objektoperationen werden fortwährend auf einzelne Objekte ange-

wendet. Die auf das Objekt angewendete Operation gehört zu der jeweiligen Klasse des Objekts. Beispielsweise ist die Operation *stanzeBauteil()* des Objektes *bauteilX* Teil einer möglichen Klasse Bauteil.

- **Konstruktionsoperation:** Objekte können in einem Produktionsprozess neu erzeugt werden. Dieser Vorgang wird durch Konstruktionsoperationen beschrieben. Ein Objekt wird entsprechend der Spezifikationen einer Klasse initialisiert. Solch eine Konstruktionsoperation könnte in einem Produktionsprozess in etwa *neuesWerkstück()* lauten.

Um die bisherige Thematik zu verdeutlichen, wird nun ein Beispiel eingebracht, das die bisher beschriebenen Aspekte aufgreift und das Vorgehen der objektorientierten Modellierung mit der UML verdeutlicht.

Ein Realweltobjekt sei in diesem Fall eine Gießanlage in einem metallverarbeitenden Unternehmen. Die Anlage wird nun als Objekt in einem Modell modelliert, das den Gießprozess abbilden soll. Mögliche Attribute, die das Objekt Gießanlage beschreiben können, sind etwa *füllstandMetall*, *artMetall* oder *legierungMetall*. Die Attribute definieren je nach Ausprägung den Zustand der Gießanlage. Um Eigenschaftsausprägungen des Objektes zu ändern, werden Operationen benötigt, die darüber hinaus das Verhalten der Anlage beschreiben. Beispiele für Operationen wären *gießen()*, *entleeren()* und *befüllen()*. Ein Attribut, das den Bereitschaftszustand beschreiben könnte und durch die Operationen *befüllen()* veränderbar wäre, ließe sich durch das Attribute *füllstandMetall* repräsentieren.

4.2.5. Das Klassenkonzept

Bisher wurde in den vorangegangenen Abschnitten herausgearbeitet, was man sich in der UML und Objektorientierung unter einem Objekt, den Eigenschaften und den Funktionen eines Objektes vorstellen kann. Um ein weiteres zentrales Elemente der UML sowie der Objektorientierung näher zu beleuchten, sollen die Aspekte über ein Objekt noch einmal aufgegriffen werden. Die wichtigsten Aspekte der zurückliegenden Kapitel sind aus diesem Grund in **Tabelle 4.3** noch einmal aufgeführt.

Für den nun folgenden Sachverhalt ist hervorzuheben, dass Objekte aufgrund der ihnen zugeordneten Attribute und Operationen eine gewisse Identität besitzen. Die Identität erlaubt es, Objekte voneinander zu unterscheiden. In vielen Fällen lassen sich Realweltobjekte eines Anwendungsbereiches aber durchaus in eine gleiche Kategorie einordnen. Entsprechende Modellobjekte können bezüglich ihrer Attribute und Operationen infolgedessen Ähnlichkeiten

Wichtige Gesichtspunkte von Objekten
· Objekte sind Gegenstände der realen Welt. Insbesondere repräsentieren sie Dinge oder auch Begriffe von größerem Interesse. · Ein Objekt lässt sich aufgrund seiner Identität von anderen Objekten unterscheiden. · Ein Objekt besitzt bestimmte Eigenschaften und ein Verhalten, repräsentiert durch Attribute und Operationen. · Die Eigenschafts- und Zustandsausprägungen eines Objektes werden durch Attributwerte näher spezifiziert. Das Objektverhalten sowie die Eigenschaftsausprägungen können durch eine Menge von Operationen verändert werden. · Ein Objekt kann mit anderen Objekten interagieren und in Beziehung treten

Tabelle 4.3.: Zusammenfassung wichtiger Aspekte von Objekte in der UML

aufweisen. Somit ist auch der umgekehrte Fall möglich, dass sich Objekte aufgrund ihrer Identität gleichen können. Objekte mit ähnlichen Attributen und denselben Operationen auf typ- und strukturgleiche Daten können in ihrer internen Struktur demnach gleich gestaltet sein. Bei solchen Gegebenheiten lassen sich Attribute sowie Operationen zusammenfassen und in einer sog. Klasse generalisieren. Die Klasse aggregiert folglich gemeinsame Eigenschaften und gleichartiges Verhalten von Objekten [Rumpe, 2011; Balzert u. a., 2011].

Aus dem gerade Beschriebenen folgt die Definition:

> „Eine Klasse beschreibt eine Sammlung von Objekten mit gleichen Eigenschaften (Attributen), gemeinsamer Funktionalitäten (Operation), gemeinsamen Beziehungen zu anderen Objekten und gemeinsamer Semantik" [Forbig, 2007, S.18].

Martin u. Odell postulieren, dass die Klasse in der objektorientierten Modellierung auch als eine Art Konzept beschrieben werden kann. Das Konzept definiert in diesem Sinne, was unter einem Objekt dieses Konzeptes zu verstehen ist. So beschreibt bspw. die Klasse Betriebsmittel ein Konzept, in das sich unter Berücksichtigung der Definitionen dieses Konzeptes ein Objekt Fertigungsmaschine einordnen lassen würde [Martin u. Odell, 1999].

Klassen dienen neben der Gruppierung von gleichartigen Objekten des Weiteren dazu, um Objekte zu erzeugen. Die von einer Klasse erzeugten Objekte werden Instanzen genannt und

der Prozess der Erzeugung wortgetreu Instantiierung. Eine Klasse wird somit als Bauplan für weitere Objekte angesehen. Dieser Bauplan enthält eine Beschreibung über die Struktur und über das Verhalten eines Objektes, mit dessen Hilfe neue Objekte erstellt werden können. Dem erstellten Objekt werden folglich die in der Klasse deklarierten und teilweise definierten Attribute und Operationen zugewiesen [Staud, 2010].

Die Ausprägung der Attribute kann jedoch divergieren und wird innerhalb des betreffenden Objektes definiert. Ausnahmen stellen konstante Attributwerte dar, die für die gesamte Menge von Objekten einer Klasse gelten. Gleiches gilt für die Operationen einzelner Objekte. Der Name einer gemeinsamen Operation kann durchaus der Gleiche sein, die Methode, d. h. die Reaktion oder Ausführung in Form eines Algorithmus, definiert sich innerhalb eines jeden Objektes dennoch auf eine andere Weise [Oestereich, 2009; Rupp u. a., 2012].

Neben den in **Abschnitt 4.2.4** beschrieben Operationstypen gibt es im Rahmen der Klasse zusätzlich noch die *Klassenoperationen*. Klassenoperationen werden nicht auf singuläre Objekte einer Klasse angewendet, sondern beziehen sich auf alle oder mehrere Objekte einer Klasse. Solch eine Operation könnte beispielsweise die Erhöhung des Stundenlohns einer bestimmten Menge von Angestellten innerhalb einer Klasse Mitarbeiter sein oder die Bestimmung des Vorrates bestimmter Rohstoffe einer möglichen Klasse Material. Die Darstellung von Klassenoperationen erfolgt in gleicher Weise wie die der Klassenattribute durch Unterstreichen des Namens der Operation [Balzert u. a., 2011; Rumpe, 2011; Staud, 2010].

Die Darstellung von Klassen erfolgt in der UML durch ein Rechteck, in das der Name der Klasse zentriert, in Fettschrift und mit einem Großbuchstaben beginnend eingetragen wird [Staud, 2010]. Besteht der Name der Klasse aus mehreren Wörtern, so wird das jeweils neue Wort mit einem einleitenden Großbuchstaben ohne Leerzeichen an das vorherige Wort angereiht. Unterhalb des Rechtecks können anschließend die Klassenattribute respektive die Objektoperationen und Klassenoperationen entsprechend den syntaktischen Regeln, wie sie auch für Attribute und Operationen von Objekte gelten (vgl. **Kapitel 4.2.2**), eingetragen werden.

Die Instanzen (Objekte) einer Klasse werden wie in **Abschnitt 4.2.2** beschrieben dargestellt. Als zusätzliches Darstellungselement, um die Zugehörigkeit eines Objektes zu seiner Klasse zu verdeutlichen, dient der mit einem Doppelpunktsymbol angefügte Klassenname [Staud, 2010]. Zur Veranschaulichung der soeben beschriebenen Darstellung von Klassen und deren Instanzen dient **Abbildung 4.4**.

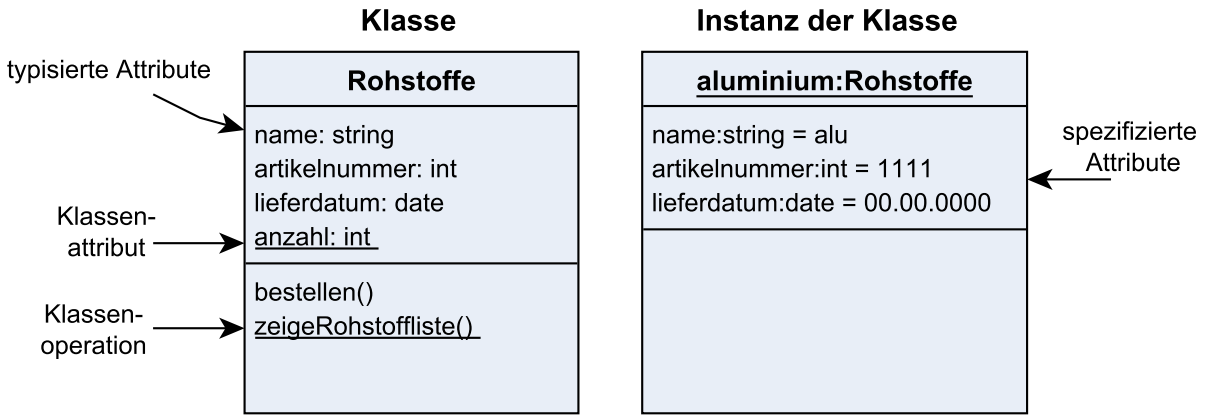


Abbildung 4.4.: Darstellung einer Klasse und einer Instanz in der UML

4.2.6. Botschaften als Form der Kommunikation

Wie bereits in **Abschnitt 2.1.1** beschrieben wurde, definiert sich ein System dadurch, dass die Systemelemente zusammenwirken, um einen zielgerichteten Zweck zu erfüllen. Gleiches gilt für Produktionssystem und involvierte Produktionsprozesse. Des Weiteren wurde deutlich gemacht, dass Elemente in der objektorientierten Modellierung durch Objekte abgebildet werden. Hier gilt ebenfalls, dass ein einzelnes Objekt für sich genommen höchstens eine Teilaufgabe innerhalb der Modellierung erfüllen kann [Kiess, 1995]. Im Gegensatz zum reduktionistischen Ansatz lässt sich ein Produktionssystem demnach nur durch die Gesamtheit an Elementen respektive Objekten beschreiben. Erst durch Interaktionen und eine Abfolge von Ereignissen kann eine Gesamtaufgabe erfüllt werden. Dazu müssen die Objekte und Klassen miteinander interagieren bzw. kommunizieren, was im Folgenden näher betrachtet werden soll.

Um Oestereich zu paraphrasieren, sind Objekte in diesem Zusammenhang eigenständige Einheiten, deren Zusammenarbeit und Interaktion mithilfe von Nachrichten bewerkstelligt werden, die sich die Objekte untereinander zusenden [Oestereich, 2009, S. 58]. In der UML kann eine Nachricht, auch Botschaft genannt, ein Signal oder der Aufruf einer Operation sein. So ist eine Nachricht wie folgt definiert.

> „Eine Nachricht überbringt einem Objekte die Information darüber, welche Aktivität von ihm erwartet wird, d. h. ein Objekt fordert ein anderes Objekt zur Ausführung einer Operation auf" [Staud, 2010, S. 94].

Die Interaktion zwischen mindestens zwei Objekten ist demnach durch die Realisierung einer durch eine Nachricht aufgerufenen Operation bestimmt. Nachrichten werden diesbezüglich

immer durch den identischen Namen der korrespondierenden Operation beschrieben. Das bedeutet, dass in dem Empfängerobjekt diejenige Operation aufgerufen wird, die den gleichen Namen, die gleichen Parameter und die gleichen Parametertypen wie die eintreffende Botschaft hat. Es muss also die entsprechende Operation mit der gleichen Bezeichnung vorhanden sein, die mit der Botschaft des Sendeobjektes übereinstimmt [Kiess, 1995; Forbig, 2007].

Die Darstellung von Nachrichten erfolgt durch einen richtungsweisenden Pfeil zwischen zwei kommunizierenden Klassen. Zusätzlich zu dem Namen der Nachricht können Übergabeparameter in ovalen Klammern angefügt werden, sodass weitere Informationen in Bezug auf die Operation übergeben werden können. Die Reaktion wird durch einen gestrichelten Pfeil dargestellt, kann aber, sollte die Antwort im jeweiligen Kontext ersichtlich sein, vernachlässigt werden. **Abbildung 4.5** verdeutlicht die Kommunikation zwischen Klassen.

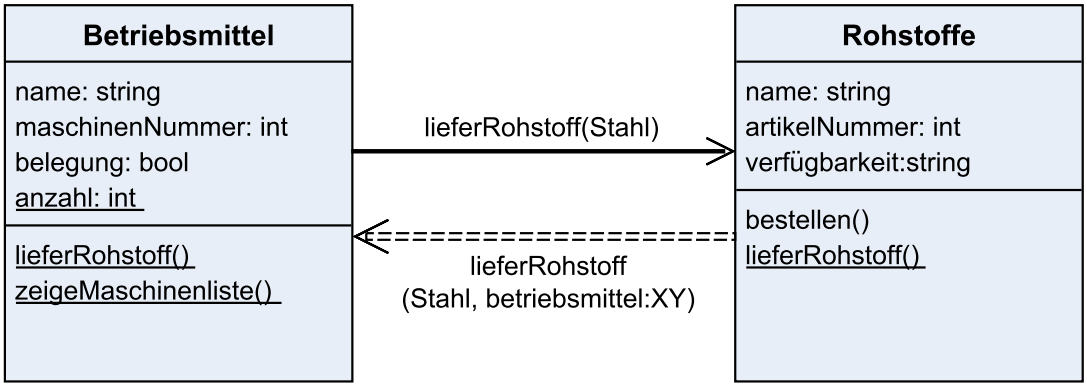


Abbildung 4.5.: Kommunikation zweier Klassen, dargestellt mithilfe einer Botschaft und der Reaktion auf diese Botschaft

In **Abbildung 4.5** ist ein mögliches Szenario aus einem Produktionsprozess abgebildet. Ein Objekt der Klasse Betriebsmittel sendet eine Botschaft *lieferRohstoff(Stahl)* an die Klasse Rohstoffe. Der Übergabeparameter macht deutlich, dass es sich um das Objekt Stahl handelt, das hier benötigt wird. In der Folge reagiert die Klasse Rohstoffe auf die eintreffende Nachricht und bedient das Objekt *betriebsmittel:XY* entsprechend der Interpretation der Nachricht.

Operationen können auf verschiedenen Wegen ausgetauscht werden. Diese zwei Wege werden zum einen durch den asynchronen und zum anderen durch den synchronen Nachrichtenaustausch beschrieben. Synchron bedeutet in diesem Zusammenhang, dass ein Sendeobjekt mit seinen weiteren Aktivitäten wartet und diese erst wieder aufnimmt, wenn das Empfängerobjekt den entsprechenden Operationsaufruf beendet hat. Asynchrone Nachrichten führen dazu, dass das Sendeobjekt mit seinem weiteren Tun nicht wartet, bis eine Operation durchgeführt wurde. In Produktionsumgebungen ist die Kommunikation üblicherweise asynchron [Staud, 2010;

Westkämper u. a., 2013].

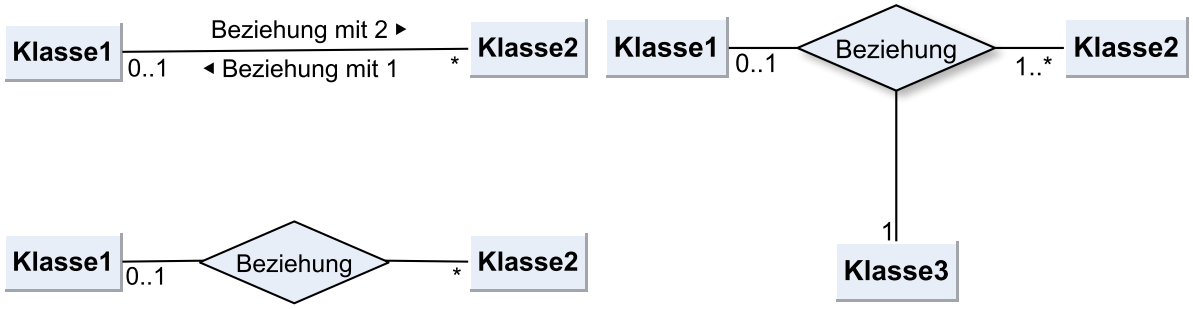
4.2.7. Assoziationen

Instanzen treten nicht isoliert auf, sondern gehen mit anderen Objekten Beziehungen ein. Eine Objektbeziehung wird in der UML Assoziationen genannt und modelliert die semantische Verbindung von Realweltobjekten im Modell [Martin u. Odell, 1999]. Assoziationen werden meist auf Klassenebene betrachtet, beschreiben jedoch Beziehungen zwischen den Instanzen der Klassen [Staud, 2010; Forbig, 2007]. So können verschiedene Beziehungen zwischen den Instanzen der Klassen **Material** und **Betriebsmittel** wie folgt aussehen:

- Betriebsmittel greift auf eine Vielzahl von Materialien zu,
- Betriebsmittel liefert eine Menge von Materialien,
- Material versorgt Betriebsmittel.

Unterschieden werden binäre und n-näre Assoziationen. Binäre Assoziationen sind Beziehungen zwischen zwei, n-näre Assoziationen sind Beziehungen zwischen mehreren Klassen. Binäre Assoziationen können darüber hinaus noch in *normale Assoziationen*, *Aggregationen* und *Kompositionen* differenziert werden [Balzert u. a., 2011].

Eine normale Assoziation wird durch eine einfach Linie, die sog. Assoziationslinie, zwischen den an der Assoziationsform teilhabenden Objekten dargestellt. Dabei besteht die Möglichkeit, eine Beziehung genauer zu kennzeichnen. Das geschieht durch eine Beschreibung der Assoziation oberhalb der Assoziationslinie oder innerhalb einer Raute, die zusätzlich als Verbindungspunkt von Assoziationslinien dienen kann. **Abbildung 4.6** zeigt beispielhaft die zwei Darstellungsmöglichkeiten von normalen Assoziationen.



(a) Einfache (binäre) Assoziation: Dargestellt mit einer Assoziationslinie und mit einer Raute

(b) mehrstellige (ternäre) Assoziation

Abbildung 4.6.: Assoziationen zwischen Instanzen einzelner Objektklassen

In der objektorientierten Modellierung können sich Klassen aus mehreren Objekten zusammensetzen, die sich ihrerseits wiederum in verschiedene Objekte dekomponieren. Ein analoger Sachverhalt liegt in Produktionsprozessen vor, die sich aus verschiedenen hierarchisch strukturierten Einheiten zusammenfügen. So besteht ein Produktionsprozess aus mehreren Produktionsschritten, in denen jeweils verschiedene Betriebsmittel und Materialien zusammenwirken.

In der objektorientierten Modellierung steht für das beschriebene Strukturmerkmal das Konzept der *Aggregation* zur Verfügung. Die Aggregation ist eine spezielle Form der Assoziation, wobei nur zweistellige Assoziationen auch eine Aggregation sein können [Vetter, 1998]. Die Aggregation beschreibt den Tatbestand, dass ein assoziiertes Objekt ein Teil eines anderen Objektes ist. Aufgrund dessen ist die Aggregation eine stärkere Form der Assoziation und wird auch *„Ist-Teil-von"* Assoziation genannt [Balzert u. a., 2011; Martin u. Odell, 1999; Forbig, 2007]. Für die an der Aggregation beteiligten Objektklassen wird für die übergeordnete Klasse der Begriff *Aggregationsklasse* und für die untergeordnete Klasse *Komponentenklasse* verwendet [Staud, 2010].

Eine strengere Variante der Aggregation stellt die *Komposition* dar. Die Komposition beschreibt eine Beziehung, ohne die ein an der Beziehung teilnehmendes Objekt nicht existieren könnte. Insofern ist die Komposition eine existenzielle Assoziationsform und beschreibt ebenfalls eine *„Ist-Teil-von"* Assoziation. Die Bindung eines Objektes zur seinem Aggregatobjekt ist aufgrund zusätzlicher Bedingungen jedoch stärker. Im Gegensatz zur Aggregation kann das untergeordnete Objekte nicht ohne das übergeordnete Objekt existieren; es besteht eine Existenzabhängigkeit. Ferner gilt, dass ein Objekt der Komponentenklasse mit nur einem einzigen Objekt der Aggregationsklasse assoziiert sein darf [Rumpe, 2011; Staud, 2010].

Die Darstellung von Aggregation und Komposition erfolgt in der UML durch eine leere bzw. durch eine ausgefüllte Raute am Ende einer Assoziationslinie. Die „normale" Assoziation, Aggregation und Komposition werden entsprechend der visualisierten Beispiele in **Abbildung 4.7** kenntlich gemacht.

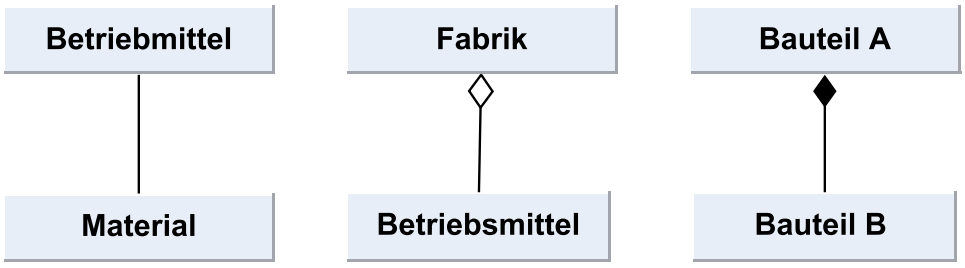


Abbildung 4.7.: Die verschiedenen Assoziationsformen der UML

Die Assoziation zwischen Betriebsmittel und Material ist eher eine weniger feste. Beide werden durch ein Nichtvorhandensein des anderen nicht in ihrer Existenz bedroht. Hiernach bietet sich also die Modellierung einer Assoziation an. Eine etwas festere Assoziation entspricht der Beziehung zwischen einer Fabrik und deren Betriebsmitteln. Ohne die Betriebsmittel kann ein produzierendes Unternehmen keine Produkte fertigen und ist demnach in seiner Existenz durchaus gefährdet. Eine Komposition könnte zwischen zwei Bauteilen A und B bestehen, da das eine nicht ohne das andere existent wäre.

Als zusätzliche Informationen besteht die Möglichkeit, sog. Kardinalitäten anzugeben. Kardinalitäten haben den Zweck, genau zu quantifizieren, wie viele Objekte der einen Klasse mit anderen Objekten einer an der Assoziation teilnehmenden Klasse in Beziehung stehen. Beispiele für Kardinalitäten in der UML zeigt **Abbildung 4.8**.

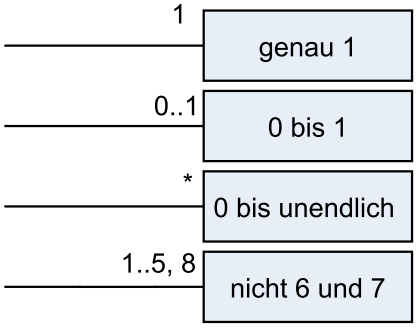


Abbildung 4.8.: Kardinalitäten der UML nach [Martin u. Odell, 1999]

4.2.8. Das Vererbungsprinzip

Ein substanzielles Konzept der UML sowie der Objektorientierung stellt das Prinzip der Vererbung dar [Forbig, 2007]. Das Prinzip der Vererbung determiniert ein Konzept, das neuen Klassen ermöglicht, die Struktur aus bereits erstellten Klassen zu übernehmen. Wird eine neue Klasse auf Grundlage einer bestehenden Klasse erstellt, so erbt die neue Klasse sämtliche definierten Eigenschaften und Operationen [Kiess, 1995].

Die vererbende Klasse wird Oberklasse bzw. Basisklasse genannt und die erbende Klasse Unterklasse. Die Unterklasse ist im Zeitpunkt ihrer Entstehung vollständig konsistent mit der Basisklasse, kann jedoch um weitere Elemente erweitert werden. Mit anderen Worten: Eine allgemeine Basisklasse kann weitere Unterklassen beinhalten, die in Bezug auf die Basisklasse um einen gewissen Grad expliziter, d. h. spezialisierter sind und durch spezifische

Attribute und Operationen ergänzen wurden [Balzert u. a., 2011; Rumpe, 2011]. Man umgeht also die Notwendigkeit, eine neue Klasse für neue Objekte zu modellieren und verwendet Ähnlichkeitseigenschaften, die hinsichtlich der neuen Objekteigenschaften entsprechend spezialisiert werden.

Das Vererbungsprinzip ist gewissermaßen bidirektional, da sich Unterklassen im Hinblick auf Gemeinsamkeiten in einer Oberklasse zusammenfassen lassen, was als *Generalisierung* bezeichnet wird. Bei der Generalisierung handelt es sich um das Zusammenfassen von Ähnlichkeiten in Bezug auf gemeinsame Attributen und Operationen von Objekten bzw. von Objektklassen. Ähnliche Objekte werden demzufolge einer generalisierten Klasse zugeordnet. Auf diese Weise entsteht eine hierarchische Klassen- und Vererbungsstruktur [Balzert u. a., 2011].

Die grafische Darstellung der Vererbung ist **Abbildung 4.9**. Zur Darstellung wird eine Linie mit einem geschlossenen, nicht ausgefüllten Pfeil verwendet, der von der Unter- auf die Oberklasse deutet. Zum besseren Verständnis sind die jeweiligen Attribute und Operationen, die aus der Oberklasse an die jeweilige Unterklasse vererbt werden, nochmals in den Unterklassen aufgeführt und farblich hervorgehoben. Üblicherweise wird die farbliche Markierung und das erneute Aufführen in einer Unterklasse nicht praktiziert.

In **Abbildung 4.9** vererbt die Klasse Rohstoffe ihre Attribute und Operationen an die Unterklasse Metalle. Man kann auch sagen: Die Klasse Metalle spezialisiert die Klasse Rohstoffe. Neben den *klassenAttributen* und *objektOperationen* hat die Klasse Metalle also weitere spezifische Attribute und Operationen, die erneut an eine Unterklasse weitergegeben und wiederum spezialisiert werden können.

Abstrakte Klassen

In der UML und der objektorientierten Modellierung gibt es Klassen, die nur aus konzeptionellen Gründen konzipiert werden. Zweck einer solchen *abstrakte Klasse* ist, dass die Klasse ihre Attribute und Operationen für weitere Klassen verfügbar macht. Abstrakte Klassen können keine Instanzen erzeugen und beinhaltet demnach keine eigenen Objekte [Hitz u. a., 2005]. Eine abstrakte Klasse ist infolgedessen lediglich eine Abstraktion von weiteren Klassen, um deren gemeinsamen Eigenschaften sinnvoll zu abstrahieren und weiterzuvererben [Martin u. Odell, 1999]. Dass eine abstrakte Klasse modelliert wurde, wird in der UML-Notation durch einen kursiv geschriebenen Klassennamen verdeutlicht, wie es in **Abbildung 4.9** bei der Klasse *Rohstoffe* zu erkennen ist.

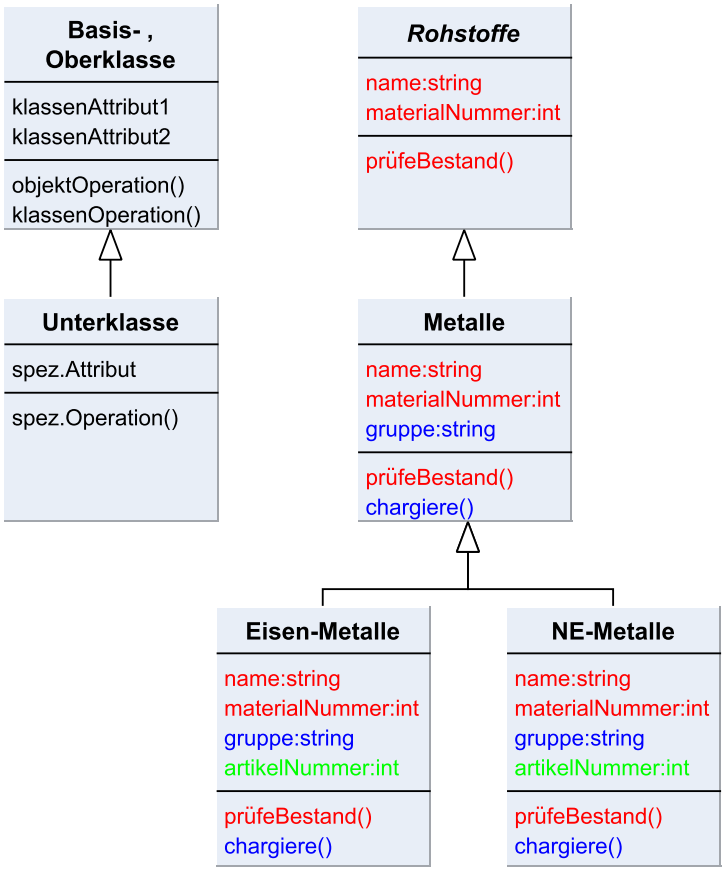


Abbildung 4.9.: Grafische Darstellung der Vererbung in der UML

4.3. Diagrammtypen der UML im Überblick

4.3.1. Strukturdiagramme

Wie in **Abschnitt 3.1.2** formuliert, fallen unter den Modellbegriff auch immaterielle Darstellungsformen wie Diagramme. Sinn und Zweck der in den vorherigen Kapiteln vorgestellten grafischen Notationssymbole ist einzig die Entwicklung und Erstellung von Diagrammen. Im Folgenden wird nun auf diese Art Modelle näher eingegangen, die den Kern einer Modellierungssprache ausmachen.

Die unterschiedlichen Sichten auf ein Produktionssystem können durch verschiedene Diagrammtypen repräsentiert werden. Die verschiedenen Diagramme beschreiben auf einer vorgegeben Abstraktionsebene einen Teil des Modells und damit wiederum einen durch das Modell repräsentierten Realitätsausschnitt. Ein Modell setzt sich somit nicht nur aus einem einzigen Diagramm zusammen, sondern in Hinblick auf eine logische, geeignete und umfassende Abbildung eines System aus mehreren Diagrammtypen [Oestereich u. a., 2003; Vetter, 1998].

Die Diagrammtypen der UML teilen sich in erster Instanz in Struktur- und Verhaltensdiagramme und lassen sich jeweils weiter untergliedern. Durch die Kombination der einzelnen Diagramme aus den beiden Oberkategorien (Struktur- und Verhaltensdiagramme) lassen sich die Produktionsprozesse in einem Produktionssystem umfassend modellieren [Hitz u. a., 2005; März u. a., 2011].

Die Strukturdiagramme dienen der Modellierung der Struktur eines abzubildenden Produktionssystems. Unter die Strukturdiagramme, die in dieser Arbeit Anwendung finden, fallen das *Klassendiagramm* und das *Objektdiagramm*. Für die Beschreibung des dynamischen Verhaltens eines Produktionssystems werden in der UML Verhaltensdiagramme zur Verfügung gestellt. In dieser Arbeit werden das *Sequenzdiagramm* sowie das *Zustandsdiagramm* beschrieben [Oestereich u. a., 2003].

Abbildung 4.10 zeigt einen Überblick über einige der Diagrammtypen der UML und hebt jene hervor, die im Folgenden detaillierter erläutert werden sollen.

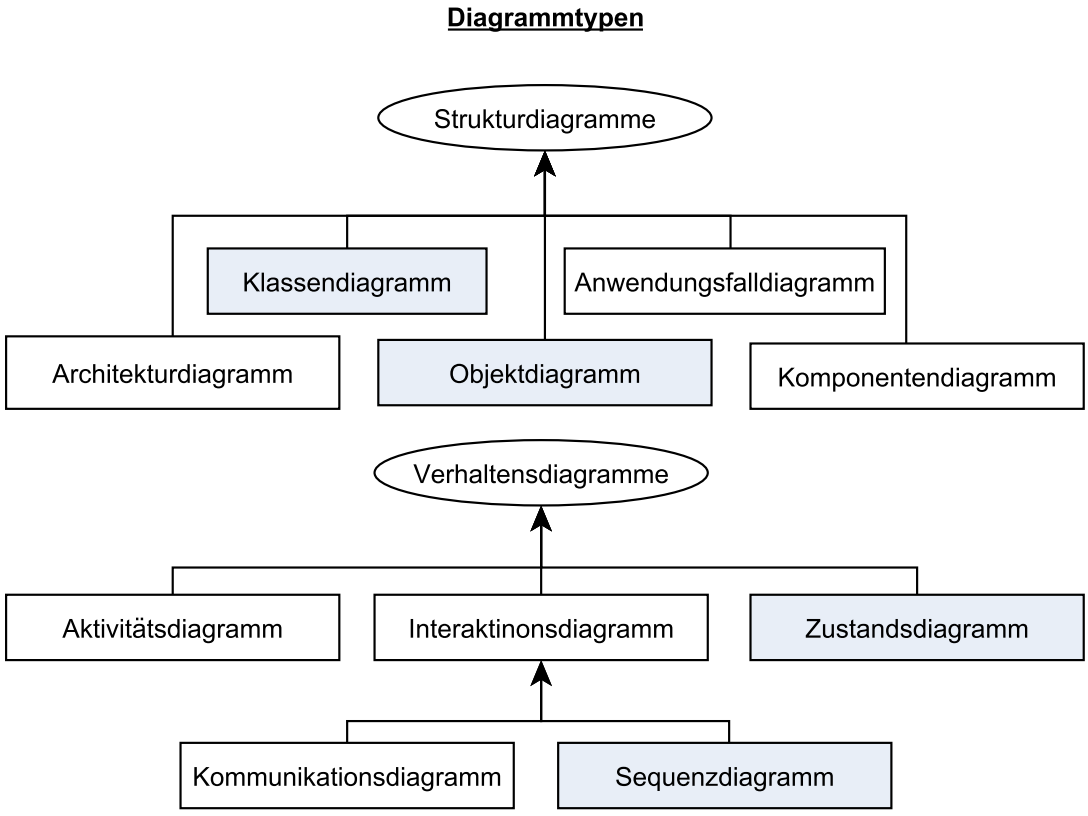


Abbildung 4.10.: Diagrammtypen der UML nach [Hitz u. a., 2005]

Das Klassendiagramm

In **Kapitel 4.2** wurde bereits auf Klassen und deren Beziehungen mit anderen Klassen eingegangen. Durch das Arrangement von Klassen und Assoziationen lässt sich deren Verbund in einem entstehenden Klassendiagramm darstellen. Das Klassendiagramm stellt eine wesentliche architektonische Beschreibung eines Systems dar und determiniert eine Art Makrosicht auf das abzubildende Produktionssystem [Rumpe, 2011].

Das Klassendiagramm ist ein Diagramm zur Darstellung eines Systems aus der zentralen Modellsicht der UML. Es stellt das Rückgrat eines zu modellierenden Systems dar und modelliert dessen Struktur. Es zeigt die wesentlichen statischen Eigenschaften der Klassen und liefert ein grundlegendes Verständnis darüber, wie der Aufbau eines Produktionssystem strukturiert ist [Rupp u. a., 2012]. Staud und andere Autoren machen deutlich, dass das Klassendiagramm in der objektorientierten Modellierung eines der wichtigsten Diagramme darstellt [Staud, 2010].

Das Klassendiagramm verwendet die grundlegenden grafischen Notationssymbole der UML wie *Klassen, Attribute, Operationen, Assoziationen (Aggregation und Komposition), Generalisierung* und *Spezialisierung*. In **Abbildung 4.11** ist die Darstellung eines Klassendiagramms am Beispiel von konstituierenden Objekten eines Unternehmens wie Personen und Abteilungen illustriert. Zur genaueren Beschreibung der verwendeten Elemente sei auf das zurückliegende **Kapitel 4.2** verwiesen.

Das Objektdiagramm

Neben den Klassendiagrammen, die die korrekten und logischen Zusammenhänge zwischen Klassen auf abstrakte Weise modellieren, sind für die Darstellung der Struktur eines Systems des Weiteren Objektdiagramme geeignet [Oestereich, 2009]. Ein Objektdiagramm hilft, eine detailliertere Sicht auf ein System zu geben und kann ein Klassendiagramm durch weitere Ausprägungen auf einer tieferen, detaillierteren Abstraktionsebene bereichern. Objektdiagramme werden somit am häufigsten dazu verwendet, Instanzen von Klassen mit ihren dazugehörigen Attributen, Attributausprägungen und Beziehungen zu modellieren. Die Darstellung der Konstellation von Instanzen beschränkt sich auf einen bestimmten Augenblick innerhalb des Systems und beschreibt demnach eine bestimmte Systemkonfiguration zu einem bestimmten Zeitpunkt. Der modellierte Systemausschnitt weist hiernach eine zeitlich limitierte Gültigkeit auf und stellt Objekte ausschließlich in ihrem aktuellen Zustand dar [Staud, 2010; Rumpe, 2011].

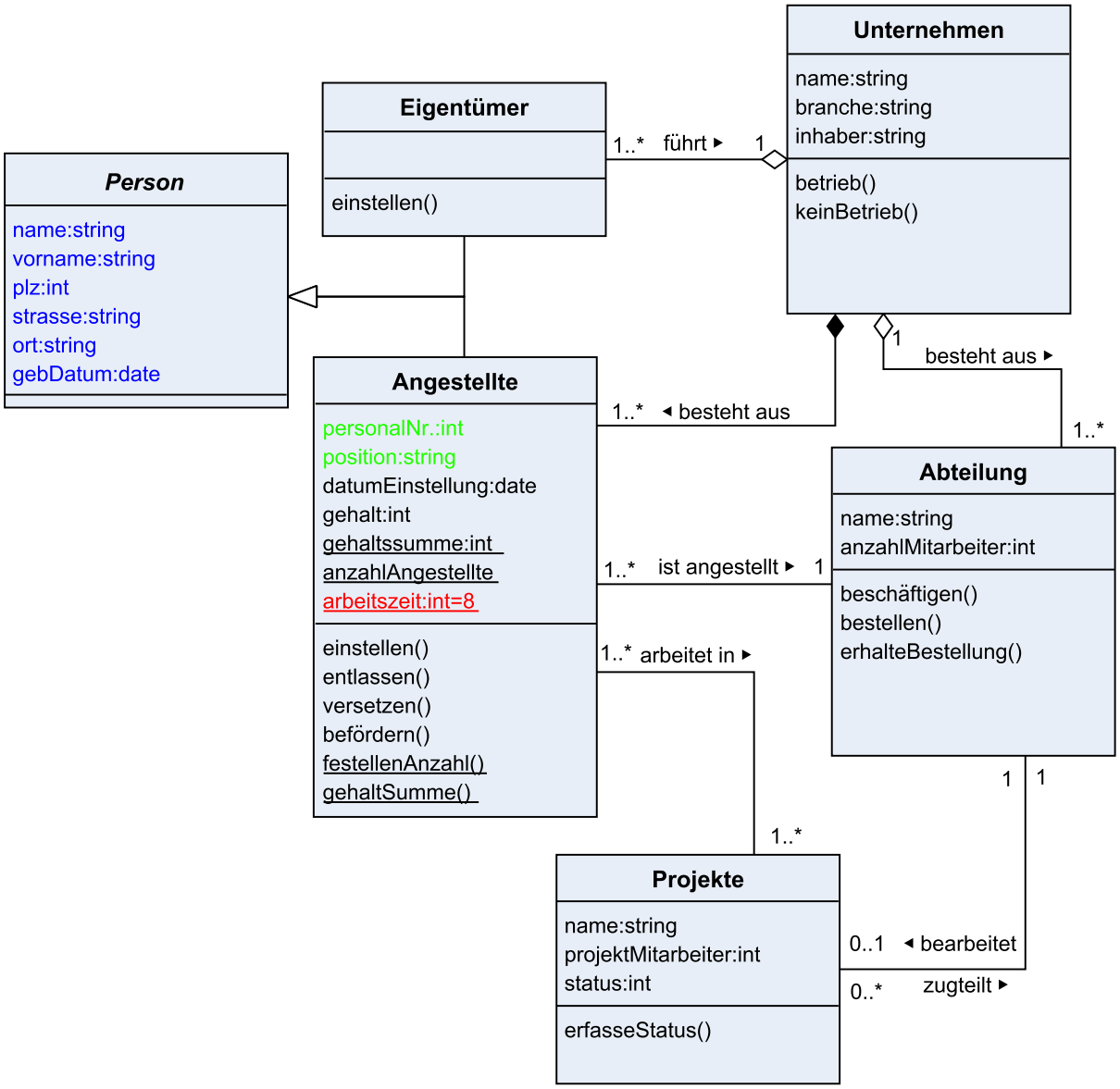


Abbildung 4.11.: Beispielhafte Darstellung eines Klassendiagramms zur Abbildung von Teilen eines Unternehmens nach [Staud, 2010]

Das Objektdiagramm zeichnet sich durch eine starke Exemplarizität aus und hat eher illustrativen Charakter. Aufgrund dessen sind Objektdiagramme auch nicht in der Lage, ein System vollständig zu beschreiben. Zur Verdeutlichung sind sie in einigen Situationen jedoch von großem Nutzen [Hitz u. a., 2005].

Für die Darstellung eines Objektdiagramms sei auf das in **Abbildung 4.11** verwendete Beispiel verwiesen, in dem die Klassen eines Unternehmens dargestellt sind. Das Objektdiagramm in **Abbildung 4.12** besteht aus Instanzen der Klassen **Angestellte, Abteilung** und **Projekte**. Teilweise und nicht in aller Vollständigkeit sind die an die Objekte vererbten Attribute definiert

und mit Attributwerten belegt. Wie in **Abschnitt 4.2.2** bereits herausgearbeitet wurde, gibt es Attributwerte, die als Konstanten angesehen werden. Ein Beispiel könnte hier etwa das rot markierte Attribut *arbeitszeit:int=8* eines Angestellten darstellen, das innerhalb der Klasse **Angestellte** im Klassendiagramm aus **Abbildung 4.11** definiert wurde und das demnach für alle Instanzen dieser Klasse gültig ist. Um die Vererbungshierarchie zu verdeutlichen, sind in dem Objekt *mitarbeiter:Angestellte* einige der Attribute aus der abstrakten Klasse **Person** und einige aus der Klasse **Angestellte** entnommen worden und farblich hervorgehoben.

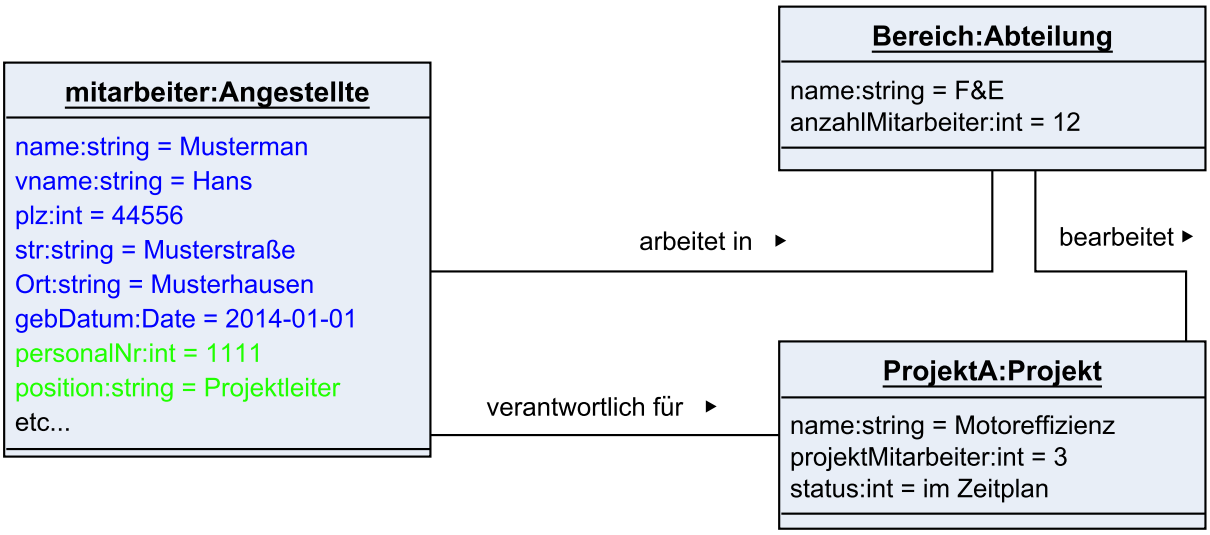


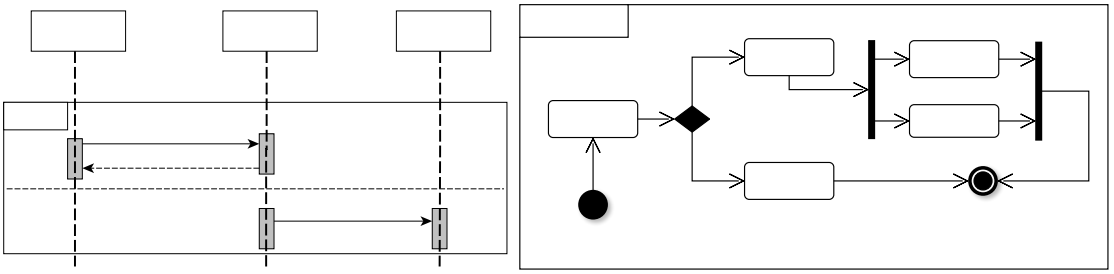
Abbildung 4.12.: Beispielhafte Darstellung eines Objektdiagramms

4.3.2. Verhaltensdiagramme

Die bisher vorgestellten Notationen und Diagrammtypen stellen ein Produktionssystem und dessen Prozesse nur statisch dar und verdeutlichen den Aufbau sowie eine Menge von Schnittstellen zwischen Objekten und Klassen. Das ist nötig, um einen geeigneten und umfassenden Blick auf die Struktur des Produktionssystems zu erlangen. Nun ist es offensichtlich, dass erst durch die Interaktion und das Zusammenspiel von Systemelementen ein interner Ablauf entstehen kann. Ohne einen festgelegten, dynamischen Ablauf wäre ein System gänzlich nutzlos, da, wie in **Abschnitt 2.1.1** eingangs erläutert wurde, ein System dadurch definiert ist, dass die konstituierenden Komponenten in gegenseitiger, organisierter Wechselbeziehung stehen und im Hinblick auf die Erreichung eines logischen und zielgerichteten Resultats zusammenarbeiten. Um diese Dynamik in einem System darzustellen, stellt die UML Verhaltensdiagramme zur Verfügung. Die Verhaltensdiagramme dienen der Darstellung von Verhaltensspezifikation eines Systems aus verschiedensten Blickwinkeln und erlauben in Summe eine vollständige

Beschreibung und Darstellung der gesamten Dynamik von Produktionsprozessen [Oestereich, 2009; Rumpe, 2011; Forbig, 2007].

In dieser Arbeit kommen zwei Arten der Verhaltensdiagramme zum Einsatz, die in **Abbildung 4.13** beispielhaft dargestellt sind: Zum einen in (a) das Sequenzdiagramm und zum anderen in (b) das Zustandsdiagramm. Beide Diagramme stellen das Verhalten von Systemen jeweils aus verschiedenen Blickwinkeln dar und betonen oder vernachlässigen bestimmte Verhaltensaspekte [Rupp u. a., 2012].



(a) Beispielhafte Darstellung des Sequenzdiagramms der UML

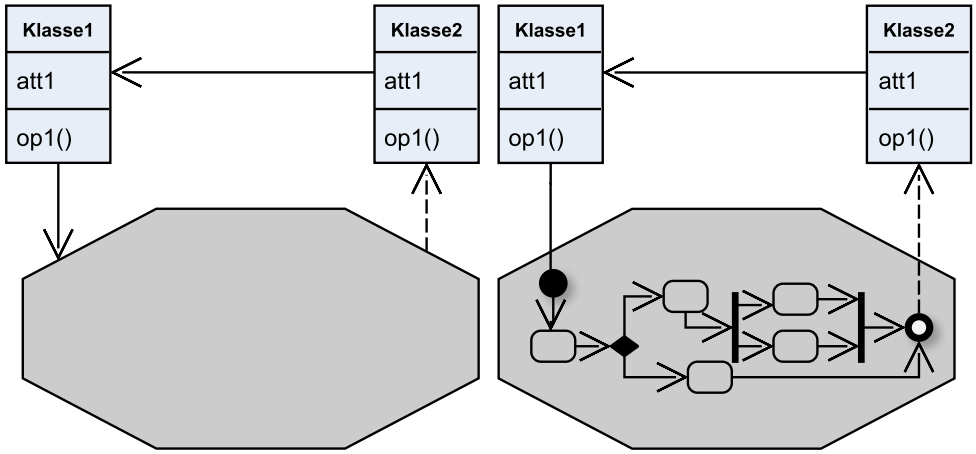
(b) Beispielhafte Darstellung des Zustandsdiagramms der UML

Abbildung 4.13.: Zwei Verhaltensdiagramme der UML nach Rupp u. a. [2012]

Das Sequenzdiagramm

Um die Zusammenarbeit und Kommunikation von Objekten und Klassen möglichst fassbar darzustellen, eignen sich in der UML Interaktionsdiagramme wie das Sequenzdiagramm [Balzert u. a., 2011]. Mit einem Sequenzdiagramm lässt sich das Verhalten zwischen Klassen durch grafische Symbole modellieren und durch Textnotation ergänzend beschreiben. Es eignet sich besonders für die Abbildung der Prozesskommunikation (Bestandteil dynamischer und ereignisorientierter Produktionssystem) und daher für die dieser Arbeit zugrundeliegende Modellierungsaufgabe. Durch die Verwendung des Sequenzdiagramms kann das Verhalten in einem Produktionssystem sehr konkret dargestellt werden, sodass die fachliche und logische Korrektheit unter Berufung auf ein solches Diagramm diskutiert werden kann. Das ist besonders nützlich hinsichtlich einer sachdienlichen Vorgabe für eine spätere Implementierung, was letztendlich das Ziel eines Modells für eine Simulationsanwendung darstellt. Weiterhin sind Sequenzdiagramme die meistverwendeten Diagramme unter den Interaktionsdiagrammen und sind anerkannte Hilfsmittel zur Repräsentation der Interaktion zwischen Klassen und Instanzen [Rumpe, 2011; Rupp u. a., 2012].

Die Verhaltensmodellierung basiert auf der Abbildung der Kommunikation zwischen mehreren Klassen. Die Kommunikation erfolgt durch den Datenaustausch mittels Botschaften, die innerhalb einer Klasse zu einer Aktion führen. Eine Aktion umfasst das Aufrufen einer Operation und das Ausführen der entsprechenden Methode, die sich hinter der Operation verbirgt (vgl. **Abschnitt 4.2.6**). Das Sequenzdiagramm stellt ausschließlich die Interaktion zwischen Klassen dar, indem die Interaktion durch den Informationsaustausch zwischen zwei Kommunikationspartner modelliert wird. Der interne Prozess der einzelnen Klassen, ausgelöst durch die eintreffende Botschaft, wird vom Sequenzdiagramm nicht dargestellt und bleibt unberührt. Es interessiert somit nicht das innere Verhalten der Systemkomponenten, sondern das äußere, das sich durch die Input-Output-Beziehungen der einzelnen Klassen beschreiben lässt. Infolgedessen wird die Interaktion der Objekte in einem Produktionssystem durch das Sequenzdiagramm als Blackbox modelliert, wie es in **Abbildung 4.14 (a)** dargestellt ist [Hitz u. a., 2005; Rupp u. a., 2012].



(a) Blackbox (b) Whitebox

Abbildung 4.14.: Beispielhafte Darstellung der Interpretation einer Blackbox (a) und einer Whitebox (b)

Die Grundelemente eines Sequenzdiagramms sind die verschiedenen Klassen in der Rolle der Kommunikationspartner und die Nachrichten zur Darstellung der Kommunikation an sich. Nachrichten werden aufgrund einer Sendeaktion von einem Sendeobjekt zu einem oder mehreren Empfängerobjekten geschickt und lösen aufseiten der Empfänger ein Ereignis aus [Oestereich, 2009]. Klassen werden durch das übliche Notationselement, dem Rechteck, dargestellt. Zur weiteren Darstellung wird an das Rechteck eine Lebenslinie gezeichnet, die weiterhin im Verlaufe des Diagramms für die Klasse respektive für den Kommunikationspartner steht. Nachrichten werden durch gerichtete Kanten mit einem Pfeil dargestellt, der sich von der Lebenslinie des Senders zu der des Empfängers erstreckt. Die Reaktion eines Nachrichtenempfängers wird

analog dazu durch einen gestrichelten Pfeil dargestellt [Balzert u. a., 2011; Oestereich, 2009].

Abbildung 4.15 (a) zeigt den Aufbau eines Sequenzdiagramms mit allen Notationselementen, die für die Modellierung eines Sequenzdiagramms erforderlich sind.

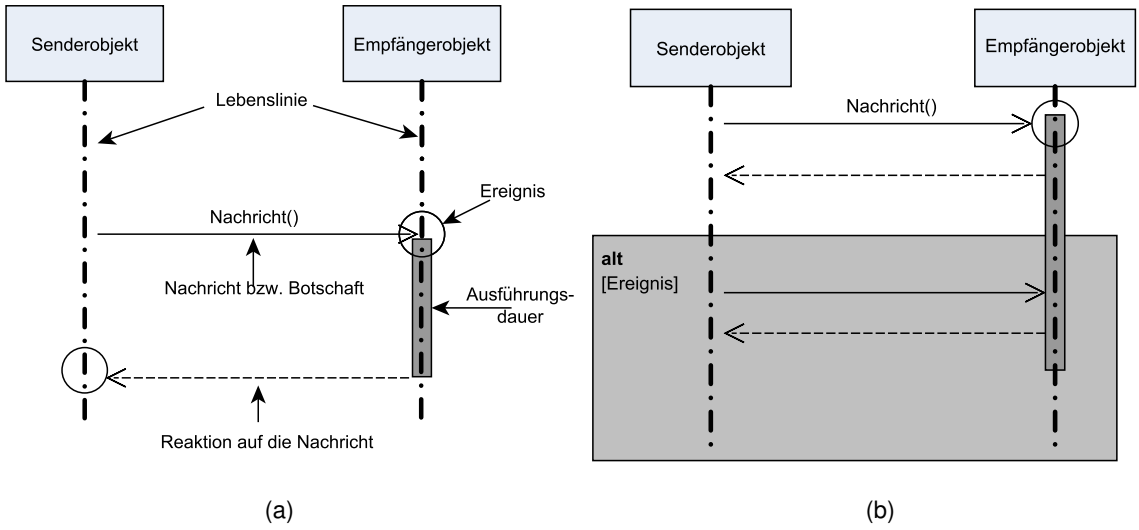


Abbildung 4.15.: Darstellung der Notationselemente von Sequenzdiagrammen

In Sequenzdiagrammen besteht obendrein die Möglichkeit, Alternativabläufe darzustellen (s. **Abbildung 4.15 (b)**). Alternativabläufe werden durch das Einfügen eines Rechtecks realisiert, das zur besseren Anschauung farblich hervorgehoben werden kann. In die linke obere Ecke dieses Rechtecks wird der Operator für den Alternativablauf eingetragen, der die Art des alternativen Ablaufs kennzeichnet. Das bedingte Ereignis, das zum Alternativablauf führt, wird in eckigen Klammer an den Operator angefügt. Es gibt verschieden vordefinierte Operatoren, die der **Tabelle 4.4** zu entnehmen sind [Oestereich, 2009; Hitz u. a., 2005].

Operatoren	**Bedeutung**
alt [Bedingung]	Verzweigung in einen anderen Ablauf, falls die Bedingung eintritt
loop (until, while) [Bedingung]	Iterative Aktion / Schleifenbedingung
break [Bedingung]	Beendet eine Schleife
par [Bedingung]	Nebenläufige Teilsequenz

Tabelle 4.4.: Vordefinierte Operatoren für Verzweigungen im Sequenzdiagramm

In einem Sequenzdiagramm werden darüber hinaus zwei Darstellungsdimensionen beschrieben: Die horizontale Dimension repräsentiert die beteiligten Interaktionspartner, die vertikale Dimension entspricht einer Zeitachse. Durch die zweite Dimension im Sequenzdiagramm lässt sich die Reihenfolge der auftretenden Ereignisse erkennen [Hitz u. a., 2005].

Die Ausführungsdauer in einem Sequenzdiagramm repräsentiert eine Periode, in der das Empfängerobjekt direkt oder indirekt auf eine Operation oder Botschaft reagiert. Dargestellt durch ein graues Rechteck, wir somit suggeriert, dass durch eine eingehende Botschaft ein interner Prozess ausgeführt wird. Der interne Prozess lässt sich dann durch das Zustandsdiagramm darstellen [Rupp u. a., 2012; Hitz u. a., 2005].

Das Zustandsdiagramm

Wie in **Abschnitt 2.2.2** bereits ausgeführt wurde, ist die industrielle Produktion stark durch die Anwendung der ereignisorientierten Simulation geprägt. Diese Art der Simulation baut auf Modellen auf, die ereignisorientiert sind. Dieser Typus Modell bildet das dynamische Verhalten von Systemen ab, indem durch das Eintreten von internen und externen Ereignissen Zustandsänderungen ausgelöst werden [März u. a., 2011]. Ein solcher Sachverhalt wird allgemein durch ein spezielles Modell dargestellt, das als endlicher Zustandsautomat bezeichnet wird [Fischer u. Ahrens, 1996]. Für die Darstellung eines endlichen Zustandsautomaten gibt es in der UML ein eigens entwickeltes Modell: das Zustandsdiagramm. In diesem Diagramm werden Zustandsübergänge als Reaktion auf eintretenden Ereignissen modelliert [Oestereich, 2009; Rupp u. a., 2012]. Demnach erfolgt eine Darstellung der inneren Wirkmechanismen einzelner Objekte aus Zustand- und Ereignisperspektive, weshalb das Zustandsdiagramm das Produktionssystem, im Gegensatz zum Sequenzdiagramm, als Whitebox darstellt (s. **Abbildung 4.14 (b)**)[Rumpe, 2011].

Ereignisorientierte Systeme bestehen aus Zuständen, Zustandsübergängen (sog. Transitionen) sowie aus Ereignissen. Zustandsdiagramme beschreiben, welche Zustände die Objekte im Laufe des dynamischen Verhaltens des Systems einnehmen können und durch welche Ereignisse die verschiedenen Zustände eintreten bzw. durch welche Ereignisse ein Objekt in einen nächsten Zustand überführt werden kann [Oestereich, 2009]. Zustände ändern sich durch exogene Stimuli, die Ereignisse hervorrufen, auf die ein Objekt situationsspezifisch reagiert. Gleichwohl vermag die Reaktion je nach Rahmenbedingungen verschieden zu sein [Balzert u. a., 2011; Hitz u. a., 2005].

Zustände sind in der UML *Aktivitäten*, in denen ein Objekt ein bestimmtes Zeitintervall verweilt. In diesem Zustand der Aktivität hält sich das Objekt so lange auf, bis die Aktivität durch Eintreten eines Ereignisses beendet wird. Ein Ereignis manifestiert einen Zustandsübergang, der in der UML als *Transition* bezeichnet wird. Transitionen sind von einem Zeitverbrauch

abstrahiert, besitzen im Gegensatz zu Aktivitäten folglich keine Dauer [Balzert u. a., 2011; Forbig, 2007].Ereignisse, die eine Transition initiieren könnten, sind in **Tabelle 4.5** aufgelistet.

Trigger-Ereignisse
· Eine Bedingung die wahr wird
· Eintreffen einer Botschaft (Aufruf einer Operation)
· Beenden einer Aktivität
· Eine verstrichene Zeit
· Eintreten eines bestimmten Zeitpunktes

Tabelle 4.5.: Potenzielle Ereignisse, die zu einer Transition führen. In Anlehnung an [Balzert u. a., 2011]

Die Notation der UML stützt sich in Bezug auf das Zustandsdiagramm auf eine Verallgemeinerung der Darstellung von endlichen Automaten [Rumpe, 2011]. Zustände werden durch Knoten in Form von abgerundeten Rechtecken dargestellt. Transitionen werden durch einen Pfeil (gerichtete Kante) zwischen zwei Zuständen (Rechtecken) visualisiert, der zur Beschreibung des Zustandsüberganges entsprechend beschriftet werden kann [Oestereich, 2009]. An Transitionen auslösende Ereignisse mag eine Bedingung geknüpft sein, damit es überhaupt zu einer Zustandsänderung kommt. Eine Bedingung könnte beispielsweise `MaterialvorratVoll=true` sein, wobei im Vorfeld eine Abfrage über den Vorrat eines Materials getätigt wurde. Sobald die Bedingung erfüllt ist, wird die Transition geschaltet[Hitz u. a., 2005].

Es gibt verschiedene Symbole, die zum Einsatz kommen, wenn es um die Darstellung einer Zusammenführung von Transitionen, Teilung von Transitionen oder Endscheidung für bedingte Transitionen geht. In **Abbildung 4.16** sind die erwähnten Notationssymbole dargestellt.

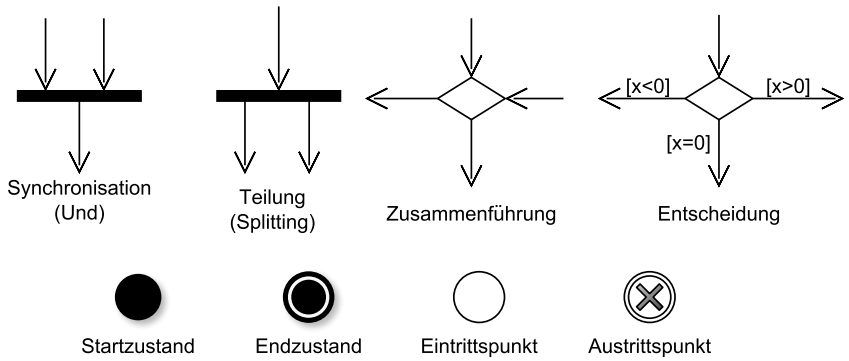


Abbildung 4.16.: Notation von Kontrollknoten und deren Bedeutung für das Zustandsdiagramm in Anlehnung an [Rupp u. a., 2012]

Die Zusammenführung wird verwendet, wenn man Transitionen zusammenführen bzw. aufteilen möchte. Soll bspw. eine Zweigstelle modelliert werden, bei der die Auswahl einer auszuführenden Transition vom Ergebnis einer vorherigen Aktion oder Zustandsänderung abhängt, wird ein Entscheidungsknoten benötigt. Die eingehende Transitionen wird auf Grundlage einer Bedingung in eine oder mehrere Transitionen zusammengefügt bzw. aufgeteilt. Die Folgetransition wird durch die aktuellen Werte der anliegenden Variablen am Entscheidungsknoten bestimmt. Um eingehende Transitionen in mehrere parallel laufende Transitionen aufzuteilen, wird die Teilung verwendet. Sollen die eingehenden Transitionen im umgekehrten Fall zusammengefügt werden, bedient man sich der Synchronisation. An einer Teilung bzw. Synchronisation sind keine Trigger oder Bedingungen geknüpft, da hier parallele Abläufe gestartet bzw. beendet werden. Weiterhin müssen Eintritts- und Austrittspunkt von des Zustandsdiagrammes innerhalb eines Modells kenntlich gemacht werden Rupp u. a. [2012]; Hitz u. a. [2005]; Oestereich [2009].

Ein Übergang in einen Folgezustand kann gleichzeitig ein Trigger-Ereignis für eine Aktivität darstellen. Ist das der Fall, wird dies durch *entry/Aktivität()* definiert. Um Aktivität innerhalb eines Zustandes auszuführen, wird der Präfix *do* wie folgt eingesetzt: *do/Aktivität()*. Do führt eine Aktivität dabei solange aus, wie sich das Objekt in dem entsprechenden Zustand befindet. Tritt das gewünschte Ergebnis der Aktivität ein, kann das Objekt den Zustand verlassen. Soll eine Aktivität in dem Moment stattfinden, in dem ein Objekt den Zustand verlässt, so wird der Präfix *exit* verwendet [Oestereich, 2009; Hitz u. a., 2005].

Wie ein Zustandsdiagramm unter Verwendung der beschriebenen Notation aussehen könnte, zeigt **Abbildung 4.17**.

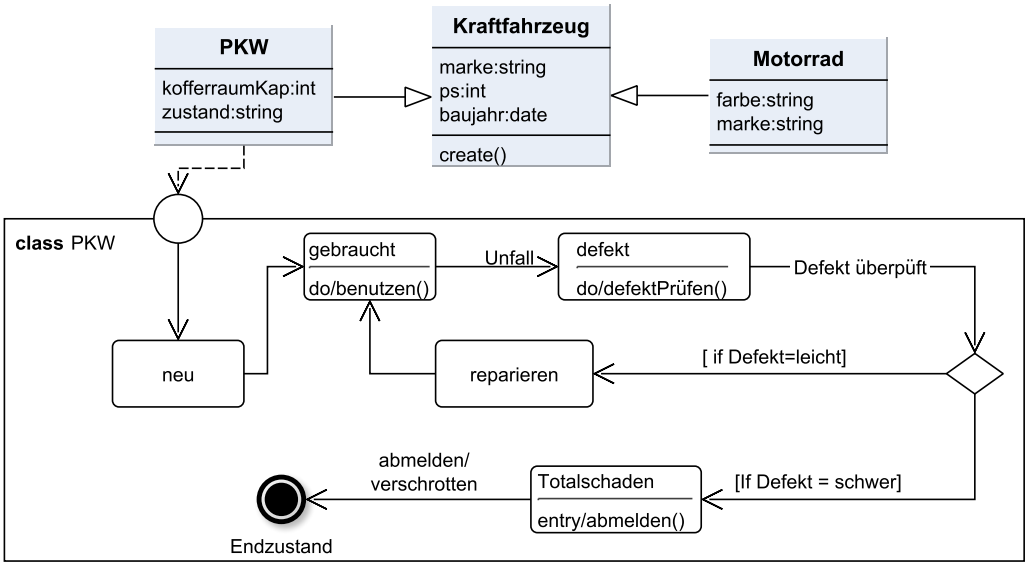
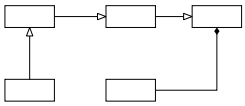
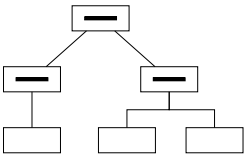
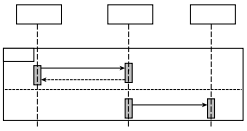


Abbildung 4.17.: Abbildung eines Zustandsdiagramms nach [Rupp u. a., 2012]

Vorteile beschriebener Diagrammtypen

In den zurückliegenden Abschnitten wurden einige der Diagrammtypen der UML vorgestellt, die unter Berücksichtigung der Aufgabenstellung und für die adäquate Repräsentation eines Produktionssystems und dessen Prozesse am geeignetsten erscheinen. Ziel war es, den Prozess der Erstellung und den Nutzen der einzelnen Diagrammtypen hervorzuheben und auf deren Vorzuge und Stärken aufmerksam zu machen. In der nachstehenden **Tabelle 4.6** sind die Diagramme bezüglich dieser Aspekte noch einmal zusammenfassend aufgeführt.

Diagrammtyp	**Diese zentrale Frage beantwortet das Diagramm**	**Stärken**
Klassendiagramm 	Aus welchen Klassen besteht mein System? Und wie stehen die Klassen untereinander in Beziehung?	Beschreibt die statische Struktur des zu entwerfenden oder – besteht das Produktionssystem bereits – des abzubildenden Systems. Enthält alle relevanten Strukturzusammenhänge und Datentypen. Bildet die Brücke zu den dynamischen Diagrammen.
Objektdiagramm 	Welche innere Struktur besitzt mein System zu einem bestimmten Zeitpunkt zur Laufzeit? (Klassendiagrammschnapschuss).	Zeigt Objekte der Klassen und deren Attributbelegungen zu einem bestimmten Systemzeitpunkt. Dient nur der Veranschaulichung und besitzt ein geringfügig höheres Detailniveau als das Klassendiagramm. Sehr gute Darstellung von Mengenverhältnissen.
Sequenzdiagramm 	Wer tritt mit am Beziehungstyp beteiligten Klassenobjekten in Verbindung und tauscht mit wem welche Informationen in welcher Reihenfolge aus?	Stellt den zeitlichen Ablauf des Informationsaustausches zwischen den Kommunikationspartnern dar. Schachtelung und Flusssteuerung (Bedingungen, Schleifen, Verzweigungen) möglich.

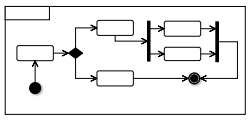
Zustandsdiagramm 	Welche Zustände können die Objekte bei welchen Ereignissen annehmen? Und in welchem Zustand befinden sie sich?	Präzise Abbildung eines Zustandsmodells mit Zuständen, Ereignissen, Nebenläufigkeit, Bedingungen sowie Ein- und Austrittsaktionen. Darstellung von Verschachtelungen möglich.

Tabelle 4.6.: Vorteile der vorgestellten Diagrammtypen nach [Rupp u. a., 2012]

5. Entwurf des Objektmodells

5.1. Betrachtungsrahmen des Modells

5.1.1. Intention des zu konzeptionierenden Modells

Die vorliegende Arbeit zielt darauf ab, ein einfaches Konzeptmodell für die Modellierung von Produktionssystemen zu entwickeln, das eine allgemeingültige Herangehensweise beschreibt und als Vorlage für das Vorhaben der Modellierung herangezogen werden soll. Ein Modellbildner sollte unter Berufung auf das konzeptionierte Modell nicht bei null beginnen müssen, sondern sich bestimmter Referenzen bedienen und das Modell als Ausgangspunkt verwenden können. Analog zu einem Referenzmodell soll das erstellte Modell demnach eine *best practive* oder *common practice* dokumentieren, sodass das Modell als Vorlage dienen und durch Erweiterung und Anpassung auf spezifische Situation adaptiert werden kann.

Durch das entstehende Modell soll gewährleistet werden, dass Produktionsprozesse unabhängig davon, ob die Prozesse bereits existieren oder sich in der Planung befinden, effizient und sachdienlich abgebildet und dargestellt werden können. Da jedes Produktionssystem eine spezifische Aufgabe erfüllt und sich im Aufbau und in den enthaltenden Komponenten von anderen differenziert, muss ein solches Modell flexibel an die unternehmensspezifischen Gegebenheiten, an Rahmenbedingungen und an die spezifischen Aufgaben anpassbar sein, die sich aus dem jeweiligen Anwendungskontext ergeben. Das Modell soll die ereignisdiskrete Simulation ermöglichen und muss demnach Veränderungen durch Ereignisse darstellen können (s. Anforderungen in **Kapitel 4.1.1**). Weiterhin soll das Modell in der Lage sein, einen Produktionsprozess mit allen relevanten Eigenschaften abzubilden. Zudem soll eine Modellierungssprache als Grundlage dienen, die von Dauer ist, eine Langzeitverwendung des konzeptionierten Modells garantiert und die Implementation in eine Simulationsanwendung unterstützt.

Um die soeben beschriebenen Intentionen zu erfüllen, soll sich das Modell an den Grundsätzen der Objektorientierung (s. **Kapitel 3.1.3**) orientieren und mithilfe der Werkzeuge einer geeigneten

Modellierungssprache entwickelt werden. In **Kapitel 4** wurden verschiedene Modellierungssprachen gegenübergestellt und an festgelegten Anforderungen evaluiert. Es wurden verschiedene Aspekte herausgearbeitet, die die Auswahl der *Unified Modelling Language* als angemessenes Instrumentarium für die Modellierungsanwendung untermauern, ein erweiterbares Objektmodell für die ereignisdiskrete Simulation zu entwickeln.

Die UML entspricht allen gestellten Anforderungen und bietet zusätzlich einige Vorteile, die mit einer Verwendung als Modellierungssprache für Produktionssysteme einhergehen. Zum einen stellt die UML einen Standard in der Datenmodellierung dar und komplettiert demnach durch die Verwendung im Bereiche der Prozessmodellierung ein umfassendes Werkzeug zur Unternehmensmodellierung. Des Weiteren – auf den ersten Punkt aufbauend – ist die UML im Bereich der Softwareentwicklung ein anerkanntes Werkzeug zur Modellierung von Programmen und Systemen und erleichtert eine Implementierung des erstellten Modells in einen Programmcode.

Das in dieser Arbeit erstellte Modell erhebt aufgrund der Verwendung als Referenzmodell nicht den Anspruch, vollständig zu sein. Es präsentiert das Vorgehen der objektorientierten Modellierung mit der Unified Modelling Language und zeigt die weitreichenden Möglichkeiten dieser Modellierungssprache auf. Ziel der folgenden Kapitel ist, die im ersten Teil dieser Arbeit beschriebenen theoretischen Grundlagen umzusetzen und deren Umsetzung zur Entwicklung eines Modells anschaulich und verständlich zu präsentieren. Zunächst erfolgt eine kurze Beschreibung der Ausgangslage, die für die Modellierung des Produktionssystems angenommen wird. Anschließend wird ein Gesamtprozess eines Produktionssystems in einem Unternehmen beschrieben und zeitgleich auf die Identifizierung der wichtigsten Komponenten hin untersucht. Nachdem die Komponenten bestimmt und der Produktionsprozess beschrieben worden sind, erfolgt die endgültige Darstellung des Produktionsprozesses in einem objektorientierten Modell, das sukzessive aus den Diagrammtypen der UML, vorgestellt in **Kapitel 4.3**, aufgebaut werden soll.

5.1.2. Darstellung der Ausgangslage

In den nachfolgenden Abschnitten dieser Arbeit soll ein zuvor nicht real existierendes Produktionssystem mit essenziellen Komponenten beschrieben und in ein dafür geeignetes und leicht erweiterbares Objektmodell überführt werden. Da es sich um ein Referenzmodell handelt, soll das der Modellierung zugrunde liegende Produktionssystem ein homomorphes Abbild (vgl.

Abschnitt 3.1.2) eines realen Produktionssystem sein, das üblicherweise in Unternehmen vorzufinden ist. Für die Modellierung des Produktionssystems werden somit nicht alle Details der Produktionsprozesse und der Peripherie berücksichtigt. Das Produktionssystem besteht demnach aus Komponenten, die Grundbausteine von Produktionsprozessen darstellen, die je nach Bedarf repliziert werden können, um ein umfangreicheres Produktionssystem zu modellieren.

Wie in **Kapitel 2.1** beschrieben, setzt sich ein industrielles Produktionssystem aus technischen, logistischen und unterstützenden Prozessen zusammen. In dieser Arbeit finden die unterstützenden Prozesse keine Berücksichtigung. Es wird sich ausschließlich auf die logistischen und technischen Prozesse eines Produktionssystems beschränkt, wobei der Fokus auf den technischen Prozessen liegt. Technische Prozesse setzen sich aus technologischen und produktorientierten Prozessen zusammen. Die technologischen Prozesse umfassen Fertigungsprozess und die produktorientierten Montageprozesse von Produkten. Die strikte Trennung beider Bereiche erlaubt es, Fertigungs- und Montageprozesse auch getrennt von einander zu modellieren.

Fertigungsprozesse stellen die primären Herstellungsprozesse von Teilen dar, die bei Bedarf in Eigenfertigung hergestellt werden und zur Fertigung von Endprodukten notwendigerweise vorhanden sein müssen. Der produktorientierte Bereich umfasst die Montagearbeiten von Baugruppen zu Endprodukten, die sich aus zuvor hergestellten Teilen oder bestellten Bauteilen zusammensetzen. Letztgenannte müssen von einem externen Dienstleister bezogen werden, da diese nicht in Eigenfertigung produziert werden können. Im Falle der Eigenfertigung müssen für die Herstellung der Bauteile Rohstoffe zur Verfügung stehen, die je nach Vorrat ebenfalls extern bezogen werden müssen.

Überdies wird das zugrunde liegende Produktionssystem als ein optimales System verstanden. Das bedeutet: Jegliche Ressourcen zur Unterstützung der einzelnen Produktionsprozesse sind in beständiger Menge verfügbar. Logistische Prozesse spielen wie schon bemerkt keine Rolle für die Modellierung in dieser Arbeit. In einem industriellen Produktionssystem, das ein soziotechnisches System darstellt (vgl. **Kapitel 2.1**), in dem der Mensch als Personal eine wichtige Rolle spielt, gelten Ressourcen wie das Personal, Betriebs- und Hilfsstoffe demnach als ein unerschöpfliches Reservoir, das keiner Planung und Modellierung bedarf.

5.2. Objektorientierte Darstellung des Produktionsprozesses

5.2.1. Vorbereitung und Vorgehen

Bei der objektorientierten Analyse geht es primär darum, ein Fachkonzept zu entwickeln, indem Objekte der realen Welt, ohne Berücksichtigung von Relationen modelliert werden. Die objektorientierte Analyse stellt also einen konzeptuellen Entwurf des Produktionssystems in Form eines Analysemodells dar und identifiziert auf einer ersten Abstraktionsebene Realweltobjekte als späterer Objekte des Modells. Die identifizierten Realweltobjekte gilt es schließlich, durch geeignete Modellierung und Abstraktion als Entitäten in einem objektorientierten Modell darzustellen. Ein solches Analysemodell wird von Fischer als ein entscheidendes Kriterium für eine erfolgreiche Modellierung angesehen. In diesem Zusammenhang wird von ihm postuliert, dass ein Analysemodell eines zu modellierenden Sachverhalts unabdingbar zur Erreichung eines holistischen Abbilds der Realität sei [Fischer, 1995].

Eine der ersten Aufgaben, die einem Modellierer nun zuteilwird, ist die, sich Gedanken über die Entwicklung von Objekten zu machen. Die Schwierigkeit bei dieser Aufgabe liegt darin, dass der Modellierer Realweltobjekte nach festen Regeln als relevante Objekte identifizieren und schließlich in sein Modell integrieren muss. Es stellt sich dann die Frage, welche Klassen sinnvollerweise gebildet werden und welche Objekte in einem Produktionsprozess überhaupt modelliert werden sollen. Welche Realweltobjekte letztendlich relevant sind, hängt, wie bereits offengelegt wurde, von der Problemstellung und dem Ziel des Modells ab und ist keine triviale Aufgabe, wie Oestereich ebenfalls zum Ausdruck bringt.

> „Etwas einfach zu machen ist nicht einfach, es bedarf der geschickten Abstraktion vom Komplexen zum Wesentlichen" [Oestereich, 2009].

Um die Frage der Klassenbildung zu beantworten, benötigt man letztendlich ein Vorgehen, um auf sinnvolle Weise Klassen zu bilden. Staud beschreibt ein solches Vorgehen und postuliert die Notwendigkeit folgender Schritte, die zu Beginn einer Modellierung a priori durchzuführen seien. In diesen Schritten ist es das primäre Ziel, Gegenstände der realen Welt – innerhalb des betrachtenden Produktionssystems – als notwendige Objekte zu identifizieren. Den identifizierten Objekten werden dann Attribute und Operationen zugeordnet, und anschließend in gebildeten Klassen aggregiert [Staud, 2010]. Wie der Prozess der Objektidentifizierung und Klassenbildung abläuft, soll durch **Abbildung 5.1** deutlich werden.

Ein System setzt sich, wie in **Abschnitt 2.1.1** beschrieben, aus einem statischen und aus

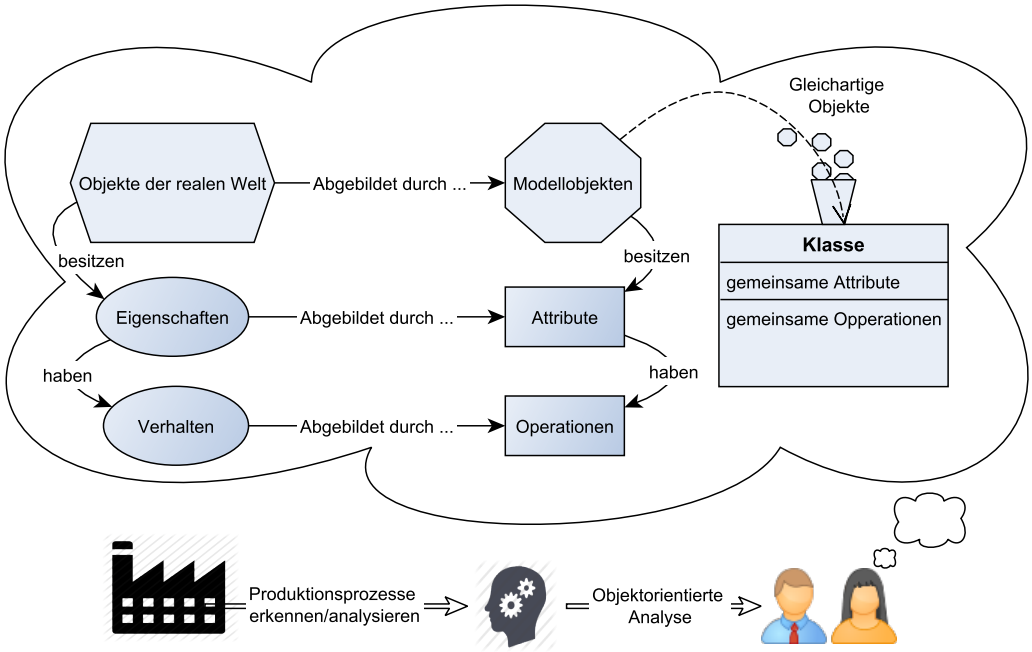


Abbildung 5.1.: Modell der objektorientierten Analyse zur Bildung von Objekten und Klassen

einem dynamischen Anteil zusammen. Demgemäß erfolgt auf Grundlage der erarbeiteten Klassen eine Phase, in der die Modellierung von Relationen und von Verhaltensaspekten der erstellten Klassen durchgeführt wird. In dieser Phase der Modellentwicklung, genannt objektorientierter Entwurf (vgl. **Abschnitt 3.1.1**), erfolgt die Komplementierung der Struktur- und Verhaltensdiagramme. Beide Teilmodelle (Struktur- und Verhaltensdiagramme) müssen sinnvoll modelliert werden. Die vorliegende Arbeit orientiert sich bei dieser Aussage an den Ausführungen von Balzert u. a., der für eine erfolgreiche Modellierung das Zusammenwirken von statischen und dynamischen Modell für unabdingbar hält. Balzert u. a. indizieren, dass für die Validierung des statischen Modells das dynamische Modell benötigt wird und vice versa [Balzert u. a., 2011].

Aus dem so ebenen beschriebenen Vorgehen wird deutlich, dass sich in einem ersten Schritt auf die statischen Systemaspekte im zu konzeptionierenden Modell konzentriert wird und anschließend der Schwerpunkt auf den dynamischen Systemaspekten liegt. Um das dynamische Modell zu erstellen, wird immer wieder das statische Modell referenziert. Das Vorgehen im weiteren Verlauf lässt sich in Konsens zu den vorgestellten Diagrammtypen der UML in **Abschnitt 4.3** wie in der folgenden **Tabelle 5.1** darstellen.

Schritte in der Vorgehensweise zur Modellierung eines Produktionssystems
· Die relevanten Objekte eines Produktionsprozesses werden identifiziert und attribuiert. Es werden auf dieser Grundlage Klassen gebildet. → Klassendiagramm · Die Klassen werden mit Attributen und Operationen belegt und es wird eine Vererbungsstruktur identifiziert. → Klassendiagramm · Mögliche Objekte der erstellten Klassen werden zu einem bestimmten Systemzeitpunkt aufgezeigt. → Objektdiagramm · Die Interaktion und die Kommunikation der Klassen untereinander wird analysiert und modelliert. → Sequenzdiagramm · Die Interaktionen der Klassen werden durch die Modellierung der Zustände der instantiierten Objekte verdeutlicht. Zustandsübergange werden mit den korrespondierenden Ereignissen modelliert. → Zustandsdiagramm

Tabelle 5.1.: Vorgehensweise zur objektorientierten Modellierung eines Produktionssystems mit der UML

5.2.2. Beschreibung des Produktionsprozesses

Um nun den Prozesse der Entwicklung eines objektorientierten Modells für die ereignisorientierte Simulation aufzuzeigen, soll ein einfaches Produktionssystem beschrieben werden. Die Fertigungstiefe bleibt unberücksichtigt, sodass jeweils ein Fertigungs- und ein Montageprozess ausreichen soll. Der Zweck eines Produktionssystems ist, wie in **Abschnitt 2.1.1** formuliert, im Allgemeinen die Erfüllung eines Kundenauftrages. In der industriellen Produktion müssen in Reaktion auf einen eingehenden Auftrag Produkte gefertigt werden. Jedes Produkt setzt sich aus verschiedenen Materialien zusammen. Material ist in dieser Arbeit ein umfassender Begriff für Fertigteile, Halbzeuge und Rohstoffe. Gemäß der Separation von Fertigungs- und Montageprozessen muss Material je nach Bedarf der zu fertigenden Produkte gefertigt oder

bestellt werden. Der eigentliche Produktionsprozess, angestoßen durch einen Kundenauftrag, setzt sich aus der Fertigung des eigenen Materials, sprich den Halbzeugen, und aus dem für die Herstellung von Produkten notwendigen Montageprozess zusammen. Die Prozesse in der Montage umfassen das Montieren von Einzelteilen und Baugruppen zu einem Endprodukt.

Die grundlegenden Bestandteile eines Produktionsprozesses sind infolge der zurückliegenden Beschreibung und nach Westkämper *Auftrag, Material, Betriebsmittel* und *Produkt* [Westkämper, 2006]. In **Abbildung 5.2** ist ein Produktionsprozess mit den genannten essenziellen Komponenten schematisch abgebildet.

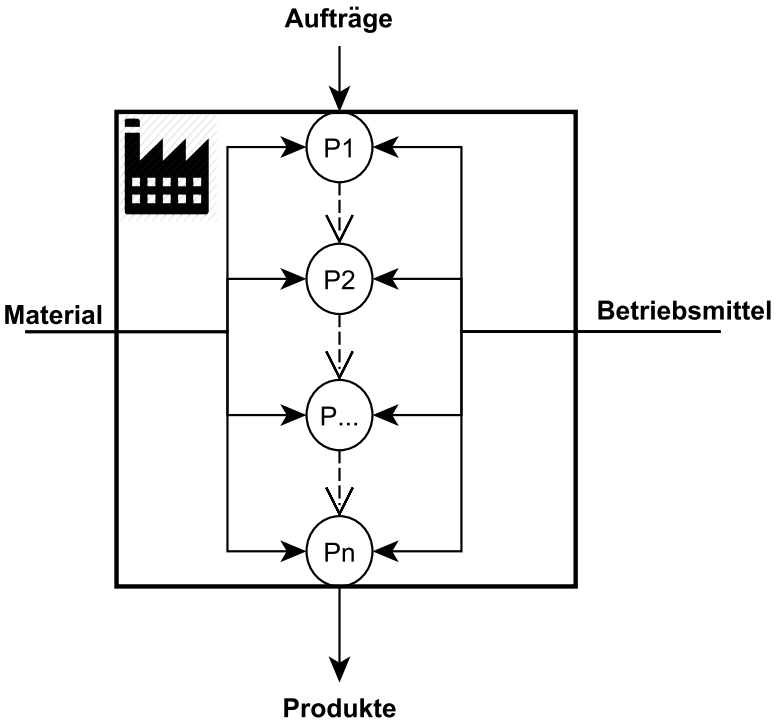


Abbildung 5.2.: Essenzielle Komponenten eines Produktionssystems nach [Westkämper, 2006]

Um ein holistisches Abbild des zu modellierenden Produktionssystems zu erreichen, soll dieses im Folgenden beschrieben werden. Gemäß der erörterten Vorgehensweise zur Bildung von Objektklassen in **Kapitel 5.2.1** sollen hier die wichtigsten grundlegenden Objekte erarbeitet werden. Hilfestellung bieten die soeben erarbeiteten Komponenten eines Produktionssystems. Im weiteren Verlauf werden die einzelnen Objekte dann weiter aggregiert und auf der nächst höheren Abstraktionsebene zu Klassen zusammengefasst. Wie bereits bemerkt wurde, lässt sich vorwegnehmen, dass die entstehenden Klassen den Komponenten von Produktionsprozessen aus **Abbildung 5.2** entsprechen müssen. Die Komponenten stellen essenzielle Bestandteile eines Produktionssystems dar und sind infolgedessen wesentliche Bausteine des zu entwickelnden

Modells. Die folgende textuelle Beschreibung veranschaulicht verbal den Produktionsprozess und führt zur Identifikation der genannten wesentlichen Objekte, die durch eine *kursive* Schriftart hervorgehoben werden.

Der Produktionsprozess

Ein konkreter Anwendungsfall des beispielhaften Produktionssystems ist so aufgebaut, dass ein Kunde eine Bestellung durch einen *Auftrag* aufgibt und die durch diesen Auftrag angestoßenen internen Abläufe des Produktionssystems modelliert werden. Das Produktionssystem soll in der Lage sein, verschiedene *Produkte* zu fertigen, die je nach Bedarf in der Montage aus ihren Bauteilen zusammengesetzt werden. In dieser Arbeit wird sich beispielhaft auf zwei Produkte beschränkt, was jedoch nur Auswirkung auf das erstellte Objektdiagramm hat. Bauteile können in zwei Kategorien klassifiziert werden. Eine Kategorie von Bauteilen soll in Eigenfertigung bedarfsgerecht hergestellt werden können. In Eigenfertigung hergestellte Bauteile werden durch *Halbzeuge* repräsentiert, für deren Herstellung eine *Fertigungsmaschine* im Bereich der Fertigung unterschiedliche *Rohstoffe* benötigt. Rohstoffe werden je nach Bedarf von einer Fertigungsmaschine angefordert, um benötigte Halbzeuge für bestellte Endprodukte herstellen zu können. Eine Fertigungsmaschine, die je nach Kapazität und Belegung fertigen kann, lässt sich durch die Zustände aktiv, frei und defekt beschreiben. Ist die Maschine bspw. frei, ändert ein eintretendes Ereignis, wie etwa das Eintreffen von benötigten Rohstoffen, den Zustand der Maschine. Fertig produzierte Halbzeuge werden einer *Montagemaschine* zur Verfügung gestellt und werden, wenn das resultierende Endprodukt nicht aus anderen Teilen als aus Halbzeugen besteht, unmittelbar zusammengebaut. Analog zu der Fertigungsmaschine lässt sich die Montagemaschine mit den gleichen Zuständen beschreiben. Halbzeuge und Rohstoffe fallen wie bereits bemerkt unter den Begriff Material. Infolgedessen lässt sich eine Klasse *Material* generieren, die Halbzeuge und Rohstoffe beinhaltet und deren Eigenschaften kapselt.

Eine zweite Kategorie Bauteile sollen *Fertigteile* sein, die nicht in Eigenfertigung gefertigt werden können und aus diesem Grund über einen externen Produzenten bestellt werden müssen. Fertigteile zählen ebenfalls wie Halbzeuge und Rohstoffe zur der Klasse Material, werden ebenso wie Halbzeuge einer Montagemaschine im Bereiche der Montage direkt zur Verfügung gestellt und können separat oder in Verbindung mit Halbzeugen zu einem Produkt zusammengesetzt werden, wobei Letzteres die Regel darstellen soll. Alle Materialien, also Halbzeuge, Fertigteile und Rohstoffe, können die Zustände vorrätig oder verbraucht annehmen. Die Verfügbarkeit der jeweils benötigten Ressourcen ist bei jedem Fertigungs- und Montageschritt zu berücksichtigen.

Sollte die Verfügbarkeit nicht gewährleistet sein, so muss auf die benötigten Teile gewartet werden. Es wird davon ausgegangen, dass Fertigteile und Rohstoffe in einem Lager vorrätig sind. Die Informationsfluss orientierten und logistischen Prozesse zwischen Lager und Produktion, die für die Sicherstellung des Vorrats von Rohstoffen und Fertigteilen ablaufen, werden wie bereits bemerkt nicht berücksichtigt, sodass sich auf die Modellierung der Verfügbarkeit von Materialien innerhalb des Produktionssystems beschränkt wird. Erst wenn alle Materialien an den jeweiligen Maschinen verfügbar sind, kann mit der Fertigung bzw. Montage begonnen werden.

Zusammenfassend führt die zurückliegende Beschreibung des Produktionssystems zur Identifizierung folgender Objekte, die die bereits hervorgehobenen essenziellen Bestandteile eines Produktionssystem umfassen. Die Objekte sind: *Auftrag*, *Material* (*Halbzeuge, Fertigteile* und *Rohstoffe*), *Produkt* und *Betriebsmittel* (*Fertigungs-* und *Montagmaschine*).

Stößt ein eingehender Auftrag die internen Prozesse des Produktionssystems an, wird zunächst der Auftrag hinsichtlich der enthaltenen Informationen analysiert. Dafür werden die in der Bestellung gelisteten Positionen selektiert, die Auskunft über die bestellten Produkte geben. Es wird überprüft, ob die einzelnen Produkte bereits durch vorhandene Teile hergestellt werden können oder eben nicht. Im erstgenannten Fall, wenn alle benötigten Teile bereits vorliegen, werden die Teile für die entsprechende Bestellung markiert und reserviert. Hierfür soll eine Auftragsnummer verwendet werden, die zuvor jedem neu eintreffenden Auftrag zugeordnet wird. Werden Teile für die Endfertigung in der Montage benötigt, die nicht verfügbar sind, so wird der Auftrag vorerst zurückgestellt und in eine Liste von wartenden Aufträgen eingetragen.

Sobald Fertigteile, Halbzeuge und Rohstoffe verfügbar sind, wird zunächst überprüft, welche Art von Material in welcher Menge für bereits wartende Aufträge benötigt wird. Anschließend erfolgt eine genaue Zuordnung der Bauteile zu den jeweils zurückgestellten Aufträgen. Jede Bestellung wird bei ihrem Eingang mit einem Zeitstempel versehen, sodass die Fertigteile und Halbzeuge zuerst für die am längsten wartenden Aufträge reserviert werden. Kann ein Auftrag vollständig gefertigt werden, so werden die entsprechenden Bauteile an die Montage übermittelt. Hier werden die herzustellenden Produkte mit ihrer entsprechenden Quantität zunächst der dafür vorgesehenen Bearbeitungsmaschinen zugeteilt. In dem Modell in dieser Arbeit existiert der Einfachheit halber nur eine Montage- und Fertigungsmaschine. Ist eine Maschine belegt, so werden die mit der Auftragsnummer markierten Bauteile in einem Puffer bevorratet, der als Teil der jeweiligen Maschine verstanden wird. Ist die Maschine frei, werden die Produkte gefertigt. Wurden alle Produkte eines Auftrags hergestellt, werden die Produkte kommissioniert und an den Kunden ausgeliefert.

5.2.3. Modellierung der Objekte des Produktionssystems

Um einen ersten Überblick und ein tieferes Verständnis über den soeben beschriebenen Produktionsprozess zu gewinnen, werden die in den zurückliegenden Abschnitten zur Sprache gekommenen Objekte in ihren Klassen modelliert. Die einzelnen Elemente in einem Produktionsprozess, die einer Modellierung und somit einer Repräsentation als Instanzen bedürfen, sind in **Tabelle 5.2** noch einmal zusammengefasst und werden im Folgenden beschrieben und in der Notation der UML modelliert.

Komponenten eines Produktionssystem
· Auftrag
· Material
· Betriebsmittel
· Produkt

Tabelle 5.2.: Realweltobjekte die zu Modellweltobjekten modelliert werden

Auftrag

Der Auftrag ist das Objekt, das die internen Produktionsprozesse in einem System anstößt und festlegt, welche Produkte produziert werden müssen. Da ein Unternehmen ein Produktionssystem aus dem Motiv instituiert, um eine vorher spezifizierte Dienstleistung zu erbringen, die in einen produzierenden Unternehmen zweifellos die Herstellung eines bestimmten Produktes umfasst, kann davon ausgegangen werden, dass die mögliche Folge von Prozessen fest definiert ist, um das Produkt oder die Produkte zu fertigen. Der Kundenauftrag spezifiziert ausschließlich, in welcher Quantität und in welcher Modifikation der Auftraggeber ein bestimmtes Produkt ordern möchte. Um diese Informationen preiszugeben, benötigt ein Objekt Auftrag und die generalisierende Klasse Auftrag entsprechende Attribute und Operationen.

Einem Auftrag lässt sich das Attribut `kundennummer` zuschreiben, das zum Zweck der Identifikation des Kunden dient und eine `auftragsnummer`, um den Auftrag systemintern einzuordnen. Die Auftragsnummer fungiert darüber hinaus als Identifikationsnummer unter den verschiedenen Aufträgen. Ferner benötigt ein Auftragsobjekt einen Zeitstempel, der durch das Attribut `datumBestellung` repräsentiert wird und eine Möglichkeit darstellt, die bestellten Positionen zu erfassen, die durch das Attribut `positionen` wiedergegeben werden. Um auf diverse Daten

zugreifen zu können, besitzt ein Objekt der Klasse Auftrag entsprechende Operationen. Um die Auftragsdaten erfassen zu können, muss es eine Operation geben, die durch ihre programmiersprachenspezifische Methode alle Daten des Auftrags einließt und auswertet. Eine solche Operation könnte bspw. *erfasseAuftragsdaten()* lauten. Wie die programmiertechnische Umsetzung beispielhaft aussehen könnte, ist in **Listing A.1** im **Anhang A.1** zu sehen. Für weitere Aktionen, wie die Interaktion mit anderen Objekten des Produktionssystems und für festgelegte Aktivitäten, muss es weitere Operationen geben. Bis auf die Konstruktor- und Destruktoroperationen, die auch in den weiteren Objekten nicht aufgeführt werden, sind die zusätzlich deklarierten Operationen der Klasse Auftrag verschiedene Operationen für weitere unterschiedliche Aktionen. Die Operationen wurden so bezeichnet, dass deren Bedeutung intuitiv verständlich ist. Um den Rahmen dieser Arbeit nicht zu sprengen, werden die Operation infolgedessen nicht im einzelnen detailliert beschrieben. Wie das Konzept Operation allgemein aufgebaut ist, lässt sich in **Abschnitt 4.2.4** nachlesen.

Die Klasse **Auftrag** ist mit ihren Attributen und Operationen in **Abbildung 5.3 (a)** modelliert. Ein Beispiel für das Realweltobjekt, das durch ein Modellweltobjekt Auftrag repräsentiert wird, ist in **Abbildung 5.3 (b)** zu erkennen.

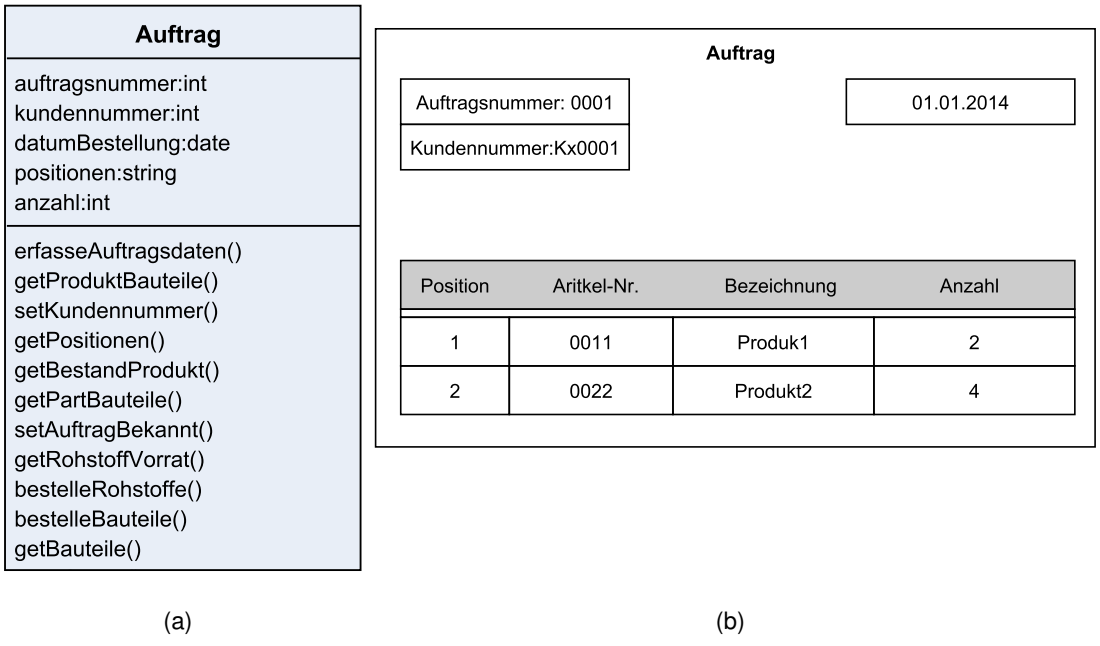


Abbildung 5.3.: Darstellung der Klasse Auftrag und des repräsentierten Realweltobjekts

Die soeben beschriebene Modellierung der Klasse Auftrag beschränkt sich auf die Modellierung mit der grafischen Modellierungssprache UML. Der Vorgang der grafischen Modellierung stellt, wie eingangs der Arbeit bereits erläutert, nur den primären Schritt einer Systemanalyse durch

Modellierung und Simulation dar. Der zweite Schritt ist die Implementierung des grafisch entwickelten Modells in einem objektorientierten Programmcode. Eine konsistente Verwendung der Objektorientierung in den Phasen der Modellbildung und Implementierung erlaubt eine einfache Generierung des Programmcodes. Um die Notationssymbole der UML mit einer objektorientierten Programmiersprache abzubilden, eignet sich die Verwendung der Sprache C++, die in **Abschnitt 3.2.2** vorgestellt wurde. Es gibt natürlich weitere objektorientierte Programmiersprachen wie beispielsweise JAVA, die aufgrund der Offenheit der UML durchaus auch für die Implementierung eines in UML erstellten Modells geeignet wäre. C++ bietet sich jedoch aufgrund des *multi-paradigm*-Ansatzes als Modellierungssprache an. Das bedeutet: C++ legt sich nicht auf einen Programmierstil fest und schreibt keine Richtung vor. Gleiches gilt in Bezug auf die Objektorientierung, deren Prinzipien und Vorgehensweisen durch C++ nicht zwangsläufig vorgeschrieben sind, jedoch durch adäquate Konzepte ausgedrückt werden können [Fischer u. Ahrens, 1996]. Somit bietet C++ ebenso wie die UML einen flexiblen Rahmen für die Implementierung des objektorientierten Modells in eine Simulationsanwendung. Das Vorgehen, wie die Klasse Auftrag in einem Programmcode der vorgestellten Programmiersprache C++ implementiert werden kann, ist im **Anhang A.1** beispielhaft an einigen Attributen und der bereits genannten Operation *erfasseAuftragsdaten()* aufgezeigt.

Produkt

Ein Kundenauftrag bestimmt, wie zuvor erwähnt, die Quantität und Art von bestellten Produkten. Da ein Auftrag demnach eine gewisse Anzahl von Positionen hat, die jeweils verschiedene Produkte repräsentieren, ist eine Instanz der Klasse Auftrag in der Lage, durch eine Botschaft die Klasse Produkt dazu aufzufordern eine Instanz der Klasse Produkt zu instanziieren. Produkte, die für einen Auftrag benötigt werden, können dann in der Produktion gefertigt werden. Ein Produktobjekt steht stellvertretend als Informationsträger von Produkten, die für die Erfüllung eines Auftrags hergestellt werden müssen. Es repräsentiert demnach kein physisches Produkt, sondern ausschließlich einen internen immateriellen Produktionsauftrag. Der Produktionsauftrag wird durch die unterschiedlichen Prozesse in der Produktion mit dem Ergebnis der realen, physischen Produkte erfüllt.

Um ein Produkt zu kennzeichnen, dienen die Attribute `bezeichnung` und `produktnummer`; um es einem Auftrag zu zuordnen, benötigt ein Produktobjekt des Weiteren eine `auftragsnummer`. Die Anzahl (`anzahl`) an bestellten Produkten muss kenntlich gemacht werden, und die Bauteile, aus denen sich ein Produkt zusammensetzt (`partsProdukte`), müssen bestimmbar sein. Für

die Operationen gilt dasselbe, wie für die Operationen der Klasse Auftrag beschrieben wurde. Die Objektklasse **Produkt** ist in **Abbildung 5.4** zu sehen.

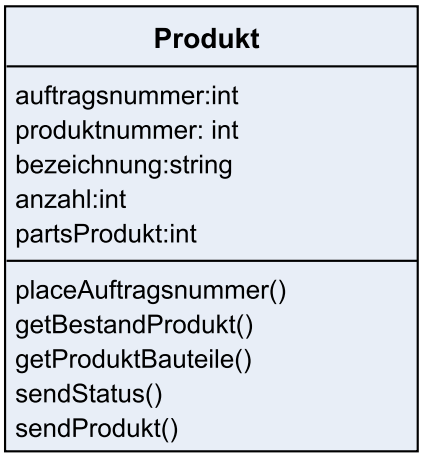


Abbildung 5.4.: Darstellung der Objektklasse Produkt

Material

Die Klasse Material beschreibt alle materiellen Ressourcen, die in einem Produktionssystem im Umlauf sind und am Wertschöpfungsprozess teilhaben. In dem Produktionssystem dieser Arbeit sind Materialien Fertigteile, Halbzeuge und Rohstoffe, die jeweils eine Unterklasse der im Folgenden modellierten abstrakten Klasse *Material* darstellen.

Eine abstrakte Klasse ist wie in **Abschnitt 4.2.8** eine Klasse, die ihre Attribute und Operationen an alle Unterklassen vererbt. Gemäß Definition ist eine abstrakte Klasse nicht in der Lage, Instanzen zu erzeugen, dient also nur dem Zweck, gemeinsame Attribute und Operationen zu kapseln. Attribute, die die abstrakte Klasse Material beschreiben, sind zum einem die `bauteil-` und `rohstoffnummer`, die für die Zuordnung der richtigen Bauteile und Rohstoffe dienlich sind. Um den Vorrat zu reflektieren, dient das Attribut `status`. Da sich jedes Produkt aus unterschiedlichen Teilen zusammensetzt, werden noch Angaben über die Zugehörigkeit von Bauteilen zu einem Produkt benötigt. Gleiches gilt für Rohstoffe, aus denen die Halbzeuge hergestellt werden. Die Abfrage, aus welchen Bauteilen und Rohstoffen ein Produkt bzw. Halbzeug besteht, ermöglicht eine Überprüfung der Verfügbarkeit konstituierender Bauteile und Rohstoffe. Aus den genannten Gründen werden die Attribute `partOfProduct` und `partRohstoff` deklariert. Schließlich gibt es noch ein Attribut `menge`, das die Quantität einer jeweils benötigten Ressource angibt.

Nicht zu vergessen sind die Operationen, die die abstrakte Materialklasse im Hinblick auf eine spätere Systemdynamik enthalten muss. Da die Klasse Material eine umfangreiche Klasse

darstellt, enthält sie auch eine große Menge an Operationen, die analog zu den Operationen in den anderen Klassen dieser Arbeit durch ihre Bezeichnung und durch ihre Präfixe wie *get*, *send* und *bestelle* implizit zu verstehen sind. Darüber hinaus gilt für diese Klasse, wie auch für andere Klassen des zu konzeptionierenden Modells, dass durch eine spätere Modellierung der Systemdynamik der Nutzen der einzelnen Operationen deutlich wird. Die abstrakte Klasse ***Material*** ist mit ihren Unterklassen in **Abbildung 5.5** dargestellt.

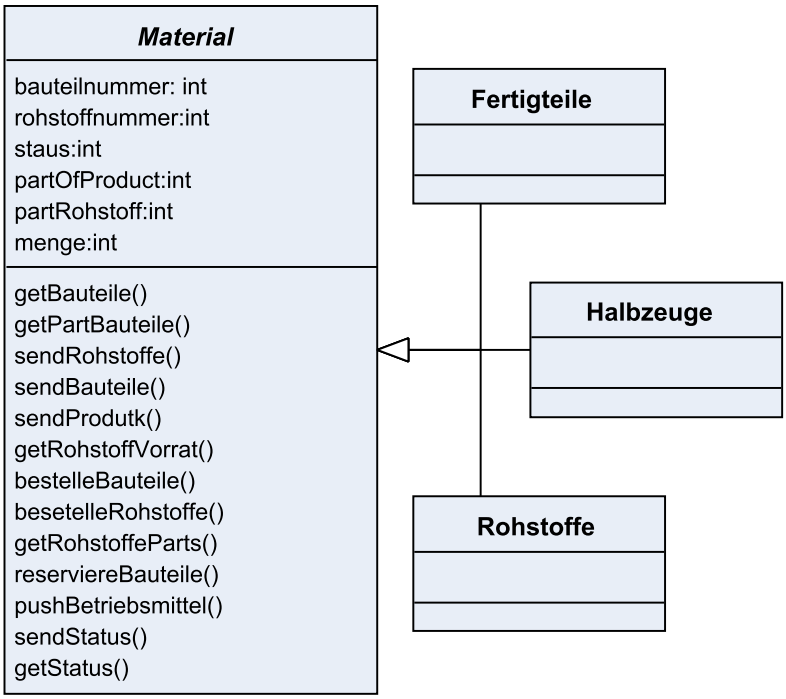


Abbildung 5.5.: Modellierung der abstrakten Klasse Material mit ihren erbenden Klassen

Betriebsmittel

Eine letzte essenzielle Komponente in Produktionssystemen sind Betriebsmittel, die vorhandene technische Einrichtungen repräsentieren, mit denen Material bearbeitet und bewegt wird. Die Klasse Betriebsmittel stellt eine generalisierte Oberklasse dar, die in weitere Klassen spezialisiert werden kann. Das hier angesprochene Prinzip der Generalisierung und Spezialisierung ist in **Kapitel 4.2.5** nachzulesen und fand schon zuvor bei der Klasse Material Anwendung.

Unterklassen der Klasse Betriebsmittel sind die Klassen Montage- und Fertigungsmaschine. Die Klasse Betriebsmittel kann viele Kennwerte in Form von Attributen besitzen, wie z.B. eine *iDNummer*, um ein Betriebsmittel zu identifizieren. Ferner benötigt man Informationen über den Status (Aktiv/Frei) der Betriebsmittel, über den Zustand – um möglich Defekte oder bevorste-

hende Wartungsarbeiten festzustellen – und Informationen über mögliche Bearbeitungsschritte sowie vorhandene Kapazitäten. Operationen, um all die nötigen Informationen zu übermitteln und verschiedene Aktivitäten zu beschreiben, sind ebenfalls erforderlich. Die abstrakte Klasse ***Betriebsmittel*** ist in **Abbildung 5.6** modelliert. Die verschiedenen Betriebsmittel wie Fertigungsmaschine und Montagemaschine sind erbende Klassen der Klasse Betriebsmittel und können durch spezifische Attribute und Operationen ergänzt werden. Je nach Anwendungszweck ist demnach eine differenzierte Darstellung möglich.

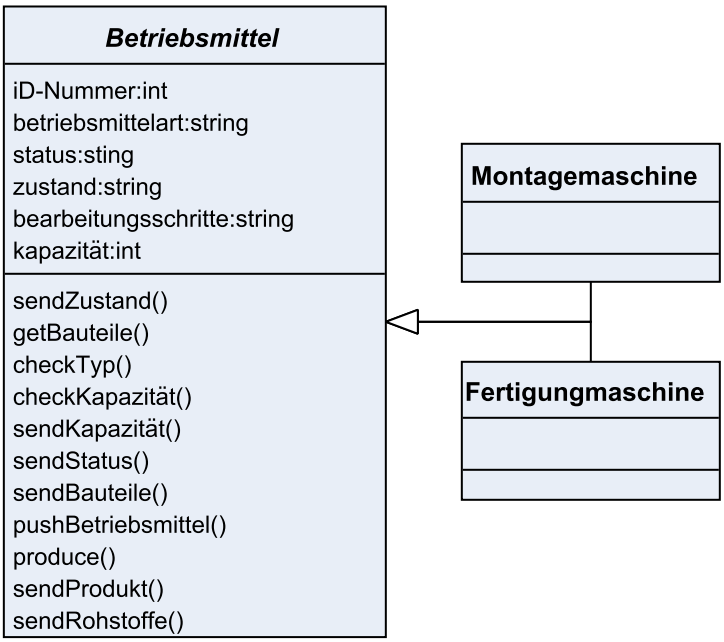


Abbildung 5.6.: Darstellung der abstrakten Klasse Betriebsmittel mit den erbenden Klassen Fertigungs- und Montagemaschine

6. Entwicklung des Modells

6.1. Modellierung des objektorientierten Modells

6.1.1. Das Klassendiagramm

Nachdem nun die am Produktionsprozess beteiligten Komponenten herausgearbeitet und modelliert wurden, kann das objektorientierte Modell mit den in **Kapitel 4.2** eingeführten und beschriebenen grafischen Notationssymbolen erstellt werden. Die zuvor modellierten Klassen charakterisieren den Kern des zur erstellenden Modells. Durch die Komplementierung von Relationen stellen die Klassen auf einer ersten Ebene das statische Modell des Produktionssystems dar. Die zweite Ebene inkludiert wie bereits dargestellt die logischen und dynamischen Systemaspekte wie Aktivitäten, Ereignisse, Zustände und Zustandsgrößen. Die aus den logischen und dynamischen Systemaspekten ermergierende Systemdynamik wird mit den aus **Kapitel 4.3.2** beschriebenen Verhaltensdiagrammen dargestellt. Die Modellierung der Dynamik schließt sich an die Modellierung der Struktur des Produktionssystems an. Das statische Modell, das die Struktur des Produktionssystems beschreibt, wird durch das Klassendiagramm modelliert und durch das Objektdiagramm auf einer weiteren Abstraktionsebene ergänzt.

Im Folgenden soll zunächst die aufbauende Struktur, das Rückgrat des Produktionssystems, in einem Klassendiagramm mit all seinen Entitäten, Eigenschaften und Relationen modelliert werden. Das Ergebnis der Modellierung ist in **Abbildung 6.1** zu sehen.

6.1.2. Das Objektdiagramm

Nachdem mit dem Klassendiagramm die grundlegende Struktur des Produktionssystems modelliert wurde, erfolgt nach dieser Makrosicht nun die Mikrosicht auf das Produktionssystem. Die Mikrosicht veranschaulicht die instantiierten Objekte der verschiedenen Klassen. Zur Umsetzung bietet die UML das Objektdiagramm als Diagrammtyp zur Visualisierung dieser Modellsicht an.

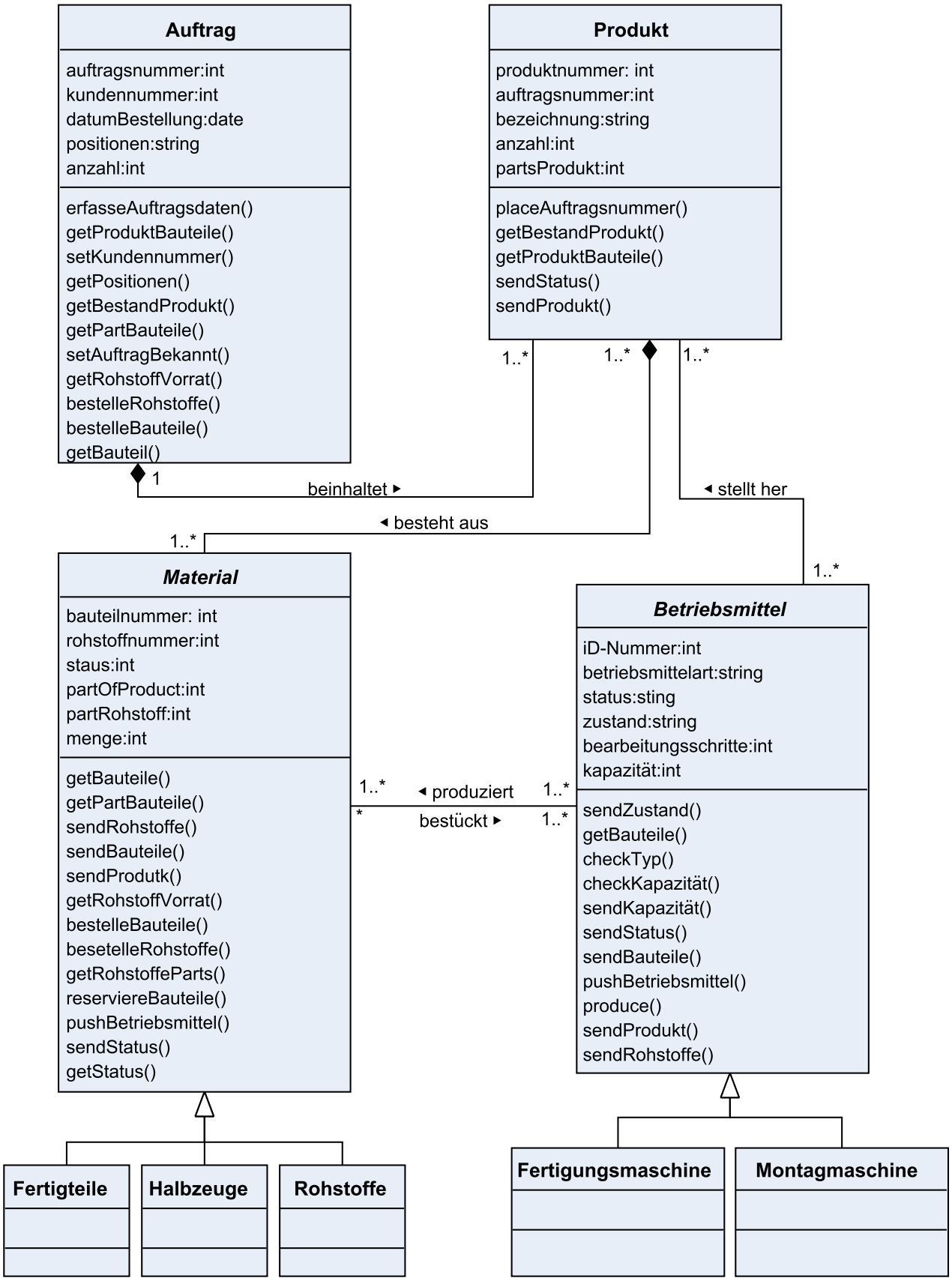


Abbildung 6.1.: Das Klassendiagramm des Produktionssystems

Durch das Objektdiagramm lassen sich Instanzen der unterschiedlichen Klassen mit konkreten Attributwerten visualisieren und deren Existenz sowie Konstellation mit anderen Objekten zu einem konkreten Augenblick im Produktionssystem festhalten.

Die Modellierung des Produktionssystems mithilfe des Objektdiagramms beginnt mit der Darstellung eines ersten Auftrags mit entsprechender Auftragsnummer und Kundennummer. Das Objekt Auftrag1:Auftrag der Klasse Auftrag wird somit instantiiert. Damit die Darstellung und Komplexität überschaubar bleibt, wird wie zu Beginn des zurückliegenden Kapitels davon ausgegangen, dass zum derzeitigen Zeitpunkt das Unternehmen und somit das Produktionssystem zwei Produkte produzieren und anbieten kann. Die Objekte der Klasse **Produkt** sind gekennzeichnet durch eine Produktnummer und werden für einen bestellenden Kunden mit der entsprechenden Auftragsnummer versehen und reserviert. Damit eine Reservierung durchgeführt werden kann, benötigt man eine Operation, die in der Klasse **Produkt** die Bezeichnung *placeAuftragsnummer()* trägt (s. **Abbildung 6.1**). Der Prozess der Reservierung wird auf der zweiten Ebene des Modells, ergo den Verhaltensdiagrammen, modelliert.

Das Produkt produkt2 mit der Produktnummer 0022 setzt sich aus bestimmten Bauteilen zusammen, die sich wiederum aus Objekten teil:Fertigteile der Klasse Fertigteile oder zu produzierenden Objekten teil:Halbzeuge zusammensetzen. Halbzeuge werden des Weiteren aus Rohstoffen produziert und müssen bei Bedarf von einer Fertigungsmaschine hergestellt werden. Wie die Objektbeziehung im Einzelnen aussehen, lässt sich in **Abbildung 6.2** erkennen. Attribute aus dem zuvor gezeigten Klassendiagramm in **Abbildung 6.1** wurden beispielhaft mit Attributwerten belegt.

6.1.3. Das Sequenzdiagramm

Die folgenden Diagramme bilden das Verhalten der Klassen und Instanzen des Produktionssystems ab. Zu Beginn wird unter Verwendung des Sequenzdiagramms die globale Kommunikation der teilhabenden Klassen untereinander dargestellt. Interne Operationen von Objekten, wie etwa *erfasseAuftragedaten()* einer Instanz der Klasse Auftrag, werden nicht abgebildet. Erst im anschließenden Diagramm, dem Zustandsdiagramm, erfolgt die Sicht auf die internen Strukturen der verschiedenen Objekte.

Die Kommunikation unter Instanzen von Klassen vollzieht sich, wie in **Abschnitt 4.2.6** erörtert, durch den Austausch von Nachrichten. Hinter einer Nachricht verbirgt sich eine Operation, die durch sog. Argumente editiert werden kann. Im Sequenzdiagramm werden die Argumente

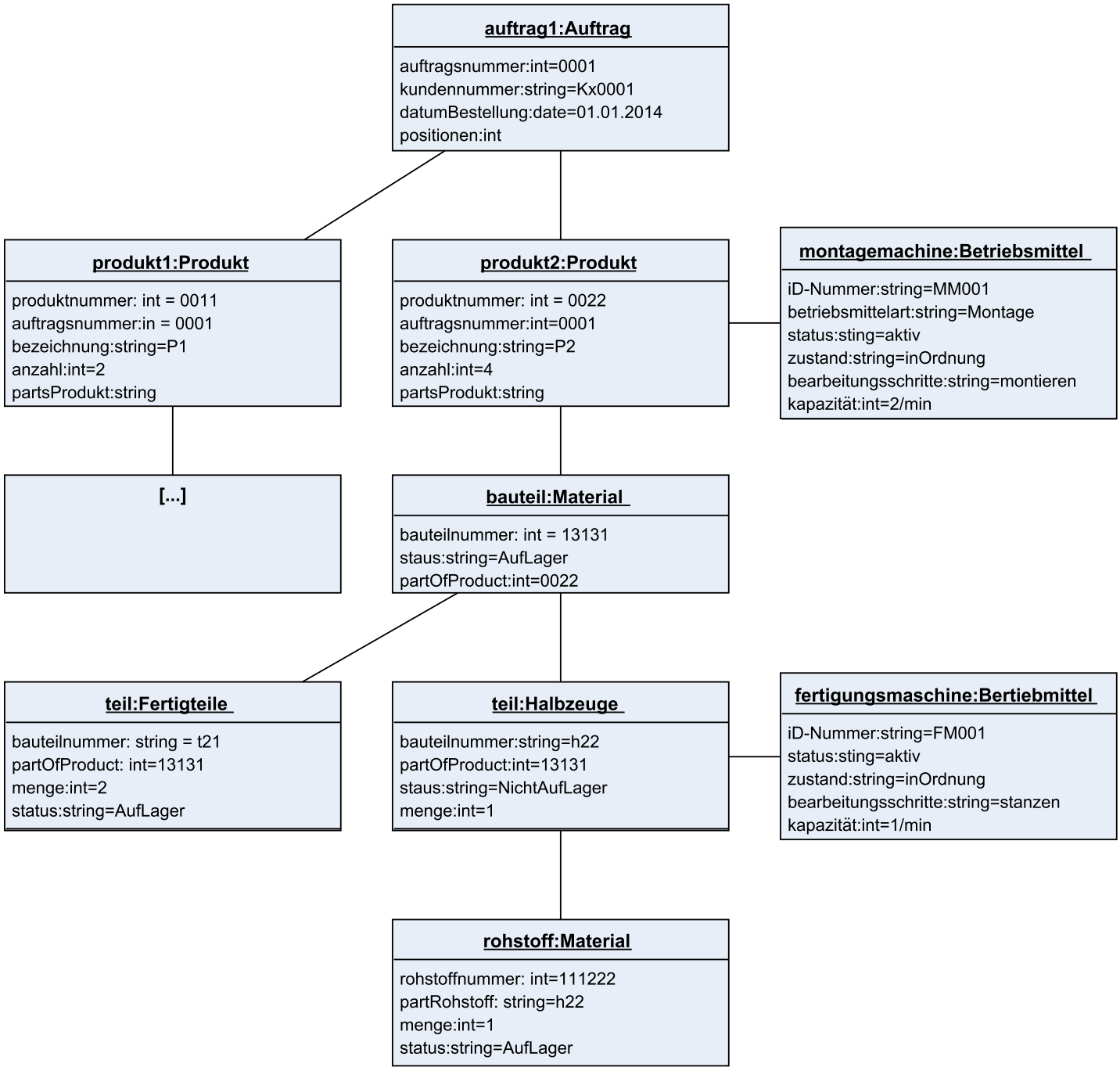


Abbildung 6.2.: Das Objektdiagramm des Produktionssystems

jeweils unterhalb einer aufrufenden Operation abgebildet, um die Bereite des Diagramms zu limitieren. Die Darstellung der Ausführungsdauer (s. **Abschnitt 4.3.2**) ist zeitlich abstrahiert und dient nur der Illustration, dass ein interner Prozess abläuft. Das Resultat interner Aktionen als Reaktion auf eine Operation wird nach der Ausführungszeit zurückgegeben. Da die Ergebnisse meist aufgrund der vorherigen Nachricht deduziert werden können, wird auf eine Bezeichnung der Ergebnisse – solange diese implizit verständlich sind – verzichtet. Anderenfalls wird das Resultat explizit angegeben.

Das Startereignis, mit dem die Prozesse im Produktionssystem angestoßen werden, wird durch die Operation `op0()` hervorgerufen. Die Operation op0 stellt eine Art Pseudooperation dar und erzeugt ein Objekt der Klasse Auftrag, das im weiteren Verlauf die Prozesse der Produktion steuert. Welche Zustände die Instanz der Klasse Auftrag nun durchläuft, ist dem Zustandsdiagramm im nächsten Abschnitt zu entnehmen, soll hier jedoch erst einmal nicht beachtet werden, da es in erster Linie um die Interaktion respektive Kommunikation unter den Klassen geht. Die erste Interaktion, die nun modelliert werden muss, ist die, die auftrag1:Auftrag mit Objekten der Klasse Produkt eingeht. Nachdem durch Aktivitäten im Objekt auftrag1 die Auftragsdaten analysiert wurden, ist bekannt, welche Produkte der Kunde nun bestellen möchte, und es kann der Bestand geprüft werden. Die erste Interaktion liegt folglich zwischen den Klassen **Auftrag** und **Produkt** vor. Die Klasse Auftrag sendet die Nachricht *getBestandProdukt(Auftragsnummer1)* an die Klasse Produkt. Damit bekannt ist, für welchen Auftrag die Verfügbarkeit von Produkten geprüft werden soll, bedarf es der Übergabe eines Identifiers. Ein Identifier ist in diesem Kontext eine eindeutige Kennzeichnung zur Kennung von Objekten. Durch die eindeutige Kennzeichnung mittels einer Folge von Symbolen oder Zeichen lässt sich ein Objekt auf diese Weise eindeutig identifizieren. Als Kennzeichnung bietet sich etwa die Auftragsnummer an, die innerhalb der Produktion einzigartig und eindeutig ist. Durch die Übergabe in Form von *getBestandProdukt(*`Auftragsnummer`*)* ist demnach eine eindeutige Kennzeichnung möglich. Im modellierten Sequenzdiagramm wird nach der vollständigen Formulierung des Übergabeparameters immer eine 1 übergeben, um zu zeigen, dass es sich weiterhin um den ersten Auftrag handelt.

Sollte die Antwort der ersten Bestandsüberprüfung positiv ausfallen, so werden die Produkte für den anfragenden Auftrag reserviert, und der Auftrag kann erfüllt werden. Ein Montageprozess, geschweige denn ein Fertigungsprozess, wird nicht angestoßen. Der Auftrag ist somit bereits abgeschlossen und wird aus dem System gelöscht. Der Destruktor der Klasse Auftrag entfernt den Auftrag aus dem Produktionssystem. Sollten die benötigten Produkte jedoch nicht vorrätig sein, müssen die einzelnen Produkte zusammengebaut werden, und es werden weitere

Prozesse angestoßen.

Im Sequenzdiagramm ist nun die abgebildete Bedingung des ersten Alternativablaufs `[Produkte nicht vorhanden]` wahr geworden. Infolgedessen wird angefragt, welche Materialien für die Herstellung der bestellten Produkte benötigt werden. Durch die auffordernde Operation *getProduktBauteile(Auftragsnummer1)* an die Klasse Produkt reagiert die Klasse mit der Überlieferung der gewünschten Ergebnisse, die in diesem Fall die Angabe über die konstituierenden Bauteile des Produktes sind. Da die Klasse Produkt bereits existiert, liegt die Klasse in der Zeitdimension des Modells über der Klasse Auftrag.

Nachdem das Objekt auftrag1:Auftrag eine Antwort erhalten hat, wird eine weitere Botschaft auf Grundlage der erhaltenden Antwort der Klasse Produkt an die Klasse Material gesendet. Die Nachricht bewirkt die Überprüfung der Verfügbarkeit der angefragten Bauteile. Sollten alle benötigten Bauteile für die Herstellung der Produkte für auftrag:Auftrag1 vorhanden sein, so ist die Bedingungen des zweiten Alternativablaufs `[Bauteile vorhanden]` wahr, und es kann direkt mit der Montage begonnen werden. Die Klasse Material sendet eine Nachricht an die Klasse Betriebsmittel, die ein Objekt montagemaschine:Betriebmittel anstößt, eine Operation auszuführen, durch die alle Bauteile für die Herstellung der Produkte montiert und fertige Produkte mit der entsprechenden Auftragsnummer gekennzeichnet werden. Der Auftrag wird durch weitere Prozesse abgearbeitet und gilt folglich als erfüllt.

Sollten keine Bauteile für die Herstellung zur Verfügung stehen und die Bedingung `[Bauteile nicht vorhanden]` demnach eintritt, so muss unterschieden werden in Bauteile, die bestellt werden müssen, und in diejenigen, die in der eigenen Produktion gefertigt werden. Dazu wird die Klasse Material beauftragt, durch (*getPartBauteile(1)*) zu bestimmen, welche Art von Bauteilen fällt. Je nach Ergebnis werden anschließend diejenigen Bauteile angefordert, die ein externer Dienstleister liefert, und die selbst zu fertigenden Bauteile produziert. Im ersten Fall müssen die fehlenden Bauteile nur aus dem Lager geordert werden. Um Halbzeuge zu produzieren, erfolgt vorab die Abfrage der Verfügbarkeit benötigter Rohstoffe (*getRohstoffeVorrat(1)*) zur Produktion der Halbzeuge. Sind alle benötigten Rohstoffe vorhanden, die Bedingung `[Rohstoffe vorhanden]` demnach wahr, müssen die Rohstoffe für die Produktion der Halbzeuge bereitgestellt werden. Die Klasse Material stößt die Klasse Betriebsmittel durch *pushBetriebsmittel(Fertigung:Auftrag1)* an und übergibt gleichzeitig die nötigten Argumente Fertigung und Auftrag1. Das Argument Fertigung definiert, welches Objekt der Klasse Betriebsmittel angesprochen wird, nämlich eine Fertigungsmaschine. Auftrag1 kennzeichnet die zu produzierenden Halbzeuge. Ist eine Fertigungsmaschine bereit, werden durch die Botschaft *sendRohstoffe(1)*

die nötigen Rohstoffe angefordert. In diesem Fall wird das Betriebsmittel Fertigungsmaschine bestückt und kann mit der Produktion beginnen. Die benötigten Teile werden anschließend der Materialklasse zugeteilt. Sollten die Rohstoffe nicht vorliegen, müssen die fehlenden Rohstoffe aus dem Lager bestellt werden.

Ist der im Sequenzdiagramm grün hinterlegte Prozess (`loop1`) durchlaufen und der Auftrag noch nicht erfüllt, beginnt der Prozess von vorn. Bestellte Bauteile und Rohstoffe sowie hergestellte Halbzeuge sollten nun – ist dies im ersten Durchlauf noch nicht der Fall gewesen – vorrätig sein und können gemäß dem beschriebenen Prozessablauf eingesetzt und be- bzw. verarbeitet werden. Im Idealfall kann der Auftrag nun erfüllt werden und wird, wenn er erfüllt wurde, durch die Destruktor-Operation *destroy()* im Ablauf `alt1` gelöscht. Die Destroy-Operation stellt gleichzeitig die Abbruchbedingung der Schleife dar und kann einen nächsten Auftrag bearbeiten.

6.1.4. Die Zustandsdiagramme der Objektklassen

Industrielle Produktionsprozesse werden in der Regel in einem ereignisorientierten System beschrieben (vgl. **Kapitel 2.1**). Für die vollständige Abbildung eines ereignisorientierten Produktionssystems sind einzelne Zustände, Zustandsübergänge und Ereignisse zu berücksichtigen, die für das Fortschreiten der Produktionsprozesse relevant sind. In einem solchen System ändern sich durch ein Ereignis initiierte Zustände nicht kontinuierlich über die Zeit. Änderungen (Transitionen) vollziehen sich nur zur bestimmen, diskreten Zeitpunkten und manifestieren sich im Auftreten von Ereignissen. Die Menge der Zustandsänderungen von Objekten ist klar definiert. Demnach lassen sich in einem Produktionssystem die Zustände klar voneinander abgrenzen. Der Zustandsraum – die Anzahl an möglichen Zuständen – ist endlich und lässt sich gut überschauen.

Um die Zustände der Objekte aus verschiedenen Klassen darzustellen, kommen gemeinhin Zustandsautomaten zum Einsatz. Die UML orientiert sich an Zustandsautomaten und stellt auf dieser Grundlage Zustandsdiagramme für die Abbildung von Zuständen, Zustandsübergängen und Ereignissen zur Verfügung (vgl. **Kapitel 4.3**). Alle Klassen, ausgenommen abstrakte Klassen, sind wie bekannt in der Lage, unterschiedliche Objekte zu erzeugen. Ein erzeugtes Objekt durchläuft im Laufe seiner Existenz unterschiedliche Zustände, die durch unterschiedliche Attributausprägungen der Objektattribute beschrieben werden können. Das Zustandsdiagramm veranschaulicht die Zustände, die im Laufe der Existenz eines Klassenobjektes von diesem Objekt durchlaufen werden, und ebenso, welche Ereignisse für die Zustandsänderungen verant-

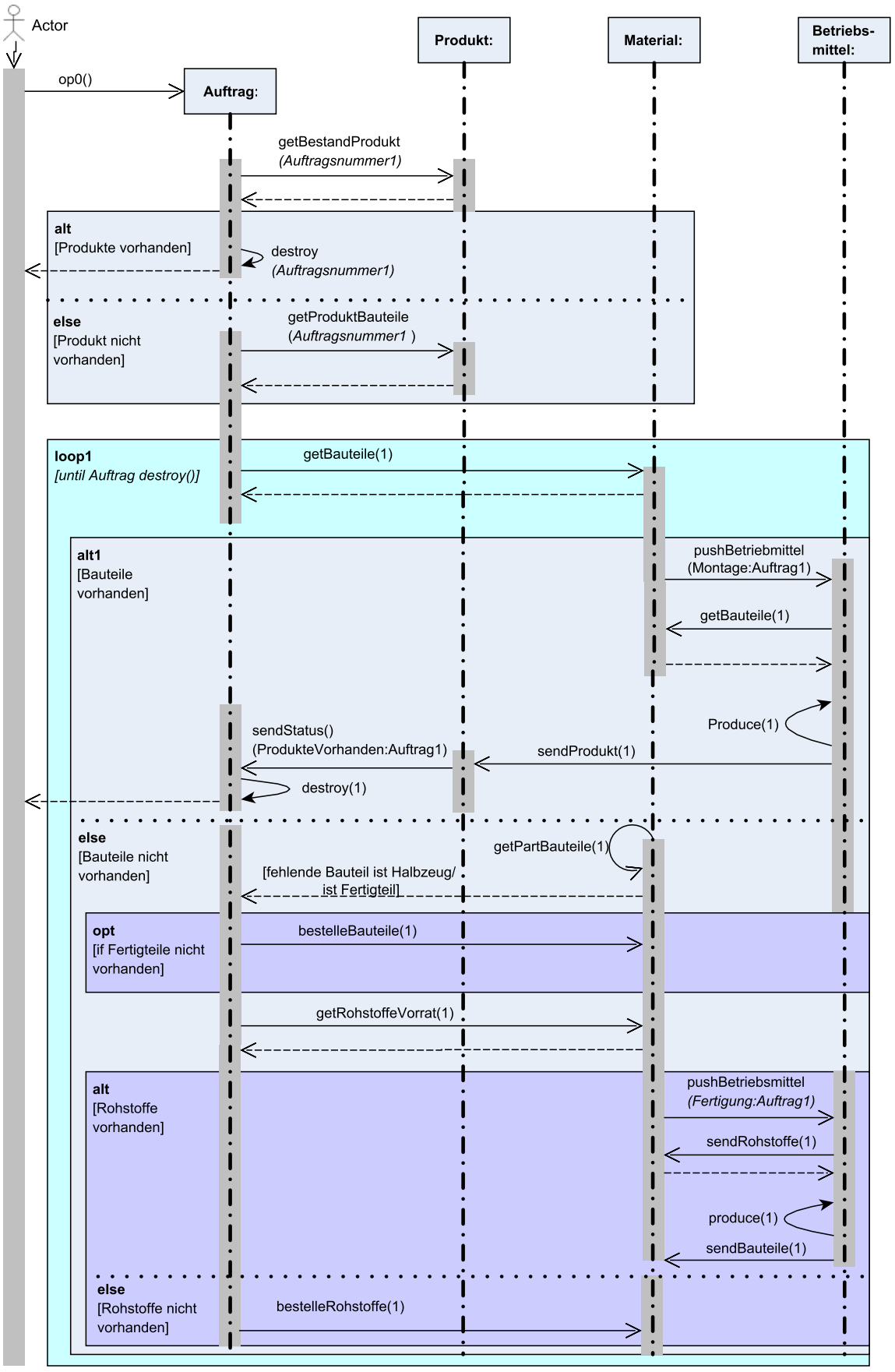


Abbildung 6.3.: Darstellung der Interaktion und Kommunikation im modellierten Sequenzdiagramm

wortlich sind.

In der nachfolgenden Ausführungen werden die Zustandsdiagramme für die verschiedenen Klassen zur besseren Übersichtlichkeit unabhängig voneinander modelliert. Um Erkenntnisse über die Zusammenarbeit und Interaktion der Klassen untereinander zu erlangen, sei auf das zuvor erstellte Sequenzdiagramm verwiesen (**Abschnitt 6.1.3**). Durch Kombination beider Diagrammtypen ist eine umfassende Darstellung der verhaltensspezifischen Aspekte der Produktionsprozesse möglich, die schließlich die Dynamik eines Produktionssystems ausmachen.

Zustandsdiagramm Auftrag

Die Prozesse in einem Produktionssystem beginnen in der Regel mit dem Eingang eines Auftrags, der einen initialen Produktionsprozess anstößt. Ein neuer Kundenauftrag geht in der Regel nicht direkt in der Produktion ein, sondern wird von einem anderen System im produzierenden Unternehmen verwaltet. Der Auftrag bekommt in diesem verwaltenden System eine Auftragsnummer und wird an den Unternehmensbereich Produktion weiter geleitet. Die erste Aktivität, die im Produktionssystem durch den eingehenden Auftrag ausgelöst wird, ist die Erfassung der Auftragsdaten. Durch die Vergabe der Auftragsnummer kann nun geprüft werden, ob dieser Auftrag bereits eingegangen ist und im Produktionssystem vorliegt. Sollte das Ergebnis der Überprüfung sein, dass der Auftrag vorliegt, so kann direkt mit der Überprüfung des Bestandes hinsichtlich der im Auftrag bestellten Produkte erfolgen. Wurde die Auftragsnummer als bekannt markiert, wurde der Auftrag bereits bearbeitet, und die produzierten Produkte liegen möglicherweise schon auf Vorrat. Sollte das der Fall sein, besitzen die entsprechenden Produkte eine Kennzeichnung, die mit der des Auftrags übereinstimmen muss. In der Regel sollte die zweite Prüfung eines bereits eingegangenen Auftrages zu dem Resultat führen, dass die Produkte vorliegen und somit kommissioniert werden können. Blickt man von oben auf das Zustandsdiagramm in **Abbildung 6.4**, lässt sich dieser Prozess durch den Verlauf der linken Kante verfolgen. An den abgebildeten Entscheidungsknoten wählt man in Richtung des Kantenverlaufs immer die rechte fortlaufende Kante. Der Auftrag gilt mit den Eintreten des Endzustands als bearbeitet.

Liegt die Auftragsnummer nicht im System vor, so erfolgt zunächst die Durchsicht, ob der bestellende Kunde bekannt ist. Wurde der Kunde aufgrund einer früheren Bestellung bereits mit einer Kundennummer gekennzeichnet, gilt der Auftrag als erfasst. Andererseits muss erst eine neue Kundennummer angelegt werden. Des Weiteren gilt es, die Bestellpositionen für

diesen Auftrag zu bestimmen und die benötigten Produkte bereitzustellen. Sind die Produkte nicht vorhanden, was der übliche Fall bei einem neuen Auftrag ist, wird der Auftrag als bekannt markiert, und die Bauteile für die Produkte werden bestimmt. Ist genügend Material vorhanden, kann sofort mit der Produktion der Produkte begonnen werden. Ist dahingegen das benötigte Material nicht vorhanden, so muss erst die Produktion für das Material angestoßen werden. Sind schließlich alle Materialien vorhanden, so wird der Auftrag mit all seinen Daten zum Produktionsauftrag. Dargestellt sind die beschriebenen Zustände und Ereignisse in **Abbildung 6.4**.

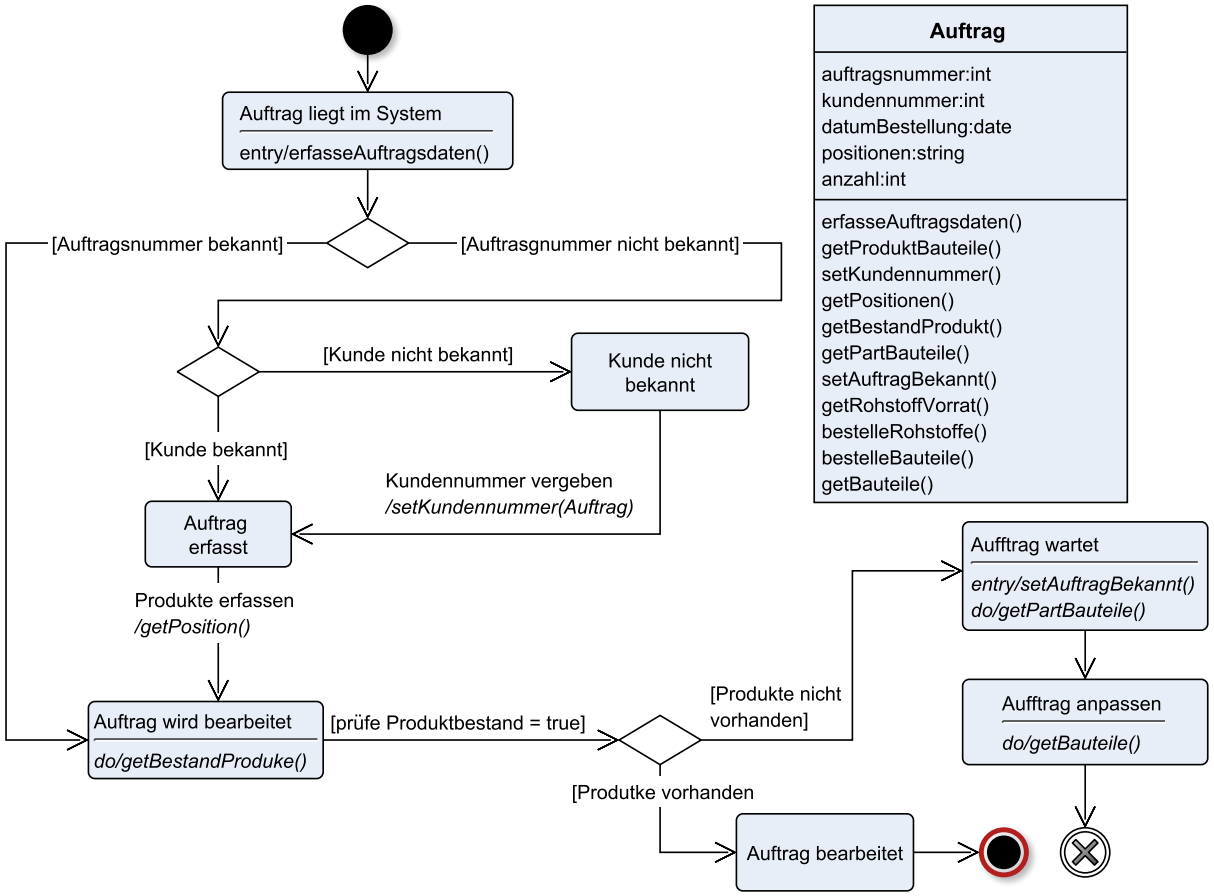


Abbildung 6.4.: Interne Zustände und Ereignisse der Klasse Auftrag. Dargestellt durch das modellierte Zustandsdiagramm

Zustandsdiagramm Produkt

Instanzen der Klasse Produkt sind die von einem Unternehmen angebotenen Produkte. Geht ein Produktionsauftrag ein, erfolgt als erster Schritt die Überprüfung, ob die bestellten Produkte des entsprechenden Auftrags im bereits produzierten Bestand vorrätig sind. Vorerst

muss hierfür durch die Operation *getBestandProdukt(Auftrag1)* der Bestand der bestellten Produkte geprüft werden. Wurde der Zustand `Produktbestand prüfen`, in dem die Operation *prüfeBestand(Auftrag1)* ausgeführt wird, verlassen, so erfolgt der Übergang in den Zustand `Bestand geprüft`, ausgelöst durch das Ereignis *[Bestand geprüft=true]*. Bei Eintritt in den Zustand Bestand geprüft wird die Operation *sendStatus(Auftrag1)* ausgeführt, die das Ergebnis der Bestandsprüfung zurückliefert. Das Resultat – sollte die Überprüfung des Bestandes bezüglich des Auftrages die erste sein – ist im Allgemeinen negativ. Das bedeutet, die im *Auftrag1* geforderten Produkte sind nicht vorrätig. Die Produktion muss also die nötigen Produkte zusammenbauen und verlangt hierfür eine Bauteilliste. Wurde die Liste durch die Methode der Operation *getProduktBauteile(1)* erstellt, beendet die Klasse Produkt ihre Aktivität.

Wurde der Produktionszyklus, dargestellt in **Abschnitt 6.1.3**, bereits erfolgreich durchlaufen, so liegen die Produkte für einen wartenden Auftrag bereits vor, und die Bestandsprüfung würde ein positives Ergebnis zurückliefern. In diesem Fall werden die Produkte reserviert und können kommissioniert werden. Das in **Abbildung 6.5** abgebildete Modell des Zustanddiagramms der Klasse Produkt gilt dann als beendet.

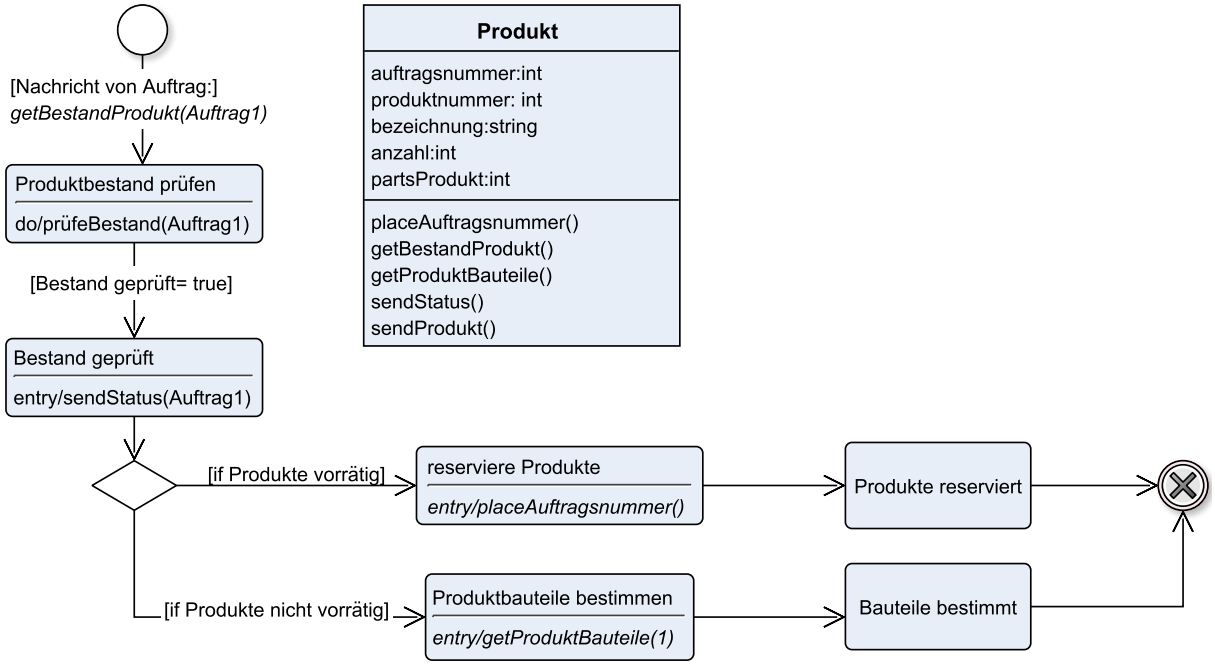


Abbildung 6.5.: Das Zustandsdiagramm mit internen Zuständen und Ereignissen der Klasse Produkt

Zustandsdiagramm Material

Wird die Klasse Material durch das Ereignis *getBauteile(Auftrag1)* angesprochen, werden die Bauteile für die bestellten Produkte bestimmt. Falls die Bauteile vorrätig sind, können durch die Klasse Material anschließend die benötigten Bauteile für die Herstellung der Produkte bereitgestellt werden. Zur Überprüfung der Verfügbarkeit wird die Operation *sendStatus(Produkte:Auftrag1)* ausgeführt. Die Methode der Operation überprüft den Bestand der benötigten Teile. Je nach Ergebnis der Bestandsprüfung der Produktbauteile nehmen die Transitionen am Entscheidungsknoten den auf dem Ergebnis basierenden Kantenverlauf. Sind alle benötigten Bauteile vorhanden, so können die Produkte produziert werden, und der Zustand wechselt in `Betriebsmittel anstoßen`. Ausgehend von diesem Zustand werden die Kanten geteilt und laufen parallel weiter (vgl. **Abschnitt 4.3.2**). Die Bauteile werden durch die Operation *reserviereBauteile(Auftrag1)* für den entsprechenden Auftrag reserviert, und eine Montagemaschine wird durch die Operation *pushBestriebsmittel(Montage)* benachrichtigt, Bauteile zu ordern. Beide Kanten werden parallel verfolgt, sodass nach der Synchronisation der Kanten der Zustandsübergang in den Zustand `warten` erfolgt. In diesem Zustand wird solange die Operation *getStatus(Montagemaschine)* ausgeführt, bis ein Status der Montagemaschine eingeht. Übermittelt die Maschine den Status bereit, so werden die Bauteile zugestellt, und es kann damit begonnen werden, die benötigten Produkte zu montieren. Sollte der Status nicht bereit übermittelt werden, liegt weiterhin der Zustand `warten` mit der internen Operation vor.

Durch die Operation zur Überprüfung des Bauteilevorrats *sendStatus(Produkte:Auftrag1)*, kann durchaus der Zustand `BauteileNichtVorrätig` eintreten. Dieser Zustand führt dazu, dass der Typ der fehlenden Bauteile identifiziert werden muss (*getPartBauteile(1)*). Sollten Fertigteile fehlen, müssen die entsprechenden Teile nur geordert werden. Fehlen jedoch Halbzeuge, die intern produziert werden können, so werden die benötigten Rohstoffe durch *getRohstoffeParts(1)* ermittelt. Sind die benötigten Rohstoffe vorhanden, so können die nachgefragten Halbzeuge direkt produziert werden. Die folgenden zu durchlaufenden Zustände und Transitionen verlaufen analog zu denen im oberen Teil des Zustanddiagramms, indem die Produkte bei vorhandenen Bauteilen produziert werden. Aus diesem Grund wird der Ablauf der Halbzeugproduktion nicht noch einmal in seinem vollen Umfang beschrieben. Der einzige Unterschied besteht darin, dass die Klasse Fertigungsmaschine mit ihren Instanzen angesprochen wird und nicht die Klasse Montagemaschine. Durch den soeben beschriebenen Sachverhalt lässt sich das Zustandsdiagramm der Objektklasse Material wie in **Abbildung 6.6** modellieren.

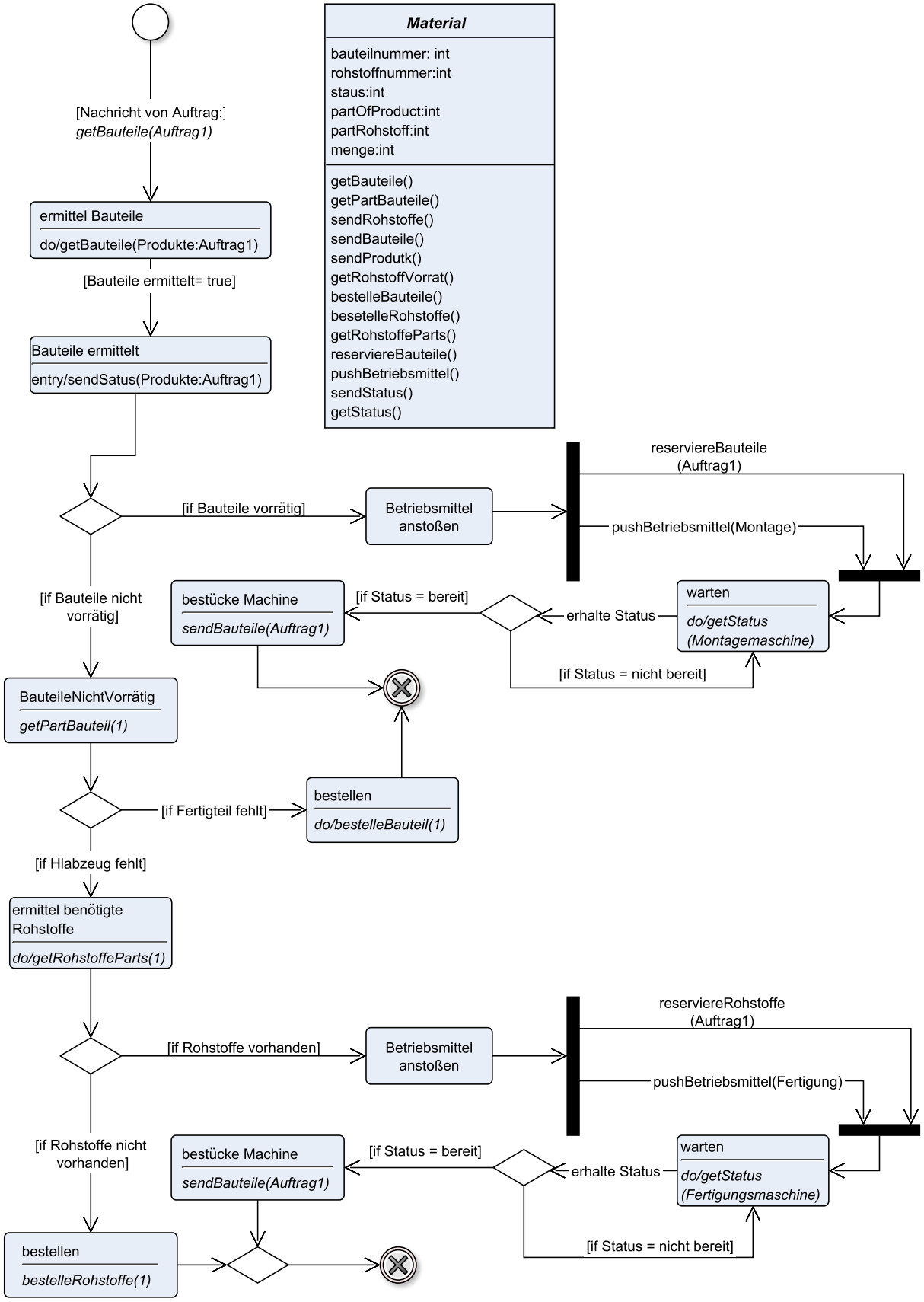


Abbildung 6.6.: Das Zustandsdiagramm mit internen Zuständen und Ereignissen der Klasse Material

Zustandsdiagramm Betriebsmittel

Die Klasse Betriebsmittel stellt die letzte Objektklasse dar, deren Zustände einer Modellierung bedürfen. Das Ergebnis der Modellierung des Zustandsdiagramms ist in **Abbildung 6.7** dargestellt. Angestoßen werden die jeweiligen Objekte *fertigungsmaschine:Betriebsmittel* und *fontagemaschine:Betriebsmittel* durch eine Nachricht, die von der Klasse Material gesendet wird. Primär wird aufgrund der übergebenen Operationsparameter der Klasse Material geprüft, welcher Typ Betriebsmittel überhaupt gebraucht wird. Ist die Überprüfung durch die Operation *check-Typ()* erfolgt, so wechselt der Zustand von `betriebsmittel ermitteln` in `Betriebsmitteltyp bestimmt`. Daraufhin werden die verfügbare Kapazität und der Zustand des jeweiligen Betriebsmittels überprüft und in einem Status zusammengefasst. Sollte das Ergebnis der Ermittlung des Status `[Status = nicht bereit]` lauten, wird erneut eine Kontrolle durchgeführt. Ist alles in Ordnung, ist also Kapazität vorhanden, und eine Wartung oder Ähnliches steht nicht aus, kann mit der Produktion respektive der Montage begonnen werden. Dazu wird das entsprechende Material angefordert. Liegt das Material vor, kann das Betriebsmittel seine entsprechende Tätigkeit ausführen.

6.2. Erweiterbarkeit des Objektmodells

Mit Sicherheit wird das in dieser Arbeit erstellte Modell nicht in seiner ursprünglichen Form bleiben, da, wie eingehend beschrieben worden ist, die subjektive Auffassung des Verfassers von einem Produktionssystem in Bezug auf wichtige Aspekte wie Relationen und Aktivitäten von Komponenten die Modellierung geleitet haben. Das Modell stellt jedoch auch nicht den Anspruch, vollständig zu sein, sondern soll ein Referenzmodell darstellen, das situationsspezifisch angepasst werden kann. Zusätzlich spielt das Modell eine wichtige Rolle als Anschauungsmodell für die in dieser Arbeit vorgestellte Methode.

Das entwickelte Objektmodell biete viele Möglichkeiten der Anpassung. Da es auf dem objektorientierten Ansatz beruht, können Klassen, Objekte sowie Attribute und Operationen nach Belieben und ohne großen Aufwand angepasst und erweitert werden. Es besteht also die Möglichkeit, das Modell als Basis zu verwenden und entsprechend vorliegender Rahmenbedingungen zu editieren und zu erweitern. Um bspw. das Personal in einem Produktionssystem zu berücksichtigen, kann einfach eine neue Klasse erstellt werden, die dann entsprechende Eigenschaften besitzt, die für eine Repräsentation im Modell relevant erscheinen. Beziehungen

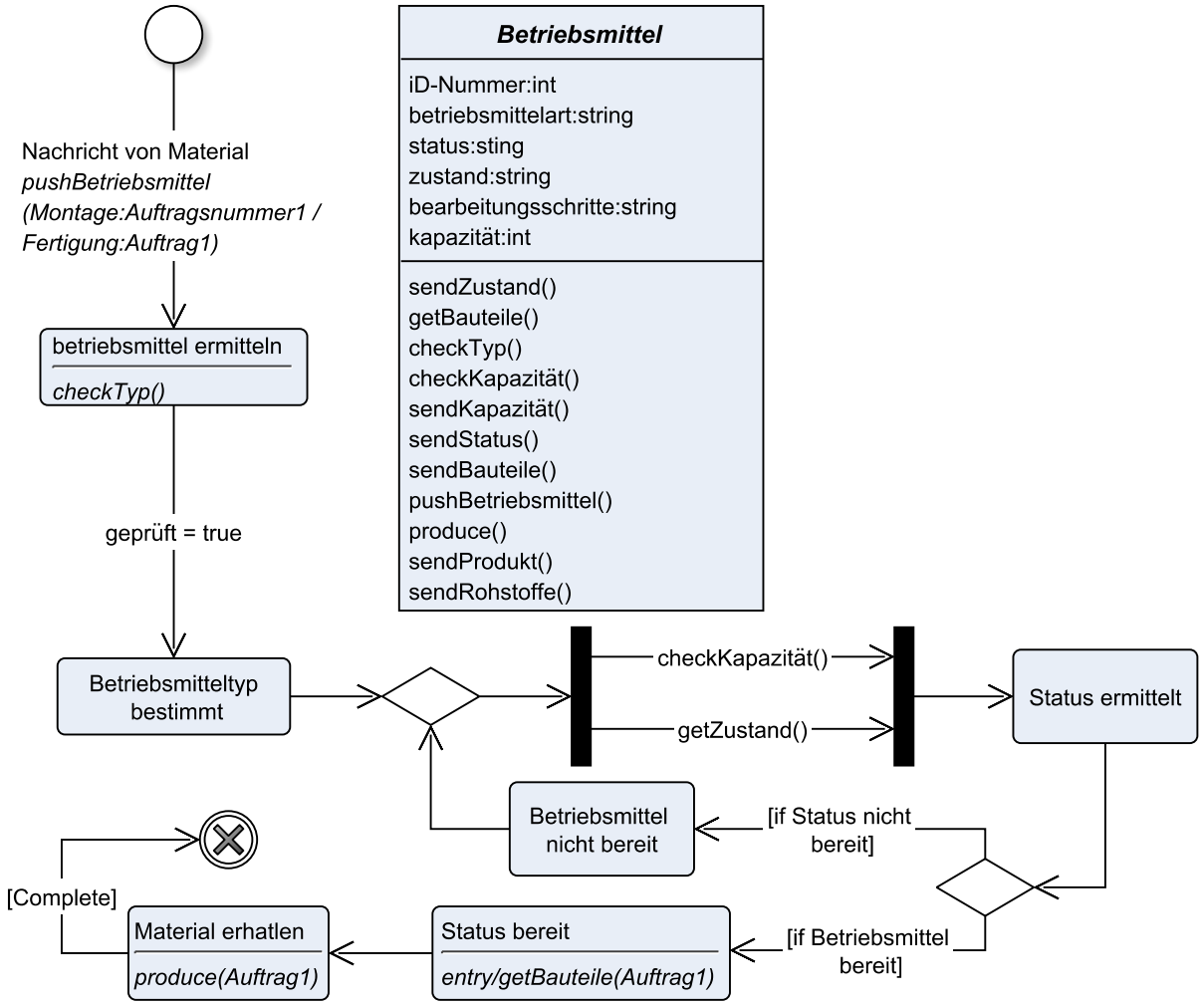


Abbildung 6.7.: Interne Zustände und Ereignisse der Klasse Betriebsmittel

zu anderen Klassen sind überdies ebenfalls ohne Schwierigkeiten zu ergänzen.

Eine der limitierenden Faktoren in einem Produktionssystem ist in der Regel die durch die im Produktionsprozess vorhandenen Betriebsmittel zur Verfügung gestellte Kapazität. Je nach Bedarf und einer gründlichen Überlegungen seitens der Unternehmensführung könnte eine mögliche Strategieentscheidung die Anschaffung neuer Produktionsfaktoren sein. Das konzeptionierte Objektmodell muss ausschließlich durch die Anzahl geplanter Neuanschaffungen erweitert und weitere Unterklassen der Klasse Betriebsmittel erstellt werden. Eine umfassende Anpassung des Modells ist demnach nicht Nötig, da die grundlegende Struktur, die Eigenschaften und Beziehungen zu anderen Klassen im Produktionssystem, bereits modelliert ist. Darüber hinaus kann je nach Fertigungstiefe die Anzahl benötigter Teile und Rohstoffe variieren, was die Erweiterung des technologischen oder produktorientierten Bereichs zur Folge hat. Auch in diesem Fall lassen sich die Objekte aus den bereits entwickelten Klassen ableiten und erstellen.

Schließlich erleichtert das objektorientierte Modell die spätere Umsetzung und Anpassung an ein beliebiges Produktionssystem.

Die Verwendung der Zustands- und Sequenzdiagramme für die Verhaltensbeschreibung ermöglicht es, über die fachliche Korrektheit des dargestellten Verhaltens zu diskutieren. Bei verschiedenen Auffassungen über die Richtigkeit und Detailgenauigkeit der abgebildeten und beschriebenen Dynamik lässt sich einfach debattieren, da die Anpassung an jeweils andere Verhaltensaspekte durch einen nicht absolut vorgeschriebenen Detaillierungsgrad der UML ohne Weiteres möglich ist. Durch die Grundlage der Objektorientierung müssen nur entsprechende Operationen und Attribute editiert werden, um das situationsspezifische Verhalten darstellen zu können.

Weiterhin liegt die Einfachheit in der Erweiterbarkeit darin begründet, dass den vorhandenen Objekten im Modell neue Beziehungen zugeschrieben werden können. Lassen sich neue Entitäten in einem Prozess identifizieren, so können sie einfach als neue Objekte in den vorhandene Objektklassen ergänzt werden. Eine leichte Erweiterbarkeit wird überdies durch eine sorgfältig aufgebaute Vererbungshierarchie garantiert. Die Vererbung ist nach [Balzert u. a., 2011] ein wichtiges Element der Objektorientierung und bringt einen großen Vorteil mit sich. Durch eine strenge und sorgfältig aufgebaute Vererbungshierarchie wird zum einen die leichte Erweiterbarkeit garantiert und darüber hinaus ein hohes Maß an Wiederverwendbarkeit innerhalb eines Projektes. Ohne größeren Aufwand können aus bereits existierenden Klassen neue Klassen generiert werden. Dadurch, dass die UML eine prosperierende Modellierungssprache darstellt, sorgt sie für eine gewisse Langlebigkeit erstellter Modelle.

7. Zusammenfassung

Resümee

Die vorliegende Arbeit stellt das zugrunde liegende Konzept und Vorgehen der objektorientierten Modellierung mit den Werkzeugen der Unified Modelling Language theoretisch dar und zeigt darauf aufbauend deren Anwendung in Bezug auf ein industrielles Produktionssystem. Um die Anwendung plausibel zu beschreiben, wurden die Entwicklung und die Modellierung eines objektorientierten Modells aufgezeigt. Ziel war es, ein einfaches Modell zu entwickeln, das ausreicht,um ein Produktionssystem und dessen Prozesse objektorientiert zu beschreiben. Das entstehende Modell sollte möglichst flexibel auf verschiedene Rahmenbedingungen anpassbar sein, sodass in einem konkreten Kontext entschieden werden kann, ob das konzeptionierte Modell bereits angemessen ist oder erweitert werden muss. Zudem sollte das Modell als Basis für die Verwendung in einer ereignisdisktreten Simulationsanwendung dienen.

Geforderte Eigenschaften, die eine geeignete Modellierungssprache mit sich bringen sollte, um ein Produktionsprozess struktur- und verhaltensspezifisch sowie verständlich und nachvollziehbar abzubilden, waren zum einen eine klare, eindeutige und intuitive Symbolnation und die Möglichkeit, parallele und alternative Abläufe darzustellen. Zum anderen musste die Systemdynamik der Produktionsprozesse abzubilden sein. Obendrein sind Anforderungen zu nennen, die sich mit dem Paradigma der Objektorientierung decken. Die angesprochenen Anforderungen sind ein leichte, problemlose Erweiterung und Adaption des Modells auf verschiedene Anwendungsbereiche.

Durch das konzeptionierte objektorientierte Modell wird nicht der Anspruch auf Vollständigkeit erhoben, sondern das Modell soll dem Anspruch als Referenzmodell gerecht werden, das nach Belieben verändert, erweitert und angepasst werden kann, ohne den prinzipiellen Aufbau des Modells und die grundlegende Konzepte der Objektorientierung zu verwerfen. Das Modell beschränkt sich auf grundlegende Bestandteile eines Produktionssystems wie Auftrag, Material, Betriebsmittel und dem herzustellenden Produkt. Für die Modellierung des Modells

wurde auf die Unified Modelling Language zurückgegriffen, die es erlaubt, durch ihre Fülle an grafischen Notationselemente und Diagrammtypen die strukturellen Beziehungen sowie das dynamische Verhalten der Komponenten in einem Produktionssystem abzubilden. Die Verwendung von Verhaltensdiagrammen ermöglicht es, ein Produktionssystem im Hinblick auf die ereignisorientierte Simulation adäquat und sachdienlich mit den zur Verfügung gestellten Werkzeugen der UML abzubilden. Um das Modell als Ausgangspunkt einer ereignisorientierten Simulationsanwendung zu verwenden, war die Erfüllung besonderer Anforderung erforderlich, die zum einen eine mögliche ereignisorientierte Darstellung mit der Illustration von Ereignissen, Zuständen und Zustandsänderungen forderte. Die genannten Eigenschaften sind, wie in der Arbeit beschrieben, fundamentale Aspekte einer ereignisorientierten Produktion und konnten durch die Verwendung der UML modelliert werden.

Die objektorientierte Modellierungssprache UML erlaubt eine detaillierte Darstellung und Abbildung eines Produktionssystems und der auszuführenden Produktionsprozesse. Die grafische Darstellung ist zur Erlangung eines grundlegenden Verständnisses über die generelle Struktur und Dynamik eines Produktionsprozesses sehr gut geeignet. Es werden ausreichend Diagrammtypen für eine geeignete Darstellung angeboten, deren Verwendung sich jeweils für einen bestimmten Zweck anbietet. Für die Intention des in dieser Arbeit erstellten Modells – die Konzeptionierung eines erweiterbaren Objektmodells für die ereignisdiskrete Simulation – wurden jeweils zwei Diagrammtypen aus den Struktur- und Verhaltensdiagrammen ausgewählt. Klassen- und Objektediagramm stellen den strukturellen Aufbau des Produktionssystems dar. Um die Dynamik des Systems darzustellen und zu beleuchten, erwiesen sich das Sequenz- und das Zustandsdiagramm als äußerst fruchtbar.

Zudem lässt sich das Modell aufgrund des Konzepts der Objektorientierung sehr leicht um neue Elemente erweitern. Es ist somit für die volatile Unternehmensumwelt gut geeignet und erfüllt den Anspruch, flexibel und erweiterbar zu sein. Die Vererbung ist hierfür ein wichtiges Element der Objektorientierung und bringt einen großen Vorteil mit sich. Denn durch eine strenge und sorgfältig aufgebaute Vererbungshierarchie wird zum einen garantiert, dass ohne größeren Aufwand aus bereits existierenden Klassen neue Klassen generiert werden können und darüber hinaus ein hohes Maß an Wiederverwendbarkeit innerhalb eines Projektes besteht. Die Objektorientierung wird zudem von einigen Autoren als prosperierendes Modellierungskonzept angesehen und ist demzufolge, ebenso wie die UML, ein zukunftsträchtiger Ansatz für die Prozessmodellierung. Überdies ist die UML eine anerkannte Standardmethode in der Softwareentwicklung und ein bekanntes Modellierungsinstrument in diesem Bereich und findet in der

Industrie breite Verwendung (vgl. **Abschnitt 4.2**). Da sich die UML, wie in dieser Arbeit ausgearbeitet, ebenfalls für die Prozessmodellierung eignet, ist eine schlüssige Durchgängigkeit von der Prozessmodellierung bis hin zur Softwaremodellierung unter Berufung auf ein Standartverfahren möglich. In Konsequenz ist somit die Modellierung eines umfassenden objektorientierten und integrierten Produktionsprozessmodells erreichbar.

Durch die konsistente Verwendung der Objektorientierung in der Modellierung ist durch die Benutzung einer objektorientierten Programmiersprache des Weiteren eine problemlose Überführung in eine textuelle Darstellungsform wie Programmcode realisierbar. Durch die grafischen Symbole und die eindeutige Abbildung von Eigenschaften, Beziehungen, Kommunikationen, Ereignissen und Zuständen ist eine leichte und verständliche Interpretation des erstellten Modells möglich. Somit ist das Konzept des in dieser Arbeit erstellten Modells durchaus für die primäre Darstellung eines Produktionssystems praktikabel, um es im Anschluss in einen Programmcode zu überführen. Als geeignete Sprache, um das Modell in einen Programmcode zu überführen, wurde die Sprache C++ erwähnt.

Nachteilig zeigen ich dennoch die unterschiedlichen Interpretationsmöglichkeiten eines erstellten Modells im Hinblick auf eine Implementierung. Trotz klarer Darstellung und Verständlichkeit obliegt es dem Programmierer, das durch einen Modellbildner erstellte Modell zu interpretieren. Überdies ist die UML eine äußerst komplexe Modellierungssprache und bietet ein breites Spektrum an Diagrammtypen und eine enorme Fülle an symbolischen und textuellen Darstellungsmöglichkeiten. Dementsprechend ist eine gewisse Einarbeitungsphase in die UML erforderlich.

Grosso modo ist das mit der UML konzeptionierte objektorientierte Modell für die Abbildung und Darstellung von Produktionssystemen für die ereignisorientierte Simulation zweckmäßig. Es visualisiert alle wichtigen Eigenheiten und relevanten Aspekte und ermöglicht die Darstellung dynamischer Systemaspekte. Aufgrund der Symbolnation und Objektorientierung ist es einfach in ein Simulationsmodell zu überführen. Inwieweit das erstellte Modell für die Simulation tatsächlich nützlich ist, lässt sich nur durch eine Implementierung in ein Simulationsprogramm überprüfen.

Ausblick

Model Driven Development (MDD), die Modellgetriebene Softwareentwicklung, ist eine stark an Bedeutung gewinnende Entwicklungstechnik, die dabei helfen soll, Entwicklern eine produktivere, d. h. effizientere und effektivere Softwareentwicklung zu ermöglichen. Im Gegensatz zur üblichen

Entwicklungsmethoden, bei denen bspw. erstellte Modelle durch eine umfangreiche Analyse interpretiert werden müssen, basiert die MDD auf der Grundidee, dass ein zuvor erstelltes Modell als Ausgangsbasis für die Erstellung von Programmcode genutzt wird. Aus einem durch Modellierungssprache erstellten Modell soll so automatisch bzw. teilautomatisch ein Programmcode und die dazugehörigen Implementierungs- sowie Konfigurationsdateien erstellt werden. Diese Vorgehensweise beabsichtigt, den Schwerpunkt der Softwareentwicklung auf die Modellierung zu legen. Der Fokus liegt somit nicht mehr auf der Programmierung; die Entwicklung von Software findet somit auf einer höheren Abstraktionsebene – der Modellsicht – statt. In dieser Entwicklung spielt die in dieser Arbeit vorgestellte und verwendete Unified Modelling Language eine entscheidende Rolle [Rupp u. a., 2012]. Rupp u. a. indizieren jedoch, dass die Erwartungen an den bisherigen Entwicklungsstand nicht zu hoch angesetzt werden sollten.

Modelle von Produktionssystem finden auch aufgrund gestiegener Anforderungen an Unternehmen durch eine notwendige Flexibilität immer größere Beachtung. Ein herausragender Trend wird derweilen als vierte industrielle Revolution oder auch Industrie 4.0 bezeichnet. Auf Basis geeigneter Modelle, die wie das in dieser Arbeit erstellte Modell einen Produktionsprozess abbilden, sollen Produkte und Betriebsmittel zunehmender vernetzt werden, wodurch die Verfügbarkeit von Detailinformationen über die Produktion erhöht werden soll. Dadurch wird ermöglicht, eine neue Qualität in der Verknüpfung von Produktionssystem mit ihren Modellen zur Planung und Steuerung zu erreichen [März u. a., 2011].

Literaturverzeichnis

[ITW 2014] ITWISSEN (Hrsg.): *Stichwort: Datentyp*. http://www.itwissen.info/definition/lexikon/Boolesche-Algebra-Boolean-algebra.html. Version: 2014. – Online, Stand: 20.08.2014 10:15

[Gab 2014] WIRTSCHAFTSLEXIKON, Gabler (Hrsg.): *Stichwort: Modell*. http://wirtschaftslexikon.gabler.de/Archiv/495/modell-v11.html. Version: 2014. – Online, Stand: 08.07.2014 12:26

[Balzert u. a. 2011] BALZERT, Heide ; PROBANDT, Wolfgang ; VERING, Oliver: *Lehrbuch der Objektmodellierung. Analyse und Entwurf. 2. illustrierte Ausgabe.* Spektrum, Akad. Verl. Heidelberg, Berlin, 2011

[Batz u. a. 1994] BATZ, Thomas ; ANHEUSER, Frederik ; HERRMANN, Frank: *Objektorientierte Modellierung von Produktionsprozessen. Eine effiziente Methode zu Bewertung von Produktionsprozessen*. Fraunhofer-Institut für Informations- und Datenverarbeitung, 1994

[Becker 2012] BECKER, Jörg: *Grundsätze ordnungsmäßiger Modellierung. Konzeption und Praxisbeispiele für ein effizientes Prozessmanagement*. Springer-Verlag Berlin Heidelberg, 2012

[Birta u. Arbez 2007] BIRTA, Louis G. ; ARBEZ, Gilber: *Modelling and Simulation*. Springer-Verlag London, 2007

[Böge 2013] BÖGE, Alfred: *Handbuch Maschinenbau. Grundlagen und Anwendung der Maschinenbau-Technik. 21., aktualisierte und überarbeite Auflage 2013*. Springer Vieweg, 2013

[Byoung u. Donghun 2013] BYOUNG, Choi K. ; DONGHUN, Kang: *Modeling and Simulation of Discrete-Event-Systems*. John Wiley and Sons, Inc. Hoboken, New Jersey, 2013

[Fischer 1995] FISCHER, Axel R.: *Objektorientierte Modellierung von Prozessketten. 11*. Universität Karlsruhe, 1995

[Fischer u. Ahrens 1996] FISCHER, Joachim ; AHRENS, Klaus: *Objektorientierte Prozesssimulation in C++*. Addison-Wesley GmbH, 1996

[Forbig 2007] FORBIG, Peter: *Objektorientierte Softwarentwicklung mit der UML. 3. Auflage.* Carl Hanse Verlag München, 2007

[Freitag 2005] FREITAG, Michael: *Informationstechnische Systeme und Organisation von Produktion und Logistik. Band 3.* Gito-Verlag Berlin, 2005

[Gadatsch 2012] GADATSCH, Andreas: *Grundkurs Geschäftsprozess-Management. 7.Auflage. Methoden und Werkzeuge für die IT-Praxis.* Vieweg+Teubner Wiesbaden, 2012

[Gebhardt 2013] GEBHARDT, Karl F.: *Intensivkurs C++*. Duale Hochschule Baden-Württenberg, 2013

[Hedtstück 2013] HEDTSTÜCK, Ulrich: *Simulation diskreter Prozesse. Methoden und Anwendung*. Springer-Verlag Berlin Heidelberg, 2013

[Hitz u. a. 2005] HITZ, Martin ; KAPPEL, Gerti ; KAPSAMMER, Elisabeth ; RETSCHITZEGGER, Werner: *UML at Work. Objektorientierte Modellierung mit UML 2. 3., aktualisierte und überarbeitet Auflage*. dpunkt Verlag Heidelberg, 2005

[Kiess 1995] KIESS, Jan U.: *Objektorientierte Modellierung von Automatisierungssystemen*. Springer-Verlag Berlin, Heidelberg, 1995

[Martin u. Odell 1999] MARTIN, James ; ODELL, James J.: *Objektorientierte Modellierung mit UML: Das Fundament*. Prentice Hall, 1999

[März u. a. 2011] MÄRZ, Lothar ; KRUG, Wilfiried ; ROSE, Oliver ; WEIGERT, Gerald: *Simulation und Optimierung in Produktion und Logistik*. Springer-Verlag Berlin Heidelberg, 2011

[Mertins u. a. 1994] MERTINS, Kai ; SÜSSENGUTH, Wolfram ; JOCHEM, Roland: *Modellierungsmethoden für rechnerintegrierte Produktionsprozesse. Unternehmensmodellierung, Softwareentwurf, Schnittstellendefinition, Simulation*. Carl Hanse Verlag München, Wien, 1994

[Nyhuis 2010] NYHUIS, Peter: *Wandlungsfähige Produktionssysteme*. GITO-Verlag Berlin, 2010

[Oestereich 2009] OESTEREICH, Bern: *Analyse und Design mit UML 2.3. 9., aktualisierte und erweiterte Auflage*. Oldenburg Verlag München, 2009

[Oestereich u. a. 2003] OESTEREICH, Bern ; WEISS, Christian ; SCHRÖDER, Claudia ; WEILKIENS, Tim ; LENHARD, Alexander: *Objektorientierte Geschäftsprozessmodellierung mit der UML. 1. Auflage*. dpunkt Verlag Heidelberg, 2003

[Patig 2006] PATIG, Susanne: *Die Evolution von Modellierungssprachen*. Frank & Timme Gmbh. Verlag für wissenschaftliche Literatur, 2006

[Priese u. Wimmel 2008] PRIESE, Lutz ; WIMMEL, Harro: *Petri-Netze. 2. Auflage*. Springer-Verlag Berlin Heidelberg, 2008

[Rumpe 2011] RUMPE, Bernhard: *Modellierung mit UML. 2. Auflage*. Springer-Verlag Berlin Heidelberg, 2011

[Rupp u. a. 2012] RUPP, Chris ; QUEINS, Stefan ; SOPHISTEN die: *UML 2 glasklar. 4. Auflage. Praxiswissen für die UML-Modellierung*. Carl Hanse Verlag München, 2012

[Schneeweiss 2013] SCHNEEWEISS, Ralf: *Moderne C++ Programmierung. Klassen, Templates, Design Patterns. 2. Auflage*. Springer Vieweg Berlin, Heidelberg, 2013

[Staud 2010] STAUD, Josef L.: *Unternehmensmodellierung. Objektorientierte Theorie und Praxis mit UML 2.0*. Springer-Verlag Berlin Heidelberg, 2010

[Vetter 1998] VETTER, Max: *Objektmodellierung. Eine Einführung in die objektorientierte Analyse und das objektorientierte Design. 2., neubearbeitete und erweiterte Auflage*. B. G. Teubner Stuttgart, 1998

[Westkämper 2006] WESTKÄMPER, Engelbert: *Einführung in die Organisation der Produktion*. Springer-Verlag Berlin, Heidelberg, 2006

[Westkämper u. a. 2013] WESTKÄMPER, Engelbert ; SPATH, Dieter ; CONSTANTINESCU, Carmen ; LENTES, Jochem: *Digitale Produktion*. Springer-Verlag Berlin, Heidelberg, 2013

[Wöllhof 1995] WÖLLHOF, Konrad: *Objektorientierte Modellierung und Simulation verfahrenstechnischer Mehrproduktanlagen*. Verlag Sharker, 1995

A. Anhang

A.1. Klassenimplementierung in C++

A.1.1. Codedarstellung in C++

Ein Objekt des erstellten Modells soll beispielhaft in der Programmiersprache C++ dargestellt werden. Um ein Objekt darzustellen, muss als Erstes eine Klasse definiert werden. Zur Illustration der Implementierung einer Klasse mit C++ soll im Folgenden die Klasse *Auftrag* (vgl. **Abschnitt 5.2.3**) aus dem Objektmodell dieser Arbeit in einen Programmcode transformiert werden. Dazu müssen unter anderem die Eigenschaften (Attribute) und Operationen dieser Klasse erfasst werden, was aufgrund der grafischen Darstellung der Klasse Auftrag durch die UML nicht all zu schwer fällt. Die Darstellung des Beispielcodes, der im Weiteren als Beschreibungsgrundlage dient, ist in **Listing A.1** zu erkennen.

Um eine Klasse in C++ zu deklarieren, wird das Schlüsselwort `class` verwendet und der Name der Klasse wird angefügt. Wie in **Abschnitt 4.2.5** auf Seite 38, beginnt der Name im Sinne der Objektorientierung mit einem Großbuchstaben. Das Schlüsselwort wird nur bei der Deklaration benötigt, nicht aber bei der Definition von Daten [Schneeweiss, 2013]. Als zweiten Schritt werden die Attribute und Operationen deklariert. Da die Implementierung in die Sprache C++ an dieser Stelle nur theoretisch zur Darstellung herangezogen werden soll, wird die Klasse nicht in vollem Umfang, d. h mit all ihren Attributen und Operationen deklariert, sondern nur mit einer begrenzten Auswahl.

Alle Attribute und folgenden Operationen unterliegen sog. Sichtbarkeitsregeln, die festlegen, wann und wer auf die Elemente in einer Klasse zugreifen darf. Durch `public` (Zeile 9) wird ein globaler Zugriff erlaubt. Demgegenüber steht das Keyword `private`, welches den Zugriff außerhalb der Klasse verweigert [Schneeweiss, 2013]. Die einzelnen Attribute in der erstellten Klasse sollen zunächst als `public` deklariert werden, um auf die verschiedenen Attribute zugreifen zu können. Da Operationen als Werkzeug der Kommunikation und Interaktion dienen (s. **Kapitel**:

4.2), werden Operationen ebenfalls als `public` deklariert. Auch von außerhalb der Klasse ist demnach ein Zugriff auf die Operation möglich. Gemeinhin ist eine feine Differenzierung der Sichtbarkeit nötig und Vorteilhaft, soll im Rahmen des hier gezeigten Beispiels jedoch keine Rolle spielen. Für genauere Informationen sei auf Schneeweiss [2013] und Balzert u. a. [2011] verwiesen.

Von Zeile 7 bis 19 wird die Klasse `Auftrag` mit einigen ihrer Attribute und Operationen deklariert. In den Zeilen 9 bis 12 stehen die Attribute, von 14 bis 19 die Operationen. Die ersten beiden Operationen sind der Konstruktor, zum Erzeugen, und der Destruktor, zum Löschen von Objekten. Der Konstruktor führt bei der Erzeugung eines Objektes der Klasse Auftrag die Funktion *setAuftragsnummer()* durch und weist dem Auftrag entsprechende Werte zu. Diese Funktion ist außerhalb der Klasse in Zeile 22 definiert. Um auf die Klassenfunktion zuzugreifen, muss dem Compiler der Name der Klasse und der Funktion mitgeteilt werden. Letztgenanntes erfolgt durch zwei an den Namen der Klasse angehängte Doppelpunkte, gefolgt vom Namen der Funktion. Die Funktion *setAuftragsnummer()* vergibt in diesem Szenario beispielhaft festgeschriebene Nummern. Die Nummern könnten jedoch auch abhängig von der Anzahl der Auftragsobjekte dynamisch vergeben werden, was programmiertechnisch entsprechend umgesetzt werden muss. In Zeile 26 bis 27 ist die Funktion *erfasseAuftragsdaten()* definiert, die als Rückgabewert die entsprechenden Werte zurückliefert.

In den Zeilen 33 wird ein Objekt der Klasse Auftrag instantiiert. Erst wird der Klassenname genannt und anschließend der Name des Objektes, hier `Auftrag1`. Wie oben formuliert, werden durch den Aufruf des Konstruktors die Auftragsdaten festgelegt. In Zeile 34 lässt sich auf diese Daten mit einer entsprechenden Funktion zugreifen, die mit dem Namen des Objektes gefolgt von einem Punkt und dem Funktionsnamen aufgerufen wird. Das Ergebnis lässt sich in einer Variablen *a* speichern. Abschließend ist beispielhaft der Aufruf des Destruktor in Zeile 37 dargestellt. Im Zustandsdiagramm aus **Abbildung 6.4** ist die Operation *erfasseAuftragsdaten()* im Modell zu erkennen.

Durch das Beispiel wird deutlich, welchen Vorteil die konsistente Verwendung der Objektorientierung mit sich bringt. Ebenso ist die Implementierung eines in der UML erstellten Modells problemlos möglich. Die subjektive Interpretation des gesamten Modelles wird durch die eindeutigen und grafischen Symbolnationen begrenzt jedoch nicht vermieden.

Listing A.1: Implementierung der Klasse Auftrag

```
1  #include <iostream>
2  #include <string>
3  #include <ctime>

5  using namespace std;

7  class Auftrag
8  {
9      public:
10         int Auftragsnummer, Kundennummer; //Attribute
11         string Positionen;
12         time_t DateBestellung;
13     public:
14         Auftrag(bool){                //Konstruktor
15             setAuftragsnummer();
16         }
17         ~Auftrag();                      //Destruktor
18         void setAuftragsnummer();        // Operation
19         int erfasseAuftragsdaten();

21 };
22     void Auftrag::setAuftragsnummer(){ //Definiton der Operationen
23     Auftragsnummer=1;
24     Kundennummer=11;
25 }
26     int Auftrag::erfasseAuftragsdaten(){
27         return Auftragsnummer, Kundennummer;
28     }

30 int main()
31 {
32     int a;
33     Auftrag Auftrag1(true);            // Instantiierung
34     a=Auftrag1.erfasseAuftragsdaten();
35     // ruft Operation des Objektes auf

37     delete Auftrag1;  // ruft den Destruktor auf

39     return 0;
40 }
```

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matr.-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/Masterarbeit* mit dem Titel

_____

_____

_____

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

_____ _____

Ort, Datum Unterschrift