Bachelorthesis

Intelligentes Datenmanagement in der Produktion
Ein objektorientierter Modellierungsansatz

Marvin Rosian

Wirtschaftsingenieurwesen
Matrikel-Nr.: 149966

Erstprüfer: Univ.-Prof. Dr.-Ing. Markus Rabe
Zweitprüfer: M.Sc. Joachim Hunker

Betreuer bei der Leopold KOSTAL GmbH & Co. KG:
Dipl.-Ing. Helge Minolla (Gruppenleiter Projekt Koordination Einzelteile)
Dr.-Ing. Martin Wagner (Qualitätsstrategie / -strukturen / -prozesse)

Dortmund, 02.07.2018

Vorwort

Die vorliegende Bachelorthesis entstand in Zusammenarbeit mit dem Lehrstuhl für IT in Produktion und Logistik an der technischen Universität Dortmund und der Leopold KOSTAL GmbH & Co. KG. Die ursprüngliche Idee zum Verfassen dieser Thesis entstand in der Unternehmenszentrale der Leopold KOSTAL GmbH & Co. KG in Lüdenscheid in der Abteilung „Production Engineering Einzelteile/ Werkzeuge“ im Abteilungsbereich der Projektkoordination für Einzelteile. Das Fallbeispiel wurde zudem in Zusammenarbeit mit dem Fachbereich des Unternehmens für Qualitäts-strategien/ -strukturen und Prozesse entwickelt.

Fakultät Maschinenbau
Lehrstuhl für IT in Produktion und Logistik
Univ.-Prof. Dr.-Ing. Markus Rabe
Leonhard-Euler-Straße 5
44227 Dortmund
Tel.: +49 (231) 755-8021

Leopold KOSTAL GmbH & Co. KG
An der Bellmerei 10
58513 Lüdenscheid
Tel.: +49 (2351) 160

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CAD	Computer Aided Design
ERM	Entity Relationship Modell
GoM	Grundsätze ordnungsmäßiger Modellierung
KOSTAL	Leopold Kostal GmbH & Co. KG
OEM	Overall Equipment Manufacturer
OOSE	Object Oriented Software Engineering
OMG	Object Management Group
PNG	Portable Network Graphic
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

1 Einleitung

Die Automobilproduktion befindet sich in einem durch Veränderungen geprägten Umfeld. Im heutigen Zeitalter der Globalisierung verfolgen internationale und besonders deutsche Automobilhersteller (Overall Equipment Manufacturer; kurz OEM) in der Produktion eine zunehmend globale Ausrichtung. Mittlerweile produzieren deutsche OEMs zwei von drei PKWs im Ausland (vgl. Verband der Automobilindustrie 2017: S.28). Von diesem Globalisierungstrend sind im gleichen Maße auch die Automobilzulieferer betroffen. Aufgrund der dominanten Beziehung zwischen den OEMs und ihren Zulieferern und dem daraus resultierenden Preis- und Kostendruck auf die Automobilzulieferer (vgl. Kinkel/Zanker 2007: S.2) stehen diese gezwungenermaßen in der Pflicht, ebenfalls ein globales Produktionsnetzwerk aufzubauen. Aus dem „Build where you sell“ – Trend der OEMs leitet sich für die Automobilzulieferer das Prinzip des „Follow your customer“ ab, wodurch Produktionsstätten der Automobilzulieferer oftmals in unmittelbarer Nähe zu den internationalen Werken der OEMs angesiedelt werden müssen (vgl. Bratzel et al. 2015: S.59).

Von der erstmaligen Herstellung von Werkzeugen bis zur finalen Produktion von Einzelteilen an vorbestimmten Produktionsstandorten, durchlaufen Werkzeuge und Einzelteile in der Produktion heutzutage also ein größtenteils globales Netzwerk. In Konsequenz daraus nimmt die Komplexität des Datenmanagements in der Koordination für Einzelteile und Werkzeuge zu. Ungeachtet dessen ist die Professionalität im Umgang mit Daten und Informationen in vielen Unternehmen noch nicht auf dem Niveau der klassischen Produktionsfaktoren wie Betriebsmittel, Arbeit oder Kapital (vgl. Dippold et al. 2005: S.10f.). Automobilzulieferer, die vor 20 Jahren noch relativ simple Strukturen zur Aufnahme und Verwaltung ihrer Daten genutzt haben, stehen heutzutage vor der Aufgabe, über die Landesgrenzen ihres Firmensitzes hinaus ein global funktionierendes Datenmanagement aufzubauen. Dabei stehen sie vor der Herausforderung, Daten aus unterschiedlichen Quellen und an diversen Standorten

weltweit aufzunehmen und zu verwalten. Diese Aufgabe wird durch den Wandel der Wertschöpfungsmuster in der Automobilindustrie (vgl. Hab/Wagner 2017: S.4) zusätzlich erschwert. Durch die Verschiebung des Wertschöpfungsumfangs vom OEM zum Zulieferer entstehen größere und vielfältigere Datenmengen, die aufgenommen und verwaltet werden müssen. Die expansive Modellpolitik der Automobilhersteller (vgl. Hab/Wagner 2017: S.8) gepaart mit einer erheblichen Steigerung ihrer Innovationsaktivitäten (vgl. Bratzel et al. 2015: S.76) haben ebenfalls für eine Steigerung der Datenmenge und Datenvielfalt gesorgt. Zur erfolgreichen Koordination müssen Daten an verschiedenen Produktionsstandorten und aus unterschiedlichen Fachbereichen des Unternehmens aufgenommen und verwaltet werden. Aus diesem Grund müssen sich die Verantwortlichen für die Koordination von Einzelteile und Werkzeugen in der Produktion von veralteten, abteilungsinternen Konzepten zum Koordinieren ihrer Daten loslösen und anfangen abteilungs- und standortsübergreifend zu denken.

Ziel dieser Arbeit ist die Entwicklung eines objektorientierten Datenmodells im Bereich der produktionsübergreifenden globalen Koordination von Einzelteilen und Werkzeugen der Leopold Kostal GmbH & Co. KG, einem großen deutschen Automobilzulieferer. Das zu entwickelnde Datenmodell soll den Grundstein für ein intelligentes Datenmanagement zur Aufnahme und Verwaltung aller anfallenden Daten im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen in der Produktion und allen dazugehörigen Bereichen legen. Intelligentes Datenmanagement bedeutet in diesem Zusammenhang, auf der Basis einer durchdachten Vorgehensweise den heutigen Ansprüchen im Umgang mit Daten im Unternehmen gerecht zu werden. Durch intelligentes Datenmanagement soll langfristig die Datenqualität verbessert und das Entstehen von Insellösungen zur Koordination von Daten an den verschiedenen Unternehmensstandorten und unter den Mitarbeitern verhindert werden. Als Unterziele dieser Arbeit werden zunächst der Datenbegriff und das Datenmanagement im Unternehmen genauer definiert.

Dabei soll insbesondere der Bereich der Datenmodellierung als eine der klassischen Funktionen des Datenmanagements im Unternehmen genauer erörtert werden. Als Basis für das zu entwickelnde Datenmodell wird außerdem als weiteres Unterziel dieser Arbeit die objektorientierte Modellierungssprache Unified Modelling Language (UML) genauer vorgestellt. Darüber hinaus wird im Kontext der Datenmodellierung auch die Referenzmodellierung aufgegriffen. Dabei wird aufgrund des Themas dieser Thesis im speziellen auf die objektorientierte Referenzmodellierung eingegangen. Als letztes Unterziel wird dann auf der Grundlage des unternehmensspezifisch zu entwickelnden Datenmodells und des angeeigneten Wissens zur Referenzmodellierung ein erster grober Ansatz für ein objektorientiertes Referenzmodell für die globale Aufnahme und Verwaltung von Daten im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen erstellt.

Die vorliegende Arbeit gliedert sich in sieben Kapitel. *Einleitend* wir die Motivation zum Verfassen dieser Arbeit vorgestellt und die Zielsetzung definiert. Im *zweiten Kapitel* werden der Datenbegriff und der Begriff des Datenmanagements thematisiert. An dieser Stelle wird auch eine genauere Definition von intelligentem Datenmanagement gegeben und das Thema Datenmodellierung des darauffolgenden Kapitels wird als Bestandteil des Datenmanagements eingeordnet. Weiterhin wird die Datenqualität als übergeordnetes Ziel des Datenmanagements anhand ihrer Qualitätskriterien genauer definiert. Das *dritte Kapitel* befasst sich dementsprechend mit der Datenmodellierung. Hier werden zunächst grundsätzliche Begriffe in Bezug auf Datenmodellierung erörtert. An dieser Stelle werden auch der allgemeine Modellbegriff und die Grundsätze ordnungsgemäßer Modellierung thematisiert. Danach wird die Unified Modeling Language als objektorientierte Modellierungssprache genauer vorgestellt, da diese in den folgenden Kapiteln zur Modellierung verwendet wird. Als letztes wird im dritten Kapitel noch zusätzlich die Referenzmodellierung genauer betrachtet. Auch hier wird der Fokus auf objektorientierte Modellierungstechniken gelegt, da diese im fünften Kapitel zum

Einsatz kommen. Im *vierten Kapitel* wird das Datenmodell im Unternehmen entwickelt. Dazu wird zunächst das Unternehmen Leopold Kostal GmbH & Co. KG vorgestellt und die Situation in der Projektkoordination für Einzelteile beschrieben. Darauffolgend wir das eigentliche Datenmodell im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen modelliert. Das *fünfte Kapite*l befasst sich mit der Erstellung eines Referenzmodells im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen. Hier wird zunächst die besondere Ausgangssituation dieser Referenzmodellierung vorgestellt und die Vorgehensweise zur Referenzmodellerstellung beschrieben. Darauf aufbauend wird dann das Referenzmodell entwickelt. Im *sechsten Kapitel* werden das Datenmodell im Unternehmen und das Referenzmodell abschließend kritisch hinterfragt. An dieser Stelle werden auch weitere Gestaltungsempfehlungen in Hinblick auf ein ganzheitliches und damit intelligentes Datenmanagement formuliert. Im *siebten und letzten Kapitel* werden die wichtigsten Punkte der Arbeit nochmal zusammengefasst und es wird ein abschließender Ausblick gegeben.

2 Daten und Datenmanagement

Im heutigen Zeitalter der Informations- und Wissensgesellschaft ist der Datenbegriff aktueller denn je. Durch aktuelle Ereignisse wie dem Facebook-Datenskandal hat der Datenbegriff sowohl in der Politik als auch unter der Bevölkerung zusätzlich an Aufmerksamkeit gewonnen. Die Bedeutung von Daten insbesondere für Unternehmen ist schon seit längerer Zeit ein geläufiges Thema. Die Entwicklung, weg von der Industrie- hin zur Informations- und Wissensgesellschaft, hatte schon vor über 10 Jahren für Datenexperten wie Dippold et al. (vgl. 2005: S.1) maßgebliche Auswirkungen auf den Umgang mit Daten und Informationen innerhalb von Unternehmen. Heutzutage hat der Umgang mit Daten für Unternehmen durch Trends, wie die fortschreitende Technisierung, die zunehmende Integration von IT in Geschäftsprozessen und die steigende Einbeziehung unternehmensexterner Daten im eigenen Unternehmen weiter an Wert zugenommen (vgl. Apel et al. 2015: S.vii).

2.1 Daten

2.1.1 Terminologie

Krcmar (vgl. 2010: S.15f.) ordnet Daten allgemein als Zeichen ein, die in einen regelbasierten Zusammenhang gebracht wurden. In dieser rohen Form sind sie aber für den Endverbraucher noch nicht von belangen. Erst wenn Daten mit zusätzlichem Kontext in Verbindung gebracht werden, gewinnen sie an Bedeutung und werden so zu Informationen (vgl. Abbildung 1). Daten bilden also das Fundament für Informationen. Allerdings sind Daten an sich nicht wertschöpfend und so gesehen wirtschaftlich erst einmal nur bedingt interessant für Unternehmen. Sie können vielmehr als „ein Baustein für ein höheres Produkt“ (Stahl/Staab 2017: S.32) gesehen werden. Erst wenn aus Daten gewonnene Informationen in Entscheidungen und Handlungen umgesetzt werden, entsteht Wertschöpfung für das Unternehmen (vgl. Dippold et al. 2005: S.4). Krcmar (vgl.

2010: S.16) spricht in dem Zusammenhang von Wissen, das durch die Vernetzung von Informationen mit anderen Informationen entsteht.

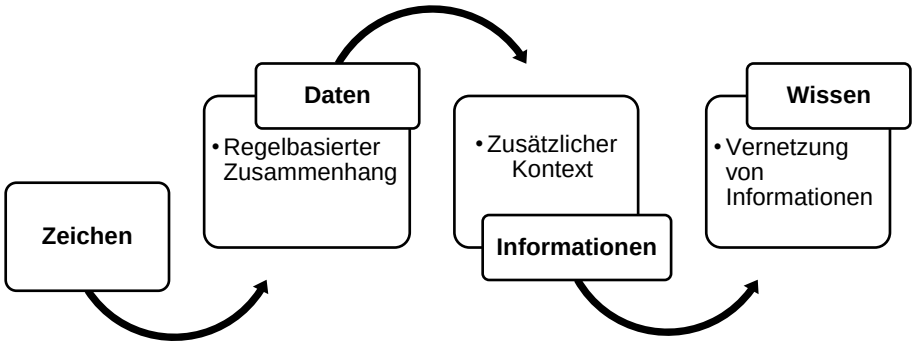


Abbildung 1: Einordnung des Datenbegriffs (nach Krcmar 2010)

2.1.2 Datentypisierung

Daten können abhängig von ihrer Zeichenart, Erscheinungsform, Formatierung und ihrem Verwendungszweck in diverse Unterkategorien klassifiziert werden (vgl. Mertens et al. 2017: S.36f.). Daten bzw. Datenbestände lassen sich dabei grundsätzlich in Stammdaten und Bewegungsdaten typologisieren (vgl. Mertens 2013: S.40). *Stammdaten* sind gemäß ihrem Namen Daten, die als Basis (Stamm) für weitere Datenbestände genutzt werden. Dementsprechend werden sie über einen längeren Zeitraum nur selten verändert. Sie können in betriebswirtschaftliche Daten (z.B. Personalstammdaten wie Namen und Adressen) und in technische Daten (z.B. Einzelteilstammdaten wie Einzelteilnummer und Einzelteilindex) unterteilt werden (vgl. Abbildung 2). Erst Stammdaten machen die Verknüpfung weiterer Daten möglich (vgl. Stahl/Staab 2017: S.53) und sind aus diesem Grund auch im späteren Verlauf dieser Arbeit bei der Entwicklung des Modells von zentraler Bedeutung. *Bewegungsdaten* werden definiert als „vorgangsbezogene Daten, die kurzlebig sind und bestimmten Stammdaten zugeordnet werden“ (SAP o.J.). In Abhän-

gigkeit ihrer Typologie untergliedert Mertens (vgl. 2013: S.40f.) Bewegungsdaten in Vormerkdaten, Transferdaten und Archivdaten. Vormerkdaten existieren nur bis zu einem bestimmten Zeitpunkt. Tritt dieser vordefinierte Zeitpunkt ein, werden sie nicht mehr benötigt und dementsprechend gelöscht. Transferdaten zeichnen sich dadurch aus, dass sie von einem Programm generiert oder bearbeitet wurden und danach zu einem anderen Programm transferiert werden. Archivdaten sind charakterisiert als abgespeicherte Vergangenheitswerte.

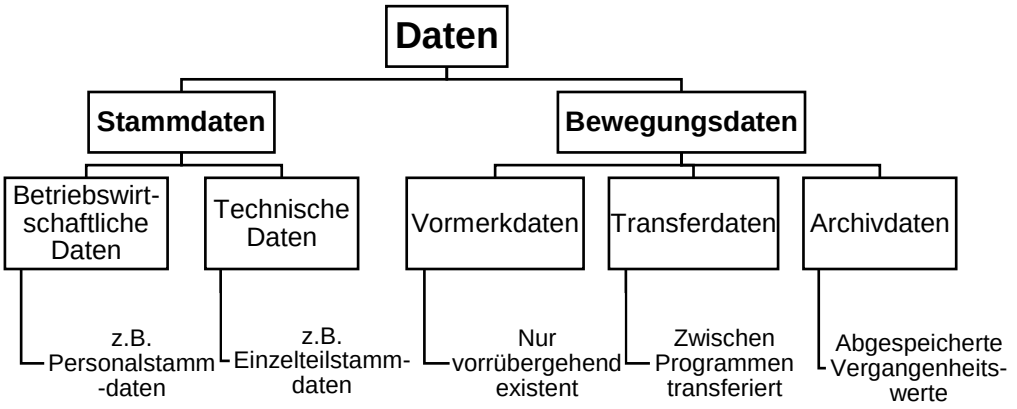


Abbildung 2: Typologie von Daten (nach Mertens 2013)

2.2 Intelligentes Datenmanagement

Unter Datenmanagement werden „alle Prozesse, welche der Planung, Beschaffung, Organisation, Nutzung und Entsorgung der Unternehmensressource Daten dienen […]“ (Dippold et al. 2005: S.21) verstanden. Darüber hinaus hat das Datenmanagement die „Bereitstellung und Nutzung der Daten im Unternehmen“ (Krcmar 2010: S.129) als Ziel. Das Datenmanagement ist also dafür verantwortlich, dass Daten in definierter und abgestimmter Form aufgenommen, verarbeitet und bereitgestellt werden, um dann von Informations- und Wissensträgern im Unternehmen genutzt werden zu können. Dadurch entsteht letztendlich auch direkt und indirekt Wertschöpfung. Versuche den Begriff der Intelligenz zu definieren gibt es im Überfluss. Im Duden wird Intelligenz definiert als „Fähigkeit

[des Menschen], abstrakt und vernünftig zu denken und daraus zweckvolles Handeln abzuleiten“. Das Datenmanagement an sich kann natürlich weder denken noch handeln. Angesprochen sind die Personen im jeweiligen Unternehmen, die für das Datenmanagement verantwortlich sind, sofern diese Personen vorhanden sind. Es ist also die Aufgabe dieser Personen, auf der Basis von durchdachten Konzepten und Vorgehensweisen, ein möglichst zweckmäßiges Datenmanagement im Unternehmen zu implementieren, das den heutigen Ansprüchen im Datenmanagement gerecht wird. Ein intelligentes Datenmanagement sollte daher die drei klassischen Funktionen *Datenmodellierung*, *Datenadministration* und *Datenbankadministration* berücksichtigen (vgl. Abbildung 3). Darüber hinaus sollte ein intelligentes Datenmanagement, ebenso wie die drei klassischen Funktionen des Datenmanagements, das übergeordnete Ziel der Verbesserung der Datenqualität verfolgen. Die vorliegende Arbeit konzentriert sich aufgrund des breiten Spektrums des Datenmanagements auf den Bereich der Datenmodellierung. Daher wird in dieser Arbeit eher der Grundstein gelegt, auf dem ein intelligentes Datenmanagement aufbauen kann.

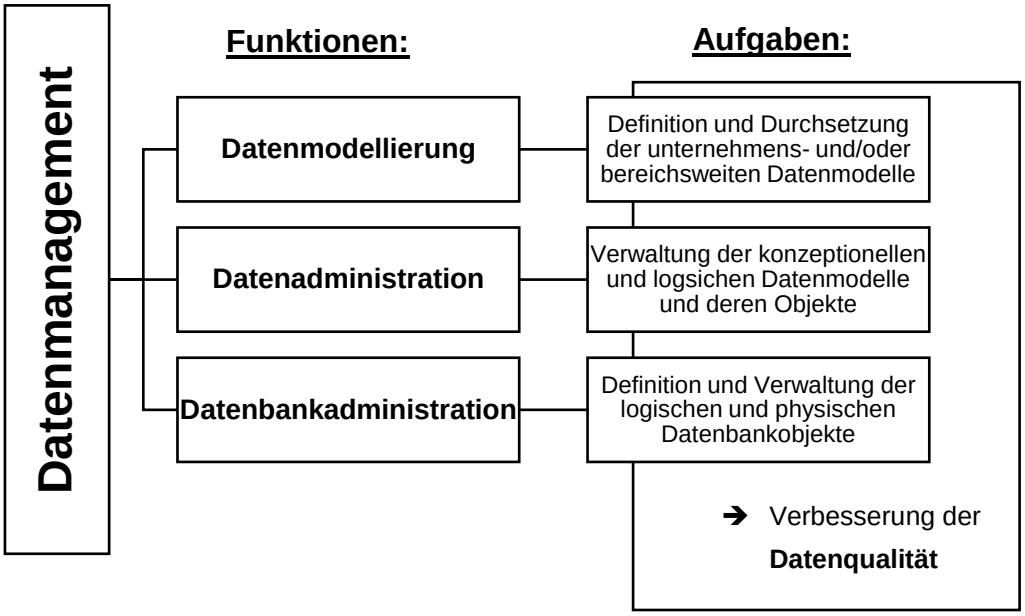


Abbildung 3: Funktionen im Datenmanagement nach (Dippold et al. 2005)

2.3 Datenqualität

Eine der größten Herausforderungen im Datenmanagement besteht in der „Sicherstellung der Konsistenz und der Qualität der Daten“ (Mertens et al. 2017: S.49). Umgekehrt ist das Erreichen der Ziele des Datenmanagements also in hohem Maße von der Datenqualität abhängig. Nur Daten, die nach bestimmten Qualitätskriterien aufgenommen, verarbeitet und bereitgestellt werden, können effizient im Unternehmen genutzt werden. Nach Würthele (2003: S.21) wird Datenqualität definiert als „Mehrdimensionales Maß für die Eignung von Daten, den an ihre Erfassung/Generierung gebundenen Zweck zu erfüllen". Durch Qualitätskriterien muss also sichergestellt werden, dass Daten möglichst effizient ihren eigentlichen Zweck erfüllen. Da der Zweck der Daten von den Bedürfnissen der Nutzer abhängig ist und somit je nach Unternehmen, Situation und Zeitpunkt der Betrachtung unterschiedlich sein kann, ist es nicht sinnvoll ein allgemein gültiges Schema für Qualitätskriterien universell anzuwenden. Um dennoch die effiziente Nutzung der Daten zu gewährleisten, sollten Qualitätskriterien je nach Situation aus einem Zusammenspiel von Literaturrecherchen, empirischen Untersuchungen und anhand von praktischen Erfahrungen definiert werden (vgl. Apel et al. 2015: S.7).

Das Erreichen von qualitativ hochwertigen Daten ist also ein komplexer Vorgang. Helfert et al. (vgl. 2001: S.7) unterscheiden in insgesamt 75 häufig genannte Datenqualitätskriterien. Abbildung 4 soll in Anlehnung an Apel et al. und Helfert et al. einen Überblick über die wichtigsten Datenqualitätskriterien in Hinblick auf ein intelligentes Datenmanagement verschaffen.

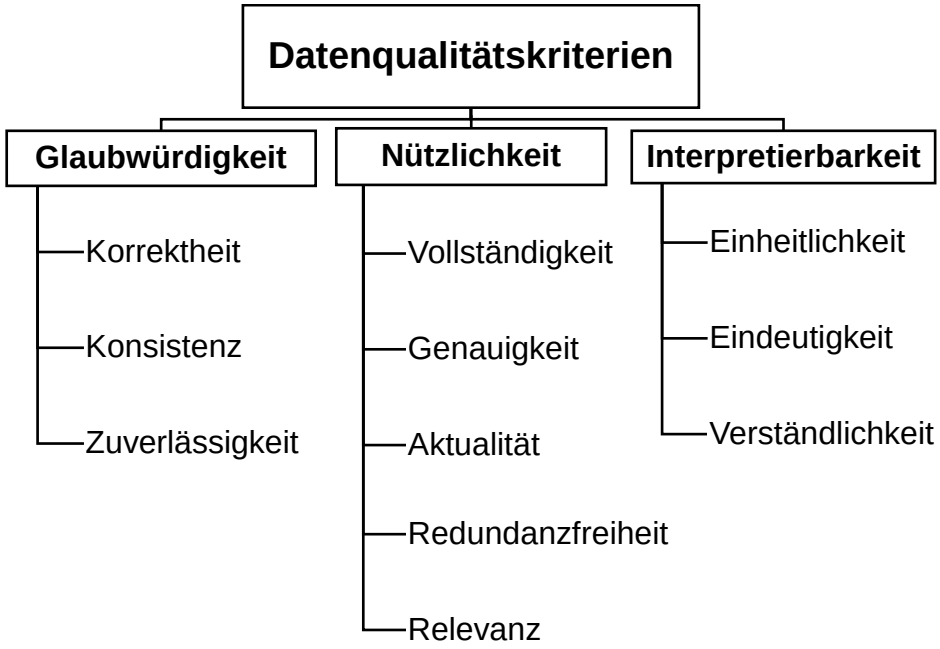


Abbildung 4: Überblick Datenqualitätskriterien nach (Apel et al. 2015 und Helfert et al. 2001)

In der nachfolgenden Tabelle werden die in Abbildung 4 genannten Datenqualitätskriterien zum besseren Verständnis in wenigen Worten definiert.

Tabelle 1: Definition der Datenqualitätskriterien nach (Apel et al. 2015)

Datenqualitätskriterium	Definition
Korrektheit/ Richtigkeit	Daten stimmen mit der Realität überein
Konsistenz/ Integrität	Zueinander widerspruchsfreie Daten
Zuverlässigkeit/ Nachvollziehbarkeit	Die Entstehung der Daten ist nachvollziehbar

Vollständigkeit	Für jede Aufgabenerfüllung ausreichende Abbildung der Realität in Daten
Genauigkeit	Abhängig vom Kontext liegen die Daten in der geforderten Genauigkeit vor
Aktualität	Daten entsprechen immer den aktuellen Zustand der modellierten Welt
Redundanzfreiheit	Keine Duplikate innerhalb der Datensätze
Relevanz	Der Informationsgehalt von Datensätzen erfüllt den jeweiligen Informationsbedarf
Einheitlichkeit	Daten werden fortlaufend gleich abgebildet
Eindeutigkeit	Ein Datensatz muss eindeutig interpretierbar sein
Verständlichkeit	Die Datensätze stimmen in ihrer Begrifflichkeit und Struktur mit den Vorstellungen der Informationsempfänger überein

3 Datenmodellierung

Bisher wurde die Datenmodellierung als eine der klassischen Funktionen des Datenmanagements eingeordnet. Im Gesamtkontext eines Unternehmens bildet sie mit der Prozessmodellierung und der Organisationmodellierung den Kern der Unternehmensmodellierung (vgl. Staud 2010: S.1). Das Ziel der Datenmodellierung ist „die Beschreibung von Unternehmensdaten in einem Datenmodell“ (Krcmar 2010: S.133). Dem *Modellbegriff* werden in unterschiedlichen Wissenschaftsbereichen auch unterschiedliche Bedeutungen zugeordnet. Im Kontext der Wirtschaftsinformatik wird ein Modell oftmals als „Abbildung der Realität oder eines Realitätsausschnittes“ (Schütte 1998: S.53) definiert. Für eine genauere Definition wird häufig auf den Modellbegriff von Stachowiak in seiner 1973 verfassten Modelltheorie verwiesen, der auch heutzutage noch den in den Ingenieurswissenschaften verwendetem Modellbegriff entspricht (vgl. Gutenschwager et al. 2017: S.14). Stachowiak (1973: S.131ff.) hebt in seiner ausführlichen Beschreibung des Modellbegriffs die drei folgenden Hauptmerkmale des *allgemeinen Modellbegriffs* hervor:

Abbildungsmerkmal: Modelle sind […] Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können.
Verkürzungsmerkmal: Modelle erfassen im Allgemeinen nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/ oder Modellbenutzern relevant scheinen.

Pragmatisches Merkmal: Modelle sind ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion a) für bestimmte (…) Subjekte, b) innerhalb bestimmter Zeitintervalle und c) unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen.

Um Modelle genauer zu typisieren können sie nach ihrer Beschreibungsform und dem zu modellierenden Gegenstand unterschieden werden (vgl. Gutenschwager et al. 2017: S.14). In Bezug auf diese Thesis ist der zu modellierende Gegenstand ein Datenmodell, welches anhand eines objektorientierten Klassendiagramms beschrieben wird. Das Datenmodell bzw. die Datenmodellierung hat wiederum die exakte Beschreibung des in der Datenbank abzubildenden Realitätsabschnitts als Aufgabe (vgl. Mertens et al. 2017: S.43). Datenmodellierung und Datenbanken sind also direkt miteinander verknüpft. Datenmodelle dienen der konkreten Einrichtung der Datenbank. Genauer gesagt ist das Erstellen eines Datenmodells gleichzeitig die Voraussetzung, um im späteren Verlauf die Verwaltung von Daten mithilfe eines Datenbanksystems zu ermöglichen (vgl. Krcmar 2010: S.133). Die klassischen Techniken der Datenmodellierung sind die relationale Datenmodellierung und die Entity-Relationship-Modellierung (vgl. Staud 2010: S.347). Diese nicht objektorientierten Techniken haben sich weitgehend als Standard etabliert. Darüber hinaus existieren auch objektorientierte Modellierungsansätze zur Datenmodellierung. In Abschnitt 3.1 und 3.2 werden zunächst die Grundelemente der beiden klassischen Modellierungstechniken erläutert. In Abschnitt 3.3 wird dann die Unified Modelling Language (UML) als Sprache der objektorientierten Modellierung genauer vorgestellt.

Um die Qualität eines Modells sicherzustellen und zu verbessern wird vermehrt auf die *Grundsätze ordnungsmäßiger Modellierung* (GoM) von Becker et al. verwiesen (vgl. Schütte 1998; Schwegmann 1999; Schlagheck 2000; vom Brocke 2003). Die übergeordnete Zielsetzung der 1995 erstmals vorgestellten GoM ist es Gestaltungsempfehlungen zu entwickeln, anhand derer eine qualitätsgerechte und vergleichbare Modellierung stattfinden kann (vgl. Becker et al. 2012: S.31). Als Kriterien dafür werden allgemeine Grundsätze der *Richtigkeit*, *Relevanz*, *Wirtschaftlichkeit*, *Klarheit*, *Vergleichbarkeit* und des *systematischen Aufbaus* von Modellen aufgestellt. Tabelle 2 soll einen groben Überblick über die aktuellste Fassung der GoM verschaffen.

Tabelle 2: Grundsätze ordnungsmäßiger Modellierung (nach Becker et al. 2012)

Grundsätze	**Definition**
Richtigkeit (syntaktisch/semantisch)	*Syntaktische Richtigkeit:* Einhaltung der durch die Modellierungssprache vorgegeben Regeln (formale Korrektheit) *Semantische Richtigkeit:* Konsensorientierte Richtigkeit in Bezug auf das Modell und Teilbereiche des Modells, bis hin zu einzelnen Begriffen
Relevanz	Forderung nur die Sachverhalte zu modellieren, die für den zugrunde liegenden Modellierungszweck relevant sind
Wirtschaftlichkeit	Erreichen eines gegebenen Modellierungsziels mit minimal notwendigen Aufwand
Klarheit	Hervorheben der Relevanz der Verständlichkeit, Anschaulichkeit und leichten Lesbarkeit von Modellen
Vergleichbarkeit	Vergleichbarkeit durch identische Abbildung von in der Realwelt gleichen Abläufen in Modellen mit gleichen und mit unterschiedlichen Modellierungssprachen
Systematischer Aufbau	Sicherstellen der Konsistenz des Gesamtmodells durch eine Vereinheitlichung unterschiedlicher Sichten in den einzelnen Modellen, bei der Beschreibung von Sachverhalten aus der Realwelt

3.1 Relationale Datenmodellierung

Die relationale Datenmodellierung basiert auf relationalen Tabellen (auch Relationen genannt). Die Relationen bestehen aus einer festen Anzahl von Spalten (Attributen) und einer beliebigen Anzahl von Zeilen (Tupeln) (vgl. Abts/Mülder 2017: S.169). Die Anzahl der Attribute bestimmt den Grad einer Relation und die Anzahl der Tupel die Kardinalität. Darüber hinaus schreibt das relationale Datenmodell mit dem Primärschlüssel und dem Fremdschlüssel zwei relationale Invarianten vor. Der Primärschlüssel ist ein Attribut oder eine Kombination von Attributen und stellt sicher, dass in jeder Relation die Tupel eindeutig identifiziert werden können. Der Primärschlüssel wird zudem in der Relation unterstrichen (vgl. Mertens et al. 2017: S.45). Der Fremdschlüssel dient dazu, Beziehungen zwischen Relationen abzubilden. Fremdschlüssel dürfen nach dem Prinzip der referentiellen Integrität nur auf bestehende Primärschlüssel verweisen (vgl. Rahm et al. 2015: S.24). Bei der relationalen Datenmodellierung sollten Beziehungen zwischen Relationen immer über Fremdschlüssel modelliert werden. Die relationale Datenmodellierung beinhaltet noch weitere Funktionen und Formen, auf die aber in dieser grundlegenden Definition nicht weiter eingegangen wird.

In Abbildung 5 werden beispielsweise die zwei Relationen *WERKZEUG* und *PERSON* dargestellt. Die Relation *WERKZEUG* hat den Grad 3 und die Kardinalität 5. Die Relation *PERSON* hat auch den Grad 3 aber die Kardinalität 4. Geeignete Primärschlüssel wären die Attribute *NR* und *ID*. Das Attribut *ID* in Relation *WERKZEUG* wäre darüber hinaus ein solcher Fremdschlüssel, mit dem die verantwortliche Person für ein Werkzeug durch den Verweis auf den Primärschlüssel der Relation *PERSON* repräsentiert wird.

WERKZEUG

NR	ID	ORT
1234	ID1	DE
2584	ID4	EN
7851	ID2	DE
1245	ID1	BUL
7898	ID3	CR

PERSON

ID	NAME	GEBDAT
ID1	Müller	03.04.1987
ID2	Schulz	01.01.1982
ID3	Meier	02.04.1977
ID4	Weber	17.12.1988

Abbildung 5: Beispiele für Relationen

3.2 Entity-Relationship-Modellierung

Die Entity-Relationship-Modellierung ist durch „eine klare Definition und übersichtliche grafische Darstellung gekennzeichnet“ (Mertens et al. 2017: S.43). Mit der Entity-Relationship-Modellierung und dem daraus entstehenden Entity-Relationship-Modell (ERM) lassen sich Objekte und ihre Beziehungen beschreiben. Die Grundelemente der ER-Modellierung sind nach Mertens et al. (2017: S.43) Entities mit ihren Eigenschaften (Attributen) sowie deren Beziehungen (Relationen) untereinander mit den dazugehörigen Attributen. Entities können dabei ein bestimmtes Objekt darstellen oder eine Klasse von gleichen Objekten. Im zweiten Fall spricht man von Entitytyp. Attribute charakterisieren Eigenschaften von Entity- und Beziehungstypen. Ihre konkreten Ausprägungen werden Attributwerte genannt. Ähnlich wie bei dem Primärschlüssel im relationalem Datenmodell werden im ERM die Entities eines Entitytyps durch einen Attributwert (Primärattribut) oder durch eine spezifische Attributwertkombination beschrieben. Eine Beziehung beschreibt eine Verknüpfung bzw. einen Zusammenhang zwischen zwei oder mehreren Entities. Beziehungen können anhand von drei unterschiedlichen Beziehungstypen be-

schrieben werden. Nach der Anzahl der an der Beziehung beteiligten Entities (Kardinalität) unterscheidet man zwischen Eins-zu-Eins-, Eins-zu-Viele und Viele-zu-Viele-Beziehungen (1:1-, 1:n- und m:n-Beziehungen) (vgl. Abts/Mülder 2017: S.165). Bei einer 1:1-Beziehung ist jeder Entity des ersten Entitytyps genau eine Entity des zweiten Entitytyps zugeordnet und umgekehrt. Bei einer 1:n-Beziehung sind jeder Entity des ersten Typs keine, eine, mehrere und bis zu unendlich viele Entities des zweiten Entitytyps zugeordnet. Andersherum ist jede Entity des zweiten Entitytyps genau eine Entity des ersten Typs zugeordnet. Bei einer m:n-Beziehung sind jeder Entity des ersten Entitytyps keine, eine, mehrere und bis zu unendlich viele Entities des zweiten Entitytyps zugeordnet und umgekehrt.

Bei der grafischen Notation werden Entitytypen durch Rechtecke und Beziehungstypen durch Rauten dargestellt und die Symbole werden durch ungerichtete Kanten verbunden, an denen die Komplexität des Beziehungstyps angetragen wird (vgl. Mertens et al. 2017: S.43). Attribute werden durch Ellipsen dargestellt und das/die Primärattribut/e werden wie bei der relationalen Datenmodellierung zusätzlich unterstrichen. Abbildung 6 zeigt die aus Abschnitt 2.4.1 bekannten Relationen, modelliert nach der Entity-Relationship-Methode.

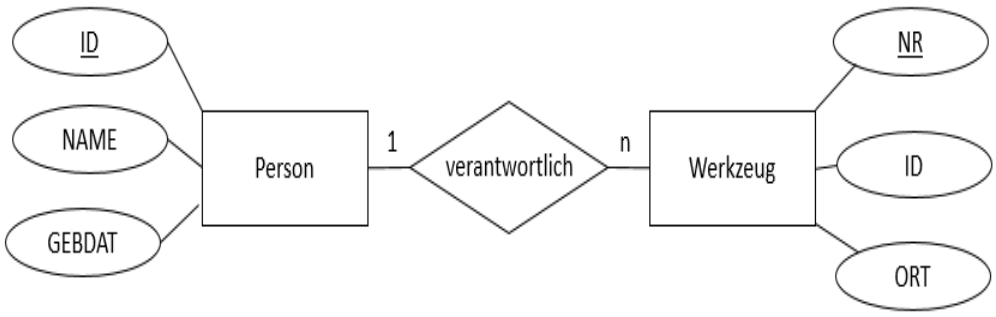


Abbildung 6: Beispiel für ein Entity-Relationship-Modell

3.3 Unified Modelling Language

Im Rahmen objektorientierter Modellierungsansätze zur Datenmodellierung wird in diesem Abschnitt die UML vorgestellt. Die UML hat sich als Standard für objektorientierte Modellierungen durchgesetzt, da mit ihr die vielen verschiedenen objektorientierten Theorieansätze vereinheitlicht, präzisiert und vertieft werden konnten (vgl. Staud 2010: S.8). Da das nachfolgend zu entwickelnde Fallbeispiel und das Referenzmodell anhand der UML modelliert werden, wird die UML in diesem Abschnitt ausführlicher vorgestellt als die klassischen Techniken zuvor.

3.3.1 Einführung in die UML

Die Datenmodellierung wird von den klassischen Modellierungstechniken dominiert. Die UML als objektorientierte Modellierungssprache bietet aber einige Vorteile gegenüber den klassischen Modellierungstechniken. Als generelle Vorteile des objektorientierten Paradigmas werden insbesondere eine hohe Verständlichkeit, Wiederverwendbarkeit, Anpassbarkeit und eine höhere Durchgängigkeit der Softwareentwicklung genannt (vgl. Schwegmann 1999: S.49ff.). Datenmodelle, die mit der UML entworfen werden, bieten daher eine Grundlage auf der auch mit Nicht-Softwareentwicklern diskutiert werden kann (vgl. Vieweg et al. 2012: S.130). Durch eine derart realitätsnahe Modellierung bietet sich die Möglichkeit schon zu einem frühen Zeitpunkt verschiedene Vorstellungen von Entwickler und Anwender in Bezug auf Erwartungen und Anforderungen der Datenmodellierung aufzudecken und beseitigen zu können. Der zusätzlich entstehende Zeitaufwand zur frühzeitigen Abstimmung zwischen Entwickler und Anwender ist in Relation zu einer Modelländerung nach Implementierung des Modells deutlich geringer und somit auch wirtschaftlicher.

Die UML etablierte sich das erste Mal 1997 in der Version 0.9, entstanden aus der Unified Method 0.8 von Rumbaugh/Booch und dem Ansatz des Object-Oriented-Software Engineering (OOSE) nach Jacobsen et al.

(vgl. Rupp et al. 2012: S.5). Bis zur aktuellsten Version 2.5.1, veröffentlicht im Dezember 2017 von der Object Management Group (OMG), wurde der Kreis der Autoren und Firmen, deren Anforderungen in die Erstellung der nächsten Versionen mit einflossen, fortlaufend erweitert. Die kontinuierliche Weiterentwicklung der UML ist nach Rupp et al. (vgl. 2012: S.6) auf die nachfolgenden Gründe zurückzuführen: In den ersten Jahren war es vor allem das Bestreben eine salonfähige Version für die Industrie zu entwickeln. Danach wurden mit den neuen Versionen in erster Linie neue Elemente und Notationen integriert. Zuletzt wurde mit der UML 2 das Konzept verfolgt, die veralteten, zu groß und zu komplex gewordenen UML 1.x-Versionen, durch zeitgemäße Anpassungen und gleichzeitige Komplexitätsreduzierungen zu verbessern

Die UML 2.5.1 besitzt insgesamt 14 Diagrammtypen, die in erster Instanz in Strukturdiagramme und Verhaltensdiagramme klassifiziert werden (vgl. Abbildung 7). Die sieben Verhaltensdiagramme dienen zur Beschreibung der Dynamik eines Systems. Dabei werden vier Verhaltensdiagramme auf einer tieferen Ebene nochmal in Interaktionsdiagramme unterkategorisiert. Diese sind dadurch gekennzeichnet, dass sie sowohl tabellarisch als auch grafisch notiert werden können (vgl. Rupp et al. 2012: S.7). Die sieben Strukturdiagramme modellieren, in Kontrast zu den dynamischen Verhaltensdiagrammen, die statischen und damit unveränderlichen Komponenten eines Systems. Da die Datenmodellierung im späteren Verlauf dieser Thesis durch ein *Klassendiagramm* realisiert wird, werden im Folgenden die wichtigsten Bestandteile des Klassendiagramms genauer definiert.

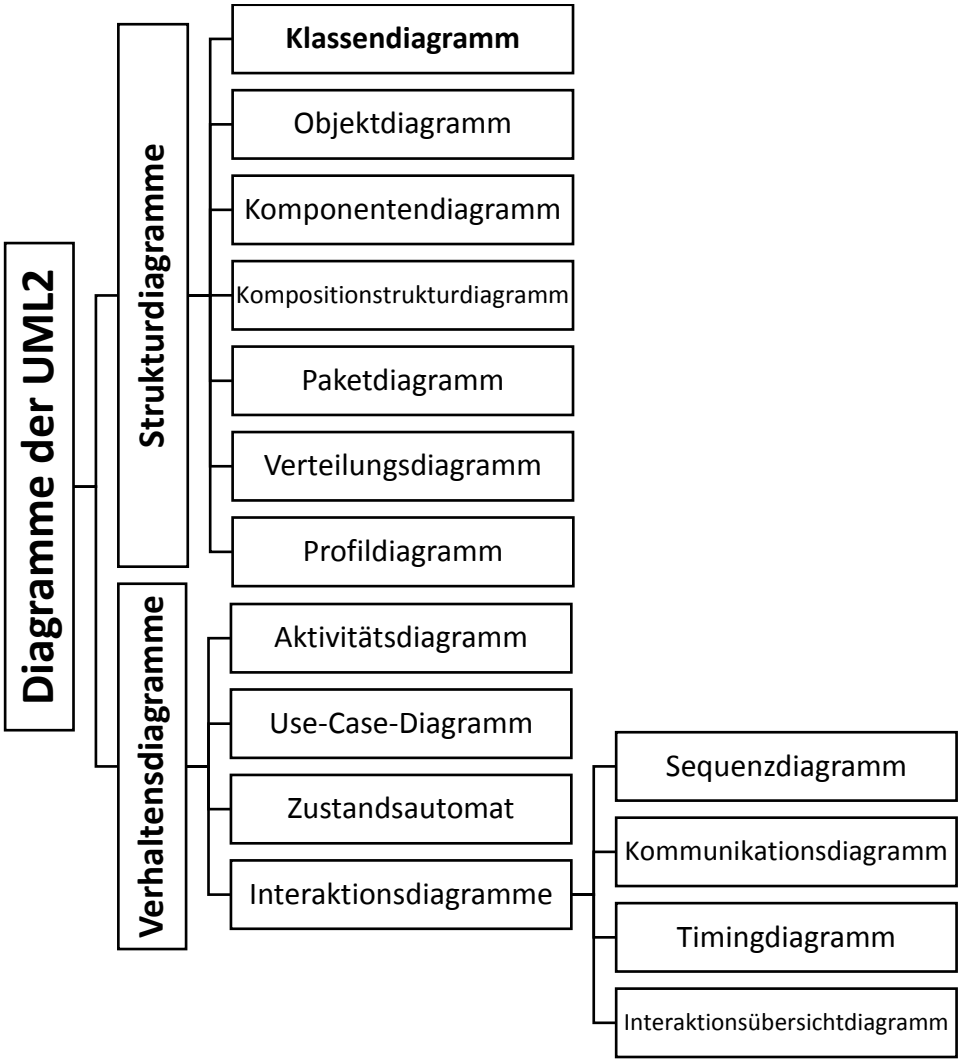


Abbildung 7: Diagramme der UML2 (nach Rupp et al. 2012)

3.3.2 Klassen und Objekte

Das *Klassendiagramm* ist das am häufigsten genutzte Diagramm der UML (vgl. Dobing/Parsons 2006: S.110). Es wird häufig auch als grundlegendes Element der UML oder Kern der Modellierungssprache betitelt, da es die Struktur eines Systems im Detail abbildet und dabei auf die grundlegenden Modellierungskonstrukte der UML zurückgreift und die wichtigsten grafischen Symbole und ihre Bedeutung einführt (vgl. Seemann/Wolff von Gudenberg 2006: S.43 und Rupp et al. 2012: S.107).

Grundsätzlich zeigt ein Klassendiagramm eine Menge von Klassen und deren unterschiedliche Beziehungen zueinander. Eine einzelne Klasse beschreibt wiederum eine Menge von Objekten, die ein gemeinsames Verhalten und gemeinsame Merkmale aufweisen (vgl. Abbildung 8). Eine Klasse bildet also die Schablone zum Erzeugen von Objekten und beschreibt deren Eigenschaften und Verhalten (vgl. Balzert 2007: S.16).

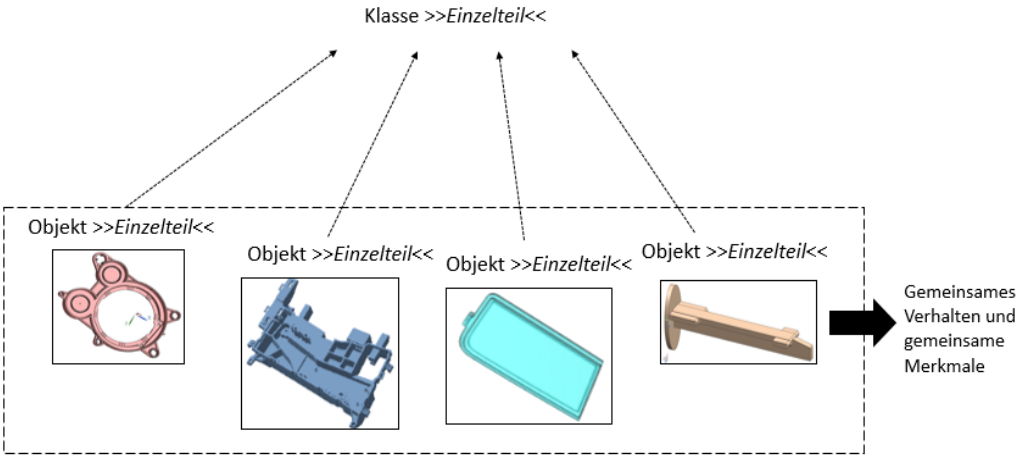


Abbildung 8: Klasse Einzelteil und vier Objekte der Klasse

Die Objekte haben der objektorientierten Modellierung ihren Namen gegeben. Ein Objekt besitzt immer einen bestimmten Zustand, reagiert mit einem definierten Verhalten auf seine Umgebung und hat eine eindeutige Identität (vgl. Balzert 2007: S.11). Im Gegensatz zu Klassen, repräsentieren Objekte konkrete Daten. In Abbildung 9 ist das Objekt *einEinzelteil* der Klasse *Einzelteil* dargestellt. Ein Objekt wird als Rechteck dargestellt und die Bezeichnung eines Objektes wird immer unterstrichen. Im unteren Feld eines Objekts wird der bestimmte Zustand eines Objektes über seine Attribute beschrieben. Im Beispiel hat das Objekt *einEinzelteil* die Attribute *nummer*, *index* und *name* mit den jeweiligen Werteausprägungen. Das Verhalten eines Objektes wird über seine Operationen oder Methoden beschrieben. Diese werden in der unteren Abbildung 9 zunächst nicht angegeben.

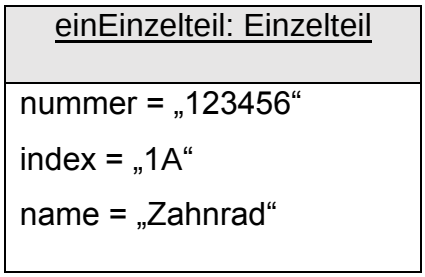


Abbildung 9: Darstellung eines Objektes mit Attributen

3.3.3 Klassen und Attribute

Genauso wie ein Objekt, besitzt auch die Klasse Attribute. Allerdings ist eine Klasse eine Abstraktion, daher werden den Attributen in der Klasse keine konkreten Werte zugeordnet, sondern sie repräsentieren die strukturellen Eigenschaften dieser (vgl. Rupp et al. 2012: S.119). Es ist daher empfehlenswert nur die wirklich wichtigen Attribute anzugeben, die tatsächlich etwas zum Verständnis der Klasse und des Modells beitragen (vgl. Seemann/Wolff von Gudenberg 2006: S.45). Das Klassensymbol ist ein Rechteck, das den fettgedruckten, mittig stehenden und mit einem Großbuchstaben beginnenden Namen der Klasse enthält. Darunter werden in einem eigenen Fach der Name und der Datentyp als die wichtigsten Informationen jedes Attributes eingetragen (vgl. Abbildung 10). Ein Attribut mit einem konkreten Wert, dass sich alle Objekte der Klasse teilen, wird Klassenattribut genannt und durch Unterstreichung hervorgehoben (vgl. Rupp et al. 2012: S.121). In Abbildung 10 wird das Klassenattribut *AnzahlEt* beispielhaft eingeführt. Es soll die Gesamtanzahl der Objekte, in diesem Fall der Einzelteile, in der Klasse darstellen und hat daher einen konkreten Wert.

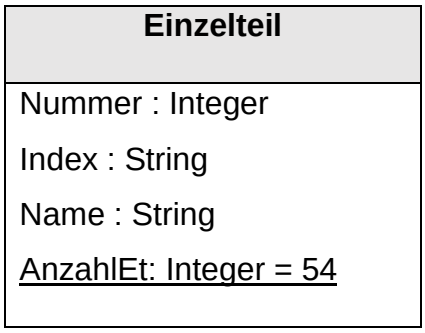
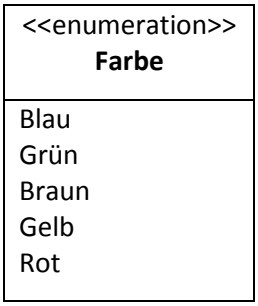


Abbildung 10: Darstellung einer Klasse mit Attributen

Neben dem Namen und dem Datentyp eines Attributes können diese in der UML noch weiter spezifiziert werden. In Tabelle 3 werden die Charakteristika genauer definiert, die ein Attribut in der UML beschreiben.

Tabelle 3: Charakteristika von Attributen nach (Rupp et al. 2012 und Balzert 2007)

CHAR.	BESCHREIBUNG	BEISPIELE
NAME	-Der (kleingeschriebene) Attributname selbst, der in einem beliebigen Zeichensatz frei von jeglichen Beschränkungen gewählt werden kann -Leer- und Sonderzeichen sind auch zugelassen, wenngleich diese vermieden werden sollten	attributname1: … attributname2: … attributname3: …
TYP	-Der Datentyp eines Attributs -Innerhalb eines Klassendiagramms können zur Typisierung der Attribute alle Datentypen und insbesondere durch Klassen eigendefinierte Typen verwendet werden Aufzählungstyp (enumeration type): -Attribut kann nur diskrete Werte annehmen -Für die Modellierung dieses Typs kann das Konzept der Klasse verwendet werden -Die Bezeichnung <<enumeration>> gibt an, dass es sich um einen Aufzählungstyp handelt	Primitive Datentypen: **String** (Zeichenkette) **Integer** (Ganze Zahlen) **Boolean** (True oder False) **Date** (Datumsangabe) enumeration type: 

MULTIPLI-ZITÄT	-Die Multiplizität legt die Unter- und Obergrenze der Anzahl der unter einem Attributnamen ablegbaren Instanzen fest -Die Multiplizität eines Attributs wird in eckigen Klammern notiert -Jede Multiplizitätsangabe besitzt eine Untergrenze, die 0 oder eine beliebige natürliche Zahl sein kann, sowie eine durch zwei Punkte „..“ abgetrennte Obergrenze, die entweder eine beliebige natürliche Zahl größer oder gleich der Untergrenze sein kann oder das Symbol „*“ für eine nach oben nicht festgelegte Anzahl -Wenn für ein Attribut keine Multiplizität angeben ist, wird [1..1] als Vorgabewert angenommen	**[0..1]:** optionales Attribut, d.h. im Objekt ist für das Attribut höchstens ein Wert angegeben **[1..1]:** zwingendes Attribut, d. h. im Objekt wird genau ein Wert angegeben **[0..*]:** optional beliebig, d. h. beliebig viele, Elementanzahl kann auch null sein. Als abkürzende Schreibweise ist der Stern („*“) üblich **[1..*]:** beliebig viele Werte, aber mindestens einer
VORGA-BEWERT	-Erlaubt die Angabe eines festen oder berechenbaren Ausdrucks mit einem Wert, der für das entsprechende Attribut bei Erzeugung des Objekts der Klasse automatisch gesetzt wird	attributname1 : int = 0 attributname2 : int = (attributname1 + 5)
SICHT-BARKEIT	-Definiert, welchen Sichtbarkeits- und Zugriffsrestriktionen ein Attribut unterworfen ist, d. h. welche anderen Systemteile das Attribut „sehen“ können bzw. seinen Wert lesen und schreiben dürfen -Vier verschiedene Sichtbarkeitsmodi werden unterschieden (in Klammern steht die jeweilige Kurznotation)	**public(+):** Jedes andere Element hat uneingeschränkten Zugriff **private(-):** Nur Instanzen der Klasse, die das Attribut definiert, dürfen zugreifen **protected(#):** Nur Instanzen der Klasse, die das Attribut definiert und Instanzen abgeleiteter Klassen dürfen zugreifen **package(~):** Das Attribut ist für alle Elemente, die sich im selben Paket wie die definierende Klasse befinden, sicht- und zugreifbar

EIGEN-SCHAFTS-WERT	-In geschweifte Klammern {} eingeschlossenen Angabe -Benennt besondere Charakteristika des Attributs -Randbedingungen können selbst definiert oder vorgegeben sein	**readOnly:** für Attribute, deren Wert nicht verändert werden darf (Konstanten), z. B. pi : Real = 3.1415 {readOnly} **ordered:** legt fest, dass die Inhalte eines Attributs in geordneter Reihenfolge auftreten, z. B.: {1, 1, 3, 5, 6, 6, 9, 24 } **unique:** legt fest, dass die Inhalte eines Attributs duplikatfrei auftreten, z. B. {3, 1, 500, 4, 2, 16, 89, 24 } **id:** legt fest, dass das Attribut Teil des Identifiers der Klasse ist

3.3.4 Operationen

Die UML bietet darüber hinaus die Möglichkeit, in einem dritten Fach des Klassenrechtecks über Operationen das Verhalten von Objekten zu beschreiben und deren Zustand zu ändern (vgl. Rupp et al. 2012: S.124). Die Zustandsänderung eines Objektes erfolgt dabei allerdings nicht direkt über die Operation. Eine Operation im Klassendiagramm beschreibt nur, was geändert werden soll. Die eigentliche Implementierung erfolgt durch eine Methode (Seemann/Wolff von Gudenberg 2006: S.46). Anhand der Methode können beispielsweise Attributwerte geändert und Objekte erzeugt oder zerstört werden. Eine Operation muss in der UML nur durch ihren Namen spezifiziert werden (vgl. Rupp et al. 2012: S.124). Der Name einer Operation beginnt, gemäß den Style Guidelines der UML, mit einem Kleinbuchstaben und darf beliebig viele Zeichen enthalten. Besteht ein Name aus mehreren Wörtern wird die so genannte Kamelhöcker-Notation angewendet, sodass mit Ausnahme vom ersten Wort jedes weitere Wort mit einem Großbuchstaben beginnt und von Leerzeichen

abzusehen ist. Hinter dem Namen folgt immer eine Parameterliste. Treten keine Parameter auf, besteht die Parameterliste lediglich aus zwei Klammern (). Abbildung 11 zeigt die bereits aus Abbildung 10 bekannte Klasse *Einzelteil*, erweitert um die Operationen *löschen()* und *verändern()*. Werden Operationen von Klassenattributen (im unterem Beispiel *AnzahlEt*) und nicht von individuellen Objekten aufgerufen, werden sie logischerweise auch als Klassenoperationen definiert. Diese sollten allerdings im objektorientierten Entwurf nach Möglichkeit nicht eingesetzt werden (Seemann/Wolff von Gudenberg 2006: S.46). Ähnlich wie bei den Charakteristika für Attribute gibt es auch für Operationen weitere Spezifikationen, die diese genauer beschreiben. Da im späteren Fallbeispiel und im Referenzmodell auf die Definition von Operationen verzichtet wird, wird an dieser Stelle auf weitere Operationsdeklarationen nicht genauer eingegangen.

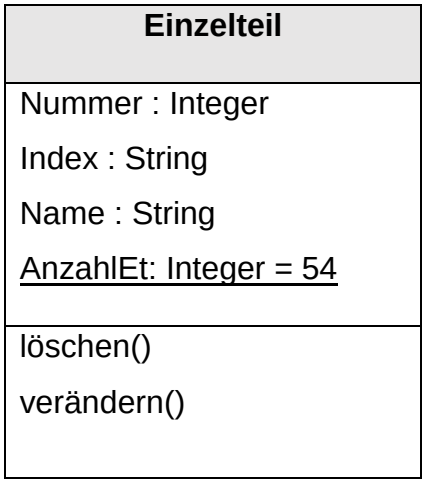


Abbildung 11: Darstellung einer Klasse mit Attributen und Operationen

3.3.5 Assoziationen

Damit Klassen in einem Klassendiagramm miteinander interagieren können, werden sie durch Assoziationen verbunden. Erst Assoziationen ermöglichen das gegenseitige Zugreifen von Klassen auf Attribute und Operationen anderer Klassen (vgl. Rupp et al. 2012: S.144). Um mit Assoziationen aus den vorher isolierten Klassen ein Netzwerk zu erzeugen,

bietet die UML mehrere Notationsmöglichkeiten mit unterschiedlichen Bedeutungen, die im Folgenden vorgestellt werden.

Grundsätzlich können Assoziationen Beziehungen zwischen zwei oder mehreren Klassen beschreiben. Eine Ausnahme bildet die reflexive oder auch rekursive Assoziation, die zwischen Objekte derselben Klasse besteht (vgl. Balzert 2007: S.42). Beziehungen zwischen zwei Klassen werden durch binäre Assoziationen beschrieben und Beziehungen zwischen mehr als zwei Klassen werden durch n-äre Assoziationen beschrieben. Solche mehrstelligen Assoziationen sollten möglichst vermieden werden (Seemann/Wolff von Gudenberg 2006: S.65). Aus diesen Grund werden mehrstellige Assoziationen an dieser Stelle nicht weiter definiert und auch bei der späteren Modellierung nicht eingesetzt. Wenn eine Beziehung mit Information angereichert werden soll, die keinem der beiden Klassen sinnvoll zuzuordnen ist, besteht zudem die Möglichkeit eine eigene Assoziationsklasse zu modellieren (vgl. Rupp et al. 2012: S.158).

Eine Assoziation wird allgemein durch eine durchgezogene Linie zwischen den in Beziehung zu stellenden Klassen beschrieben. In Abbildung 12 sind die gängigen Angaben einer binären Assoziation zu sehen. An den Enden der Assoziation werden die Multiplizität (hier *0..** und *1..**) und oftmals eine Beschriftung (hier jeweils *Rolle*) eingetragen. Die Beschriftung drückt meistens die Rolle der jeweiligen Beziehung aus. Nach der gleichen Notation wie in der vorherigen Charakteristika-Tabelle der Attribute (Tabelle 2) beschreibt die Multiplizität einer Assoziation, wie viele Objekte jeder beteiligten Klassen an der Assoziation teil haben (vgl. Staud 2010: S.43). Auch die Eigenschaften, die an Assoziationsenden geschrieben werden können (hier *{Eigenschaft}*), ähneln in Aufbau und Bedeutung den Eigenschaften aus der Charakteristika-Tabelle der Attribute. Eine Eigenschaft wird immer in geschweiften Klammern {} notiert (vgl. Rupp et al. 2012: S.148). Der Assoziationsname (hier *Assoziation1*) darf fehlen, wenn die Bedeutung der Assoziation offensichtlich ist (vgl. Balzert 2007: S.45). Durch Pfeile an den Assoziationsenden wird die Navigationsrichtung der Assoziation spezifiziert. Im unteren Beispiel ist eine

gerichtete, bidirektionale Assoziation modelliert. Die Objekte von beiden Klassen können also gegenseitig auf sich zugreifen und Objektbeziehungen können in beide Richtungen durchlaufen werden. Assoziationen mit nur einer Navigationsrichtung werden als gerichtet bzw. unidirektional deklariert und Assoziationen mit einem Kreuz an einem Ende dürfen über dieses Ende nicht navigiert werden. Assoziationen ohne Navigationsrichtung werden als unspezifiziert deklariert. Ihre Navigierbarkeit kann global für das gesamte Modell festgelegt werden (vgl. Rupp et al. 2012: S.143).

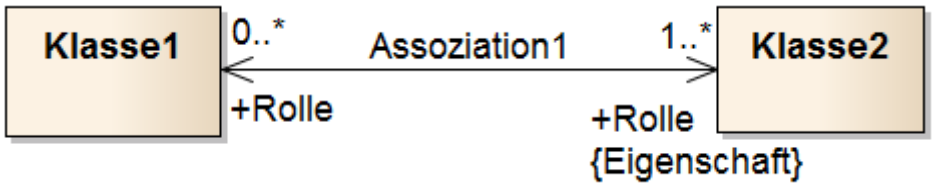


Abbildung 12: Angaben an einer binären Assoziation nach (Rupp et al. 2012)

Neben der einfachen Assoziation gibt es in der UML noch die Möglichkeit Beziehungen über Aggregationen und Kompositionen darzustellen. Eine *Aggregation* ist nach Seemann/Wolff von Gudenberg (vgl. 2006: S.57) durch eine „besteht aus“-Semantik definiert. Teilobjekte einer Teilklasse (hier *Teil*) sind also immer ein Teil von einer Aggregatklasse (hier *Ganzes*), die durch eine Aggregation miteinander verbunden sind (vgl. Abbildung 13). Eine Aggregation wird durch eine transparente Raute am Aggregationsende gekennzeichnet.

Abbildung 13: Aggregation

Eine *Komposition* stellt eine Aggregation mit einer noch strengeren Beziehung dar. In einer Komposition darf ein Teilobjekt nur innerhalb eines Aggregats auftreten und ohne dieses nicht existieren bzw. muss es bei

dessen Vernichtung ebenfalls verschwinden (vgl. Seemann/Wolff von Gudenberg 2006: S.57). Dementsprechend ist die Multiplizität in einer Komposition eingeschränkt. Die Aggregatklasse kann in der Komposition über beliebige Teile verfügen, jedoch darf ein Teil lediglich zu genau einer Aggregatklasse gehören (vgl. Balzert 2007: S.153). Sofern die Multiplizität bei der gefüllten Raute der Komposition nicht konkret angegeben ist, wird sie automatisch als „1“ angenommen (vgl. Abbildung 14).

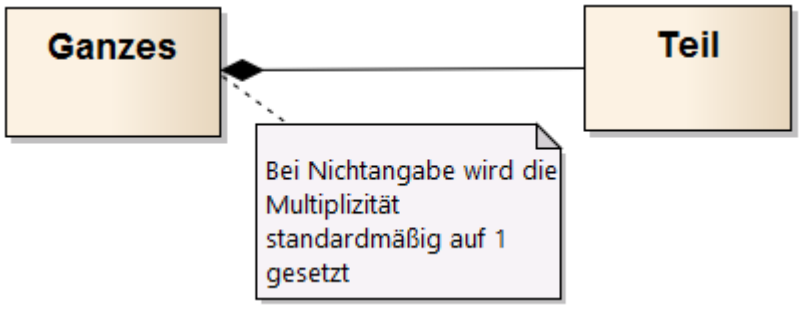


Abbildung 14: Komposition

3.4 Referenzmodellierung

3.4.1 Charakteristika

Eingangs wurde auf der Basis von Stachowiaks Modelltheorie angeführt, dass Modelle Abbildungen oder Repräsentationen natürlicher oder künstlicher Originale sind. *Referenzmodelle* als Ergebnis der Referenzmodellierung zeichnen sich dadurch aus, dass sie nicht das eine Original referenzieren, sondern als Typisierung möglicher (denkbarer) Originale zu sehen sind (vgl. Schütte 1998: S.70). Ein Referenzmodell erhebt den Anspruch auf Gültigkeit über den Einzelfall hinaus, da es eine Klasse von Anwendungsfällen repräsentiert (vgl. Becker/Delfmann 2004: S.1). Aus unternehmensspezifischer Sicht können Referenzmodelle also als Hilfsmittel verstanden werden, um den eigenen Prozess der Entwicklung ei-

nes Modells zu beschleunigen (vgl. Schütte 1998: S.9). Damit ein Referenzmodell diesem Anspruch gerecht wird, muss es bestimmte charakteristische Eigenschaften aufweisen. Solche charakteristischen Eigenschaften von Referenzmodellen, die oftmals auch als *Anforderungen an Referenzmodelle* gesehen werden, gehen über die bereits bekannten GoM (vgl. Tabelle 2) hinaus. Tabelle 4 soll daher einen Überblick über die elementaren Anforderungen speziell an Referenzmodelle verschaffen. Die einzelnen Anforderungen können sich dabei in ihren Zwecken gegenseitig determinieren und sind darüber hinaus auch in Verbindung zu den GoM zu sehen. Beispielsweise fördert die Anforderung des *modularen Aufbaus* eines Referenzmodells gleichzeitig die Anforderung der *Wiederverwendbarkeit* und zudem den *Grundsatz der Klarheit* der GoM.

Tabelle 4: Elementare Anforderungen an Referenzmodelle (nach Schwegmann 1999)

Anforderungen	**Definition**
Wiederverwendbarkeit	Hinreichende inhaltliche und modellierungstechnische Qualität aus Sicht des Referenzmodellverwenders
Allgemeingültigkeit	Hohe Abstraktion der Modelle um für verschiedene Referenzmodellverwender Gültigkeit zu haben oder hinreichende Abbildung von Varianten in Modellen, welche die Sichtweisen unterschiedlicher potentieller Verwender berücksichtigen
Anpassbarkeit	Struktur der Referenzmodelle und die gewählten Darstellungsmittel müssen die Durchführung von ggf. erforderlichen Modifikationen unterstützen

Modularer Aufbau	Anforderung Referenzmodelle für Zwecke einer Komplexitätsreduktion in Teilmodelle zu zerlegen
Akzeptanz	Determiniert durch die Qualität des Referenzmodells und die Bereitschaft des Modellierers von Dritten vorgedachte Ideen wiederzuverwenden

3.4.2 Vorgehensweise zur Erstellung von Referenzmodellen

Im Folgenden wird die Vorgehensweise zur Erstellung von Referenzmodellen nach Schwegmann (vgl. 1999: S.165ff.) zusammengefasst, da diese speziell auf die Erstellung objektorientierter Referenzmodelle ausgerichtet ist.

Wenn die Anforderungen an das jeweils anzufertigende Referenzmodell (vgl. Tabelle 4) präzise spezifiziert wurden, sind im nächsten Schritt die *potentiellen Informationsquellen* zu identifizieren und bezüglich ihrer fachlichen Relevanz auszuwählen und zu bewerten. Als potentielle Informationsquellen nennt Schwegmann (1999: S.166f.) „bereits existierende Referenzmodelle, vorhandene Dokumentationen bestehender Softwaresysteme […], technische oder betriebswirtschaftlicher Literaturquellen und qualifizierte Experten.“ In diesem Zusammenhang sollten für ein aussagekräftiges Referenzmodell idealerweise mehrere Experten nicht nur als Informationsquelle genutzt werden, sondern darüber hinaus auch am Modellierungsprozess an sich beteiligt werden (vgl. Schütte 1998: S.185).

Darauffolgend wird die eigentliche Problemdomäne des zu erstellenden Referenzmodells genauer abgegrenzt und zerlegt. Im Gegensatz zu unternehmensspezifischen Modellen, bei denen die Abgrenzung auf Basis

definierter Projektziele bzw. eruierter Anforderungen erfolgt, ist die Abgrenzung der Problemdomäne bei der Referenzmodellierung komplexer, da die Anforderungen der zukünftigen Referenzmodellverwender nicht oder nur rudimentär bekannt sind (vgl. Schwegmann 1999: S.167). Schwegmann empfiehlt daher eine *Checkliste zur Problembereichsabgrenzung* anzulegen, in welcher die verschiedenen zu berücksichtigenden Aspekte des Referenzmodells aufgeführt und auf Basis der Erfahrungen und Kenntnisse der Referenzmodellersteller über die Anforderungen der Referenzmodellverwender gewichtet werden (vgl. Tabelle 5).

Tabelle 5: Aufbau Checkliste zur Problembereichsabgrenzung (nach Schwegmann 1999)

Aspekt	Beschreibung	Gewichtung	Status
Aus fachlicher Sicht interessante Objekte, Funktionen oder Prozesse einer Problemdomäne	Definition des Aspektes	Absteigende Relevanz des Aspektes von 5 bis 1	• Berücksichtigt • Nicht berücksichtigt • Zum Teil berücksichtigt

Der nächste und gleichzeitig umfangreichste Schritt umfasst die *Identifizierung und Abbildung von Struktur- und Verhaltensmodellen*. Als Grundlage dafür sieht Schwegmann (vgl. 1999 S.168ff.) die Identifizierung und Abbildung von Klassen objektorientierter Referenz-Klassenmodelle. Aufgrund der hohen Bedeutsamkeit einer erfolgreichen Identifizierung und Abbildung von Klassen objektorientierter Referenz-Klassenmodelle insbesondere auch in Bezug auf das in Kapitel 5 zu erstellende Referenzmodell, werden diese beiden Kriterien in dem Abschnitt zur objektorientierten Referenzmodellierung (3.4.3) gesondert untersucht. In Abhängigkeit der angestrebten Konkretisierung und Variantenabdeckung des Referenzmodells können unter einem so genannten Basis-

Referenzmodell zusätzliche Erweiterungsmodelle erstellt werden (vgl. Schwegmann 1999: S.171 ff.). Diese Art der Abbildung von Varianten in spezialisierten Erweiterungsmodellen ist insbesondere bei umfangreichen Referenzmodellen zu empfehlen und wird daher im Rahmen dieser Thesis nicht weiter berücksichtigt. Das Vorgehensmodell zur Erstellung und Verwendung von objektorientierten Referenzmodellen nach Schwegmann (1999), inklusive der Einordnung von (unternehmens-)spezifischen Modellen, wird in Abbildung 15 zur Visualisierung in gekürzter Form nochmals graphisch dargestellt. Anhand der Abbildung wird zudem deutlich, dass die Erstellung und Verwendung von objektorientierten Referenzmodellen ein sukzessiver und sich wiederholender Prozess ist. Dabei sollten auch Erkenntnisse des spezifischen Modells, das auf der Basis des Referenzmodells konstruiert wird, wieder in die Erstellung des Referenzmodells miteinfließen.

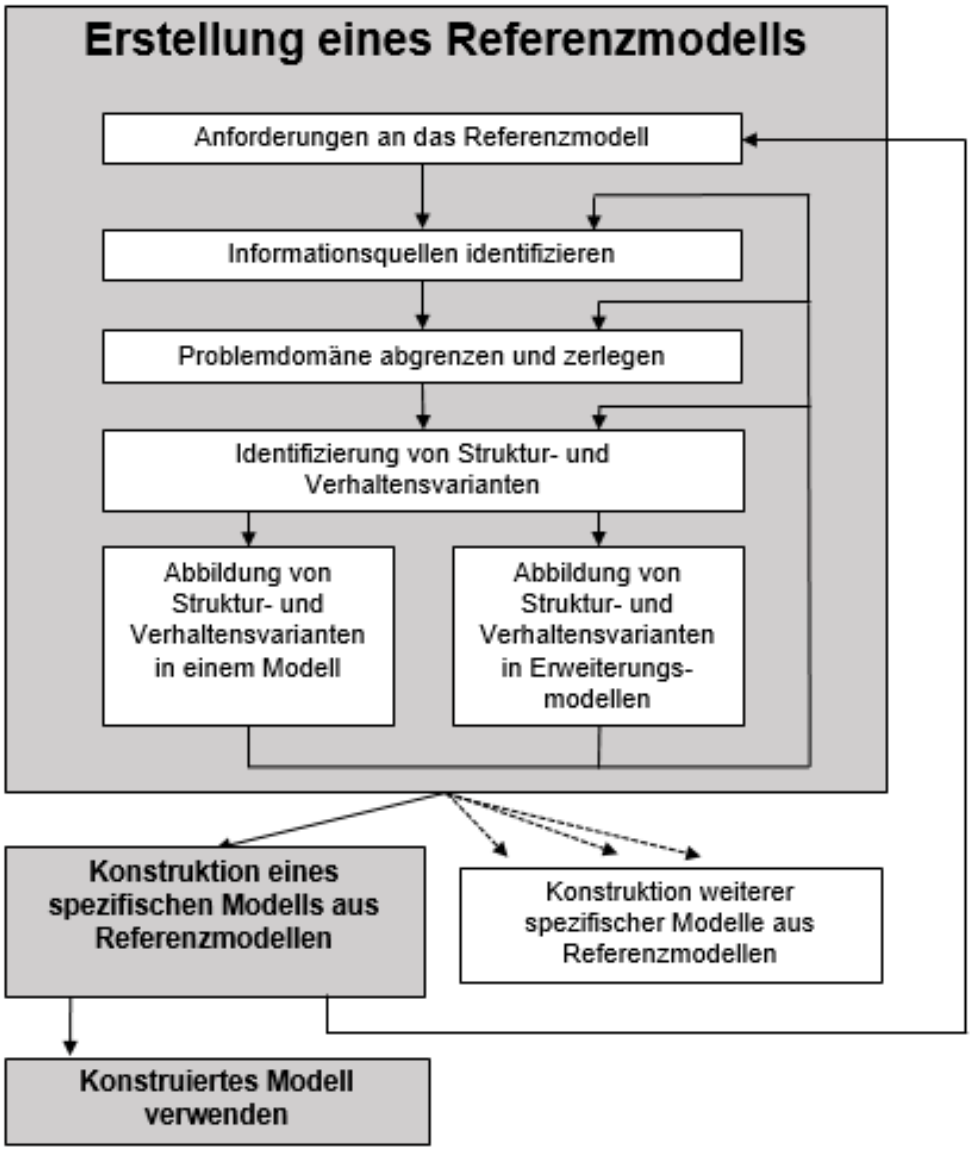


Abbildung 15: Vorgehensmodell zur Erstellung und Verwendung von objektorientierten Referenzmodellen (nach Schwegmann 1999)

Alternativ zu der Vorgehensweise von Schwegmann zur Entwicklung von Referenzmodellen existieren noch andere Vorgehensmodelle. Das Vorgehensmodell von Schütte (vgl. 1998: S.184 ff.) differenziert beispielsweise die Entwicklung eines Referenzmodells in die fünf Phasen *Problemdefinition*, *Konstruktion des Referenzmodellrahmens*, *Konstruktion der Referenzmodellstruktur*, *Komplettierung* und *Anwendung*. Da

Schütte sich im Gegensatz zu Schwegmann ausschließlich auf nicht-objektorientierte Referenzmodelle bezieht, wird dieses Vorgehensmodell zur Referenzmodellierung nicht näher betrachtet.

Schlagheck (2000) bezieht sich in seinem vorrangig auf objektorientierte Referenzmodelle ausgelegten Vorgehensmodell auf die Ansätze von Schütte (1998) und Schwegmann (1999). Kernpunkte seines Entwurfs eines Vorgehensmodells sind die *Problemdefinition*, die *Analyse und Zerlegung der Problemdomäne*, die *iterative und inkrementelle Konstruktion des Referenzmodells* und die *Evaluation*. Der Hauptgrund, weshalb in dieser Thesis auf der Basis der Vorgehensweise zur Erstellung von Referenzmodellen von Schwegmann (1999) und nicht von Schlagheck (2000) vorgegangen wird, ist Schwegmanns stärkerer Bezug auf Klassen von objektorientierten Referenz-Klassenmodellen als Grundlage für die Konstruktion eines Referenzmodells. Dieser ist bei Schlagheck in diesem Maße nicht zu sehen.

3.4.3 Objektorientierte Referenzmodellierung

In Abschnitt 3.3.1 wurden unter anderem die Wiederverwendbarkeit, die Anpassbarkeit und die hohe Verständlichkeit als Vorteile des objektorientierten Paradigmas genannt. Da diese Vorteile mit den Anforderungen an Referenzmodelle (vgl. Tabelle 4) zum größten Teil korrespondieren, scheint eine Verknüpfung des objektorientierten Paradigmas mit der Referenzmodellierung zumindest auf den ersten Blick sinnvoll und vorteilhaft. Dagegen spricht, dass in der Objektorientierung lediglich die Mikrostruktur und das Mikroverhalten in Klassen gekapselt werden kann und dass die Klassen folglich zu feingranular sind, um eine hinreichende Komplexitätsreduzierung zu erzielen (vgl. Schwegmann 1999: S.112). Da Klassen bzw. Klassendiagramme als „Kernbestandteil objektorientierter (Referenz-)Modelle“ (vom Brocke 2002: S.122) gelten, wird im Folgenden die Identifizierung und Abbildung von Klassen objektorientierter

Referenz-Klassenmodelle genauer definiert, um eine hinreichende Komplexitätsreduzierung zu gewährleisten.

Grundsätzlich kann zur *Identifizierung von Klassen* in Referenz-Klassenmodellen auf die Ausführungen zu Klassen und Objekten in der UML in Abschnitt 3.3.2 zurückgegriffen werden. Dieser grundlegende Ansatz basiert im einfachsten Fall darauf, dass Klassen als Schablonen zum Erzeugen von Objekten sich während der Modellierung bzw. Programmierung von selbst ergeben. Als mögliche Hilfsmittel für den Prozess der Klassenfindung nennt Schwegmann (1999: S.169) unter anderem die Auswertung vorhandener Dokumentationen, textuelle Analysen von Anforderungsspezifikationen, Pflichtenheften etc. und das Ableiten von Input aus Prozessmodellen, die wiederum aus der Auswertung vorhandener Dokumentationen und durch systematische Befragungen von Fachanwendern entstehen. Anhand dieser Hilfsmittel können Klassen in der Referenzmodellierung von Grund auf neu entwickelt werden. Diese Art der Identifizierung von Klassen beschreibt allerdings nicht den optimalen Weg in der objektorientierten Referenzmodellierung.

> Der wirtschaftlichste und erfolgversprechendste Weg der Klassen- bzw. Prozeßidentifikation für die Referenzmodellerstellung ist die Analyse bereits vorhandener Informationssystemmodelle. (Schwegmann 1999: S.169)

Auf diese Art können Klassen und ggf. auch Attribute, Methoden und Beziehungen zwischen Klassen aus bereits existierenden Modellen extrahiert und in das Referenzmodell übertragen werden.

Bei der *Abbildung von Klassen* objektorientierter Referenz-Klassenmodelle gilt es Gestaltungsempfehlungen zu berücksichtigen, die über die allgemeinen Regeln zur Abbildung von Klassen (vgl. Abschnitt 3.3) hinausgehen. Das umfangreiche Repertoire an Modellierungswerkzeugen der UML zur Beschreibung von Klassen muss daher an die Anforderung der Minimalität der Beschreibungssprache in der Referenzmodellierung angepasst werden. Solche Vorschläge zur allgemeinen Anpassung von

Klassendiagrammen für die Referenzmodellierung werden in mehreren Abhandlungen beschrieben (vgl. vom Brocke 2002 oder Schlagheck 2000). Da sich diese dabei stets auf die ursprünglich von Schwegmann (1999) entwickelten Gestaltungsempfehlungen zur Anpassung des Klassendiagramms beziehen, werden diese nachfolgend erläutert:

Zunächst soll auf gerichtete Assoziationen, Assoziationsklassen und mehrgliedrige Assoziationen verzichtet werden, da aus der Verwendung dieser Konstrukte kein spezifischer Nutzen für die objektorientierte Referenzmodellierung resultiert. Die Dokumentation von Klassenmerkmalen (Klassenattributen und –methoden) sollte auch möglichst vermieden werden, um die Modelleser nicht zu überfordern. Weiterhin wird die Darstellung von abgeleiteten Attributen und Methoden oder die Kennzeichnung der Sichtbarkeit von den Merkmalen nicht als sinnvoll erachtet, da diese einen primär implementierungstechnischen Charakter haben. Darüber hinaus wird zur Komplexitätsbeherrschung generell darauf verzichtet, Attribute mit weiteren Charakteristika zu spezifizieren.

Unter Berücksichtigung der Gestaltungsempfehlungen ist bei der Abbildung eines Referenz-Klassendiagramms zu entscheiden, inwieweit identifizierte Klassen, Attribute und Methoden tatsächlich aufgenommen werden sollen. Dabei ist „die Eliminierung von Klassen genauso wichtig […] wie die Identifizierung von Klassen“ (Schwegmann 1999: S.170). Schwegmann empfiehlt in diesem Kontext, eliminierte Klassen ggf. als Attribut einer anderen Klasse zu modellieren.

4 Fallbeispiel aus der Praxis

4.1 Unternehmen KOSTAL

Das folgende Fallbeispiel wird in Zusammenarbeit mit der Leopold Kostal GmbH & Co. KG (KOSTAL) entwickelt. Das 1912 in Lüdenscheid gegründete Familienunternehmen beschäftigt heutzutage 17.000 Mitarbeiter an 46 Standorten in 21 Ländern. Der Großteil davon arbeitet mittlerweile an den ausländischen Standorten. Die Quote der KOSTAL-Mitarbeiter im Ausland liegt insgesamt bei 79%. Das Unternehmen gliedert sich in die vier Sparten Automobil Elektrik, Industrie Elektrik, Kontakt Systeme und SOMA Prüftechnik. Insgesamt konnte das Unternehmen 2017 einen Umsatz von 2,564 Milliarden Euro generieren. Dieser Wert bestätigt eine seit Jahren kontinuierliche Umsatzsteigerung, die lediglich im Jahr 2009 bedingt durch die Weltwirtschaftskrise einen Rückgang vorzeigt.

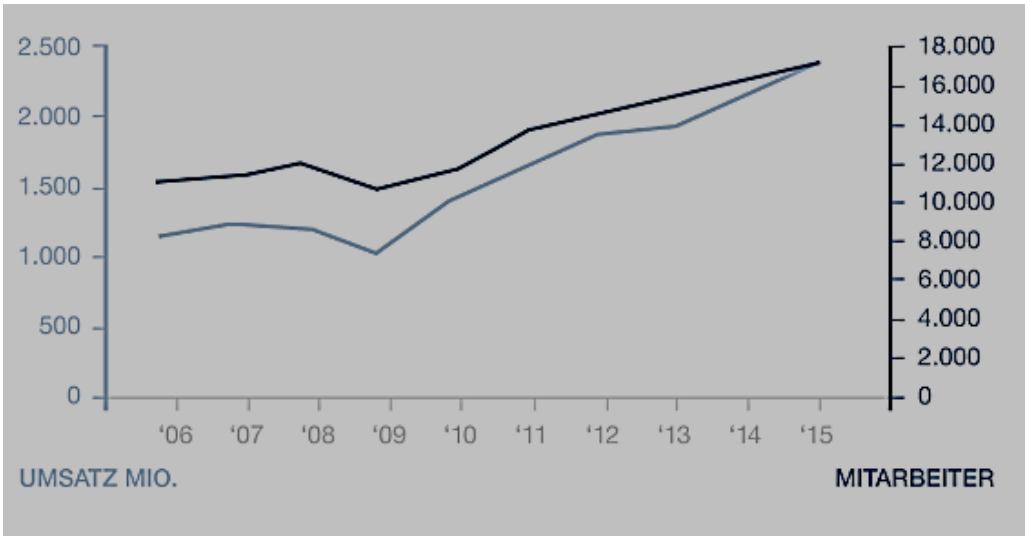


Abbildung 16: Umsatz (in €) und Mitarbeiterzahl KOSTAL von 2006 bis 2015 (KOSTAL, 2018)

Die Sparte der Automobil Elektrik beschreibt den größten Geschäftsbereich der KOSTAL-Gruppe. Im Genre der Automobilzulieferer zählt die KOSTAL Automobil Elektrik mit einem Warenumsatz von 1,95 Milliarden

Euro (2016) zu den Top 100 weltweit (KOSTAL, 2018). Dabei konzentriert sich die KOSTAL Automobil Elektrik im Wesentlichen auf die drei Geschäftsbereiche Mechatronik Module, Bedienelemente/ Schalter und elektronische Steuergeräte (vgl. Abbildung 17).

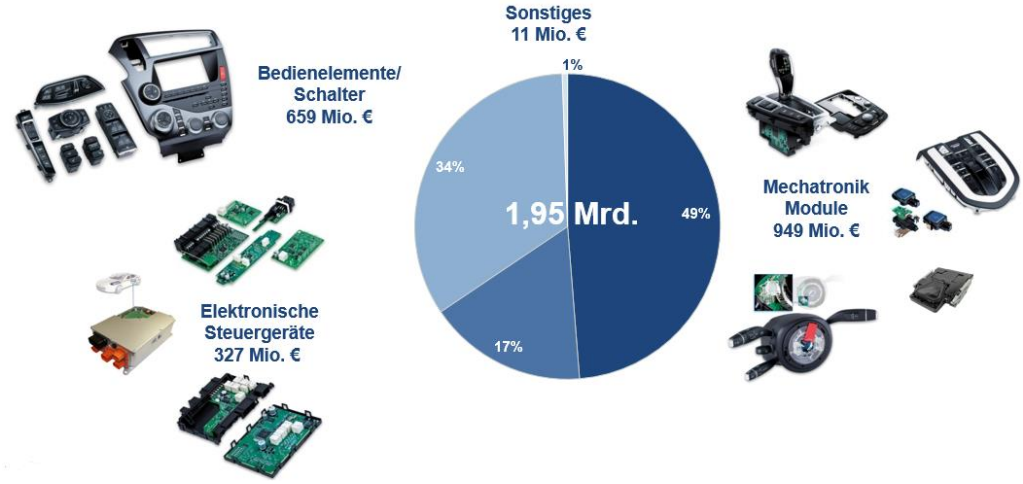


Abbildung 17: KOSTAL Automobil Elektrik Geschäftsbereiche (KOSTAL Intranet)

4.2 Situation Projektkoordination Einzelteile

Das eigentliche Datenmodell wird in dem Abteilungsbereich „Projektkoordination Einzelteile“ entworfen. Die Projektkoordination Einzelteile (nachfolgend zur Einfachheit nur noch Projektkoordination genannt) ist im Headquarter des Unternehmens in Lüdenscheid im Geschäftsbereich der Automobil Elektrik, in der Abteilung „Production Engineering Einzelteile/ Werkzeuge“ ansässig. Die grundsätzliche Aufgabe der Projektkoordination ist die vollständige, abteilungsübergreifende und globale Koordinierung von thermoplastischen Einzelteilen, von der Entwicklung bis zur finalen Übergabe an die Serienproduktionsstandorte. Die thermoplastischen Einzelteile werden durch Spritzgießverfahren hergestellt. Die dafür zu entwickelnden Spritzgießwerkzeuge (nachfolgend zur Einfachheit nur noch Werkzeuge genannt) werden ebenfalls in der Projektkoordination koordiniert. Die Spritzgussmaschinen, auf denen die Werkzeuge

zum Einsatz kommen, werden in einem anderen Bereich des Unternehmens koordiniert. In Bezug auf das Datenmodell werden sie im Rahmen der zu koordinierenden Werkzeuge aufgenommen, aber nicht mit weiteren Koordinationsschritten in Verbindung gesetzt.

Aufgrund der Trends in der Automobilindustrie (vgl. Kapitel 1) finden die zu koordinierenden Projekte mittlerweile größtenteils in einem globalen Rahmen statt. Dazu kommt die zunehmende Verzahnung mit anderen, wachsenden Bereichen des Unternehmens wie der Qualität und der Entwicklung. Durch die Verschiebung des Wertschöpfungsumfangs vom OEM zum Zulieferer (vgl. Kapitel 1) geht auch häufig die dazugehörige Entwicklung an die Zulieferer (vgl. Bratzel et al. 2015: S.61). Bei KOSTAL werden daher mittlerweile acht Prozent des gesamten Umsatzes des Unternehmens in die Forschung und Entwicklung investiert (KOSTAL, 2018). Für das Datenmanagement in der Projektkoordination bedeutet das heutzutage, eine größere und vielfältigere Menge an Daten aus mehreren Abteilungen des Unternehmens und an diversen Standorten weltweit aufzunehmen und zu verarbeiten. Die Daten werden in den verschiedenen Abteilungen und an den verschiedenen Standorten in unterschiedlichen Systemen gehalten. Die in der Projektkoordination arbeitenden Projektkoordinatoren haben zwar Zugriff auf die einzelnen Systeme, müssen aber aus *Zeit- und Übersichtlichkeitsgründen* eigene Listen mit Microsoft Excel anlegen. Da in den meisten Projekten viele Einzelteile simultan koordiniert werden müssen, werden die Excel-Listen mehrmals in der Woche über vorprogrammierte Makros mit den neusten Daten aus den anderen Systemen aktualisiert. Ein Makro lässt sich als programmierte Routine verstehen, die Prozesse automatisiert und wiederkehrende Aufgaben schneller erledigt (vgl. Feiks 2016: S.45). Dieser dennoch zeitintensive Vorgang findet in der Regel morgens statt und muss manuell eingeleitet werden. Allerdings treten bei voranschreitender Projektzeit und einer steigenden Anzahl und Varietät von Daten fortlaufend Fehler auf, sodass die Listen teilweise falsch oder gar nicht aktualisiert werden können. Die darauffolgenden Fehlerkorrekturen sind sehr

müßig und kosten viel Zeit. Erschwerend kommt noch hinzu, dass während der Expansion und Globalisierung des Unternehmens keine globalen Datenstandards zwischen den Abteilungen und Standorten definiert wurden.

4.3 Datenmodell

Im folgenden Abschnitt wird ein intelligenterer Ansatz zur Aufnahme und Verwaltung von Daten in der Projektkoordination vorgestellt. Als Grundlage hierfür wird ein objektorientiertes Datenmodell, genauer ein Klassendiagramm, mit Hilfe der Modellierungssprache UML entworfen.

Als Modellierungswerkzeug wird „Enterprise Architect (EA)“ der Firma Sparx Systems genutzt. EA ist ein umfangreiches UML Analyse und Design-Werkzeug und unterstützt das Modellieren aller in der UML 2.5 spezifizierten Modelle. Aufgrund der größtenteils globalen Ausrichtung der Projektkoordination wird das Datenmodell in englischer Sprache modelliert. Abgesehen von der Root-Klasse, die keine Attribute hat, wird jeder Klasse im Datenmodell über ein Attribut oder mehrere Attribute ein eindeutiger Identifier zugeteilt. Diese Attribute werden durch den Eigenschaftswert {id} gekennzeichnet. Durch den eindeutigen Identifier soll sichergestellt werden, dass im Datenmodell keine redundanten Objekte angelegt werden können. Attribute ohne angegebene Multiplizität haben gemäß Tabelle 2 die Multiplizität [1..1] als Vorgabewert. Aufgrund des Umfangs dieser Thesis liegt der Fokus des Datenmodells darauf, die grundlegende Struktur (und noch nicht das Verhalten) für die Aufnahme und Verwaltung aller anfallenden Daten im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen zu modellieren, dementsprechend wird in den einzelnen Klassen auf die Modellierung von Operationen verzichtet. Da das Datenmodell langfristig auch als zentrale Datenbasis zum Datenaustausch im Unternehmen gedacht ist, werden die Sichtbarkeiten aller Attribute aus Gründen des uneingeschränkten

Zugriffs als public (+) modelliert. In den Abschnitten 4.3.1 bis 4.3.12 werden die einzelnen Klassen samt ihrer Attribute und Assoziationen vorgestellt. Das gesamte Datenmodell mit allen vorher definierten Klassen und Assoziationen ist aus Gründen der Übersichtlichkeit im Anhang (Abbildung 31) zu finden und der ausgedruckten Thesis in DIN A3 Format beigelegt.

4.3.1 Klasse Root

Die Klasse *Root* beschreibt den Startpunkt einer baumartigen Struktur des Datenmodells, vergleichbar wie Ordner und Dateien auf einer Festplatte. Die Root-Klasse schließt jede weitere Klasse des Datenmodells in sich ein und ist daher direkt oder indirekt mit jeder anderen Klasse über eine Komposition verbunden. Diese Art der Datenstrukturierung ermöglicht gleichzeitig das einfache, spätere Überführen der Daten in die Extensible Markup Language (XML), welche sich als bewährte Standardstrukturierungssprache zum Informationsaustausch von Daten etabliert hat (vgl. Mertens et al. 2017: S.23). Dadurch kann die spätere Verwaltung und Bereitstellung der Daten erheblich vereinfacht werden, da die XML es ermöglicht Daten bezüglich ihres Inhalts und ihrer Darstellungsform derart zu beschreiben, dass sie zwischen einer Vielzahl von Anwendungen in verschiedensten Hardware- und Softwareumgebungen, insbesondere auch über das Internet, ausgetauscht und weiterverarbeitet werden können (vgl. Mertens et al. 2017: S.23). Die Root-Klasse besitzt keine Attribute. Über Kompositionen verbunden, ist die Root-Klasse jeweils die Aggregatklasse der Klassen *Customer*, *Resin* und *Project*. Die Multiplizität ist dabei jeweils optional beliebig.

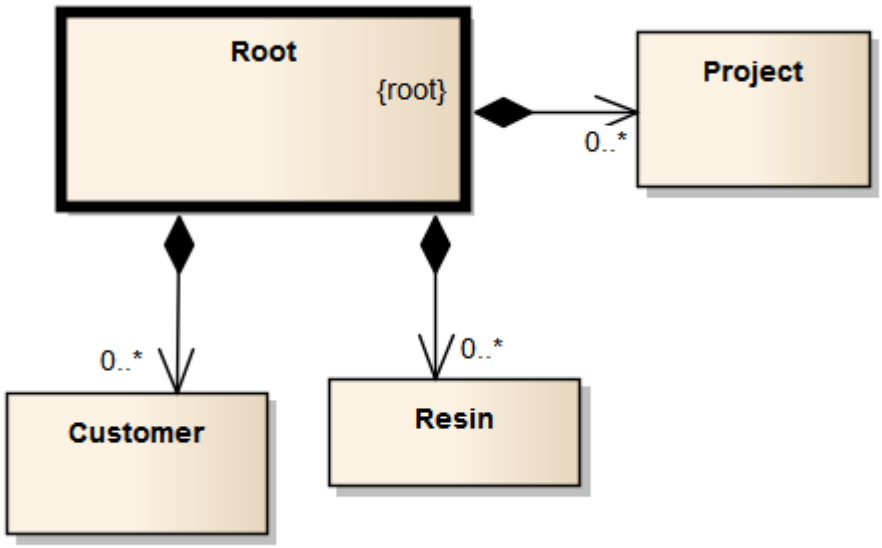


Abbildung 18: Klasse Root mit Teilklassen

4.3.2 Klasse Customer

Die Klasse *Customer* beschreibt die Kunden des Unternehmens, repräsentiert durch die OEMs. Der Identifier der Klasse ist das Attribut *ShortName* des Datentypes String. Dazu wird ebenfalls das Attribut LongName des Datentypes String eingeführt, dass den kompletten Namen des Kunden beinhaltet. Als Beispiel würde ein Objekt der Customer-Klasse den ShortName „DAI“ haben und den dazu ausgeschriebenen LongName „Daimler Benz AG“. Das dritte Attribut in dieser Klasse heißt *Picture*. Dieses optionale Attribut hat eine besondere Bedeutung und wird in dieser Form auch noch in anderen Klassen des Modells bentutzt. Dieses Attribut des Datentypes String dient zur Aufnahme einer vordefinierten Uniform Resource Locator (URL), welche als Standard für die Adressierung einer Webseite im World Wide Web im firmeneigenen Intranet an jedem Standort weltweit auf eine Portable Network Graphic (PNG) verweist. Das Intranet kann als „Synonym für ein beliebiges internes, meist unternehmensweites Daten- und Kommunikationsnetzwerk“ (Kaya 2007: S.40) verstanden werden, welches nach den

technischen Standards des Internets aufgebaut ist. Dadurch ist es bei der späteren Bereitstellung der Daten möglich, durch das Anzeigen von PNGs ein benutzerfreundlicheres Layout zu entwerfen. Das oben genannte Beispielobjekt würde in diesem Fall im Attribut Picture die URL aufnehmen, die zum Firmenlogo der Daimler Benz AG im Intranet führen würde. Die besondere Relevanz des Picture-Attributes wird in der später folgenden Part-Klasse (Abschnitt 4.3.6) deutlich. Die Customer-Klasse ist eine Teilklasse der Root-Klasse und zudem über eine bidirektionale Assoziation mit der Project-Klasse verbunden. Dabei gilt, dass ein Kunde mehrere Projekte (Objekte der Klasse *Project*) haben kann. Ein Projekt kann allerdings immer nur maximal einem Kunden zugeordnet sein. Demensptrechend sind die Multiplizitäten *0..** und *0..1*.

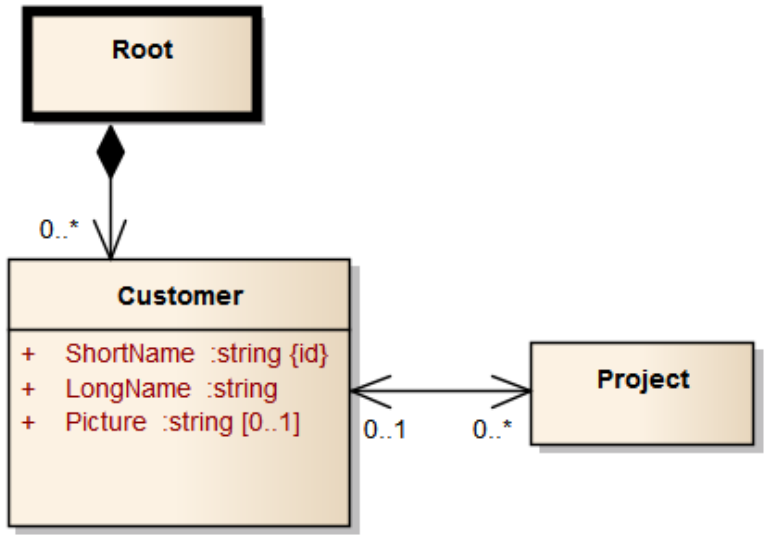


Abbildung 19: Klasse Customer mit Assoziationen

4.3.3 Klasse Resin

In der Klasse *Resin* werden alle Granulate aufgenommen, die zur Produktion der thermoplastischen Einzelteile (Objekte der Klasse *Part*) benutzt werden. Der Identifier in dieser Klasse ist das Attribut *TradeName*

des Datentyps String. Da die Granulate ausschließlich von externen Firmen eingekauft werden, ist der Identifier dieser Klasse die Bezeichnung des Granulats der externen Firma. Zusätzlich wird noch das Attribut *ShortName* des Datentyps String definiert. Eine Kurzbezeichnung des Granulats soll immer vorhanden sein. Da diese aber unter Umständen je nach Standort unterschiedlich ist, kann die Kurzbezeichnung in dieser Klasse, im Gegensatz zur Customer-Klasse, nicht als Identifier genutzt werden. Die Resin-Klasse ist eine Teilklasse der Root-Klasse und besitzt bidirektionale Assoziationen zur Location-Klasse und zur Part-Klasse. Ein Granulat kann an mehreren Produktionsstandorten (Objekte der Klasse *Location*) zum Einsatz kommen und ein Standort von einem externen Zulieferer (auch Objekte der Klasse *Location* siehe Abschnitt 4.3.8) kann mehrere Granulate zur Verfügung stellen. Des Weiteren kann ein Granulat für unterschiedliche Einzelteile (Objekte der Klasse *Part*) genutzt werden. Ein Einzelteil kann zudem in globalen Projekten mehreren Granulaten zugeordnet werden. Ein Einzelteil kann in einem globalen Projekt an mehreren Produktionsstandorten gleichzeitig hergestellt werden. Das zur vielfachen Herstellung dieses Einzelteil verwendete Granulat kann dann beispielsweise an den Produktionsstandorten in China und in Bulgarien einen anderen Handelsnamen tragen und von einem anderen Unternehmen zur Verfügung gestellt werden, solange die Inhaltsstoffe nachweislich gleich sind.

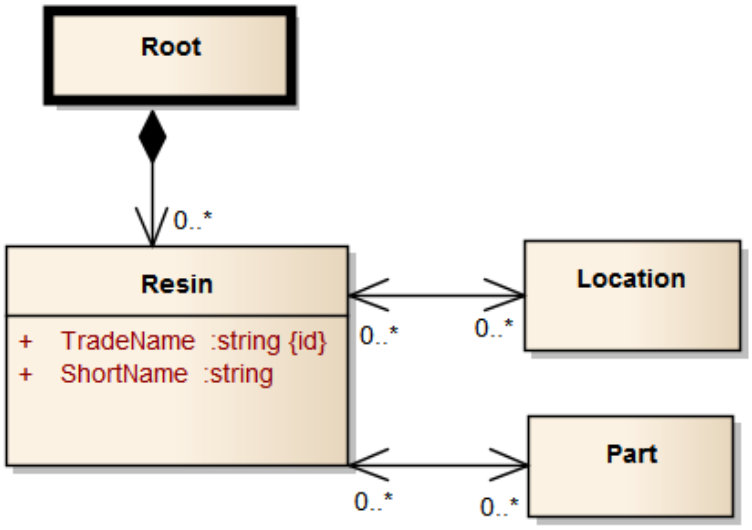


Abbildung 20: Klasse Resin mit Assoziationen

4.3.4 Klasse Project

Die Klasse *Project* ist die zentrale, wenn auch nicht die komplexeste Klasse des Modells. Die einzelnen Objekte der Project-Klasse, ergo die Projekte an sich, bilden das Grundgerüst der gleichnamigen Projektkoordination. Der Identifier der Project-Klasse ist das Attribut *Number* des Datentyps String. Die Bezeichnung *Number* hat sich allgemein durchgesetzt, obgleich Projekte vielmehr durch eine Kombination von Buchstaben und Nummern beschrieben werden (beispielsweise *Project*: „P06020“). Der zunächst zu vermutende Datentyp Integer wäre folglich nicht zulässig für dieses Attribut. Des Weiteren wird der Name eines Projektes im Attribut *Name* des Datentyps String aufgenommen. Das Attribut *ProKostalUrl* ist gemäß seinem Namen zur Aufnahme einer vordefinierten URL gedacht, die auf das Projekt im firmeneigenen ProKOSTAL System verweist. Im abteilungsübergreifenden ProKOSTAL System werden die global verteilten Aktivitäten im Produktentstehungsprozess strukturiert und synchronisiert und notwendige Eskalationen bei Zielabweichungen definiert. Es gewährt dem Benutzer eine Gesamtübersicht über das jeweilige Projekt und wird daher als optional auszufüllendes Attribut mit

aufgenommen. Das optionale Attribut *Picture* ist bereits aus der Customer-Klasse bekannt. In der Project-Klasse verweist es auf eine PNG des gesamten Projektes. Ein Projekt kann beispielsweise aus einem kompletten Lenksäulenmodul für einen OEM bestehen. Die zu koordinierenden Teile in der Projektkoordination der Produktion würden sich zwar auf die im Modul verbauten thermoplastischen Einzelteile beschränken, im Attribut *Picture* der Project-Klasse würde als PNG aber das komplette Lenksäulenmodul angezeigt werden. Auf diese Weise geht bei großen Projekten mit vielen unterschiedlichen Einzelteilen der Gesamtüberblick nicht verloren.

Die Project-Klasse ist die Aggregatklasse für die Teilklassen *Geometry*, *Person*, *ToDo*, *Mould*, *Location* und *Machine*. Die Multiplizität der Teilklassen ist bei allen optional beliebig. Zusätzlich gibt es noch eine unidirektionale Assoziation zur Part-Klasse, mit einer optional beliebigen Multiplizität und der Rolle *CurrentPart*. Dies hat den Hintergrund, dass aus logischer Betrachtung nicht die Einzelteile (Objekte der Klasse *Part*) als direkte Komposition unter dem Projekt stehen, sondern die davor entworfenen Geometrien (Objekte der Klasse *Geometry*) der Einzelteile, sprich die virtuellen Einzelteile. Um aber in der späteren Bereitstellung der Daten die Einzelteile ohne den für viele Benutzer uninteressanten Zwischenschritt der Geometrien, direkt unter dem Projekt anzeigen zu lassen, wird diese zusätzliche Assoziation eingeführt. Die Rolle *CurrentPart* bedeutet zudem, dass von jedem durch diese Assoziation verbundenem Einzelteil nur die Matrikelnummer (Attribut *MatNo* der Klasse *Part*) mit dem höchsten Index (Attribut *MatIndex* der Klasse *Part*) angezeigt werden soll. Dadurch kann in der späteren Layout-Gestaltung durch einfache Programmierung ein guter Überblick über die aktuellsten Stände der Einzelteile geschaffen werden. Des Weiteren wird noch eine zusätzliche unidirektionale Assoziation zur ToDo-Klasse mit einer optional beliebigen Multiplizität und der Rolle *OpenToDo* eingeführt. Auf dem Hintergrund, dass Objekte der ToDo-Klasse einen abgeschlossenen Status haben können (Attribut *Finished* der Klasse *ToDo*, näheres dazu in Abschnitt

4.3.12), soll diese Assoziation einen einfachen Weg beschreiben, um unter dem aktuellen Projekt nur die noch zu bearbeitenden Objekte der ToDo-Klasse (*OpenToDo*) anzeigen zu lassen. Die bideriktionale Assoziation zur Customer-Klasse wurde bereits in Abschnitt 4.3.2 beschrieben.

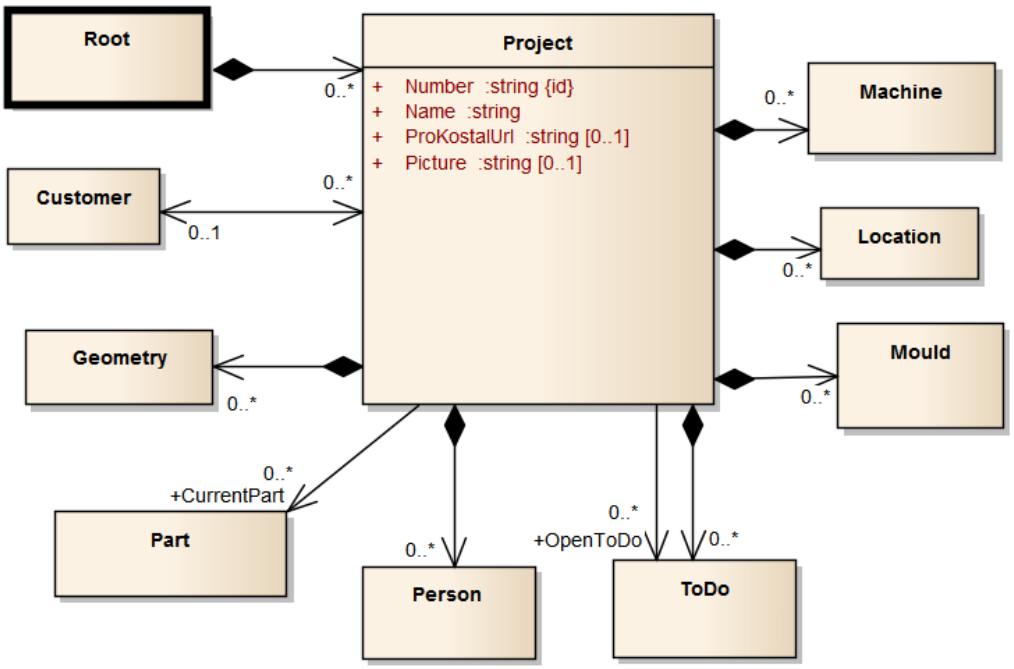


Abbildung 21: Klasse Project mit Assoziationen

4.3.5 Klasse Geometry

In der Klasse *Geometry* werden die virtuellen Einzelteile eines Projektes aufgenommen. In den Identifier *CadPart3DDocNumber* und *CadPart3DDocIndex* des Datentyps String werden Nummer und Index des per Computer Aided Design (CAD) in 3D entworfenen Einzelteils aufgenommen. Zusätzlich wird auch der Name des virtuellen Einzelteils im Attribut *Name* aufgenommen. In Relation zur Part-Klasse ist der Name eines virtuellen Einzelteils zu seinem realen Pendant identisch. Für Nummer und Index gilt das nicht. Auch in der Geometry-Klasse wird wieder optional die Möglichkeit geboten, eine URL zu einer PNG des virtuellen

Einzelteils aufzunehmen. Die Geometry-Klasse hat nur einer Assoziation in Form einer Komposition zur Klasse *Part*. Dabei kann es möglich sein, dass einem virtuellen Einzelteil mehrere reale Einzelteile (Objekte der Klasse *Part*) zugeordnet werden. Mehrere geometrisch gleiche Einzelteile mit der gleichen *CadPart3DDocNumber* und dem gleichen *CadPart3DDocIndex* können aufgrund ihrer Farbe (hervorgerufen durch das im Einzelteil genutzte Granulat als Objekt der Resin-Klasse) unterschiedliche Identifier (Attribut *MatNo* und *MatIndex* in Klasse *Part*) haben.

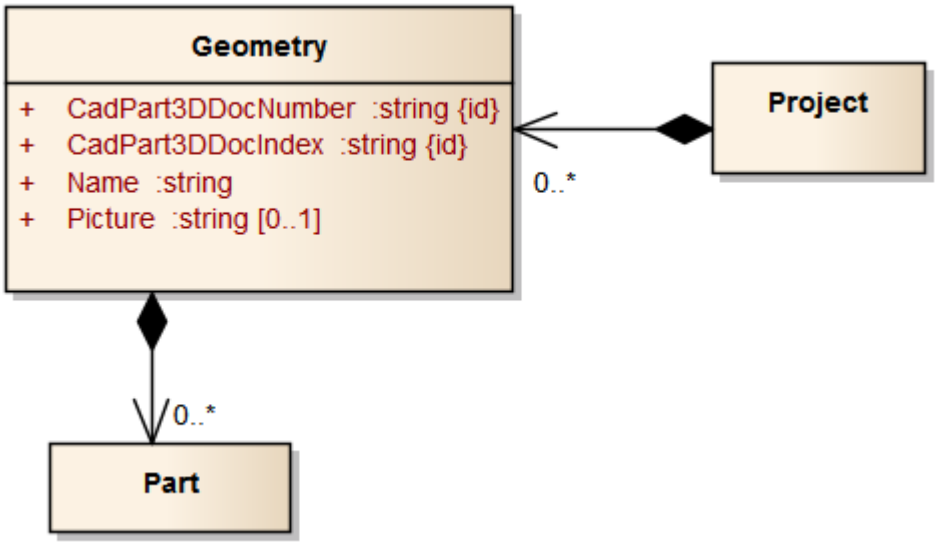


Abbildung 22: Klasse Geometry mit Assoziationen

4.3.6 Klasse Part

Die eigentlichen thermoplastischen Einzelteile werden in der Klasse *Part* aufgenommen. Diese werden identifiziert durch ihre Matrikelnummer und ihren Index. Die dazugehörigen Attribute heißen *MatNo* und *MatIndex* des Datentyps Integer. Anders als in der Geometry-Klasse dürfen die Nummern und Indizes der realen Einzelteile nur aus einer Kombination von Zahlen bestehen. Im Attribut *Name* wird zusätzlich der Name des Einzelteils angegeben. Das optionale Attribut *Picture* nimmt in diesem

Fall die URL zu einer PNG des realen Einzelteils auf. In der Part-Klasse ist das Attribut *Picture* von besonderer Wichtigkeit. Die einzelnen Matrikelnummern unterscheiden sich oftmals in nur einer Ziffer voneinander. Durch die Verknüpfung der Matrikelnummer mit der PNG des Einzelteils in der späteren Layout-Gestaltung können Verwechslungen von vorneherein vermieden und die korrekte Nutzung der Daten gewährleistet werden. In dem Attribut *MakeOrBuy* wird festgelegt, ob das jeweilige Einzelteil eine Eigenanfertigung oder ein Kaufteil ist. Dieses Datenmodell ist auf den komplexeren Vorgang der Eigenanfertigung von Einzelteilen ausgelegt. Nichtsdestotrotz wird die Produktion von manchen Einzelteilen auch teilweise an andere Unternehmen ausgelagert. Für solche *BuyParts* würde ein stark reduziertes Datenmodell genügen. Das Attribut *MakeOrBuy* hat also den Aufzählungstypen *PartTypeDef*, welches als enumeration modelliert, ausschließlich die diskreten Werte *MakePart* oder *BuyPart* annehmen kann.

Zusätzlich zu den bereits beschriebenen Assoziationen zu den Klassen *Resin* (siehe Abschnitt 4.3.3) und *Project* (siehe Abschnitt 4.3.4), besitzt die Part-Klasse noch bidirektionale Assoziationen zu den Klassen *Person*, *ToDo*, *Mould* und *Location*. Zur Assoziation mit der Person-Klasse gilt, dass einem Einzelteil immer nur genau ein Designer zugeordnet werden muss (*Designer 0..1*), eine Person der Klasse *Person* aber mehreren Einzelteilen zugeordnet werden kann. Die Beziehung zur ToDo-Klasse ist in beide Richtungen optional beliebig. Ein Einzelteil kann mehreren ToDos zugeordnet werden und ein ToDo kann sich gleichzeitig auf mehre Einzelteile beziehen. Die Beziehung zur Mould-Klasse ist ebenfalls in beide Richtung optional beliebig. Ein Einzelteil kann aus mehreren Werkzeugen (Objekte der Klasse *Mould*) fallen, da in globalen Projekten durchaus Werkzeuge mit unterschiedlichen Identifiern (Attribut *EquipNumber* der Klasse *Mould*) an verschiedenen Produktionsstandorten des Unternehmens das gleiche Einzelteil produzieren. Andersherum kann ein Werkzeug auch zur Produktion unterschiedlicher Einzelteile genutzt werden, da nur die Geometrie (Objekt der Klasse Geometry) der Einzelteile gleich sein muss (siehe

Beziehung Geometry-Klasse und Part-Klasse in Abschnitt 4.3.5). Ein Einzelteil kann allerdings immer nur einem Standort (Objekt der Klasse *Location*) gleichzeitig zugeordnet sein. Die Beziehung zur Location-Klasse ist trotzdem mit der Multiplizität *0..1* anstatt *1..1* definiert, da Einzelteile zwischen den internationalen Standorten verlagert werden müssen und in diesem Zeitraum keinem Standort eindeutig zugehören. Einem Standort können dagegen optional beliebig viele Einzelteile zugeordnet werden.

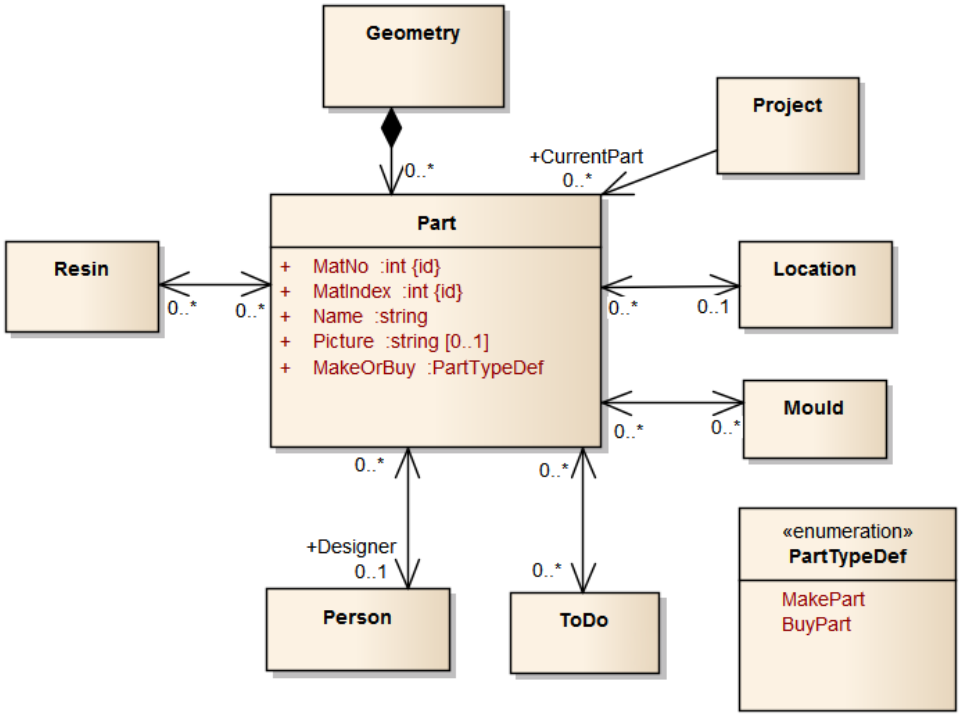


Abbildung 23: Klasse Part mit Assoziationen und enumeration

4.3.7 Klasse Person

Die Klasse *Person* dient zur Aufnahme aller am Projekt beteiligten Personen. Da KOSTAL ein Unternehmen mit insgesamt 17000 Mitarbeitern ist (siehe Abschnitt 4.1) reichen Vor- und Nachname nicht als Identifier dieser Klasse aus. Allein im Adressbuch des chinesischen KOSTAL-Standorts ist der Nachname „Wang“ mehr als 160-mal vergeben. Daher

bekommt jeder neue KOSTAL Mitarbeiter einmalig eine eindeutige ID (z.B. rosia001) vom Personalwesen zugewiesen. Diese bestehenden IDs werden in der Klasse Person im Attribut *ID* des Datentyps String als Identifier aufgenommen. Zur Vollständigkeit werden weiterhin die selbsterklärenden Attribute *LastName*, *FirstName*, und *EMail* als Pflichtattribute und *Country* und *Department* als optionale Attribute aufgenommen. Neben der bereits beschriebenen Assoziation zur Part-Klasse (siehe Abschnitt 4.3.6) hat die Person-Klasse noch zwei weitere bidirektionale Assoziationen zur ToDo-Klasse. In der oberen Assoziation (vgl. Abbildung 24) werden einer Person optional beliebig viele ToDos (Objekte der Klasse *ToDo*) zugeordnet, für welche diese Person verantwortlich ist (*RespActivity*). Gleichzeitig kann aber jedem ToDo nur maximal eine verantwortliche Person (*Responsible*) zugeordnet werden. So kann sichergestellt werden, dass bei Problemen oder Unstimmigkeiten zu einem bestimmten ToDo im Zweifelsfall eine Person die Verantwortung übernimmt. Des Weiteren können noch andere Personen an einem ToDo beteiligt sein. Diese werden über die untere Assoziation (vgl. Abbildung 24) zugeordnet. Dabei gilt, dass eine Person an beliebig vielen ToDos teilhaben kann (*ActingActivity*) und einem ToDo beliebig viele Personen zugeordnet werden können (*ActingPerson*).

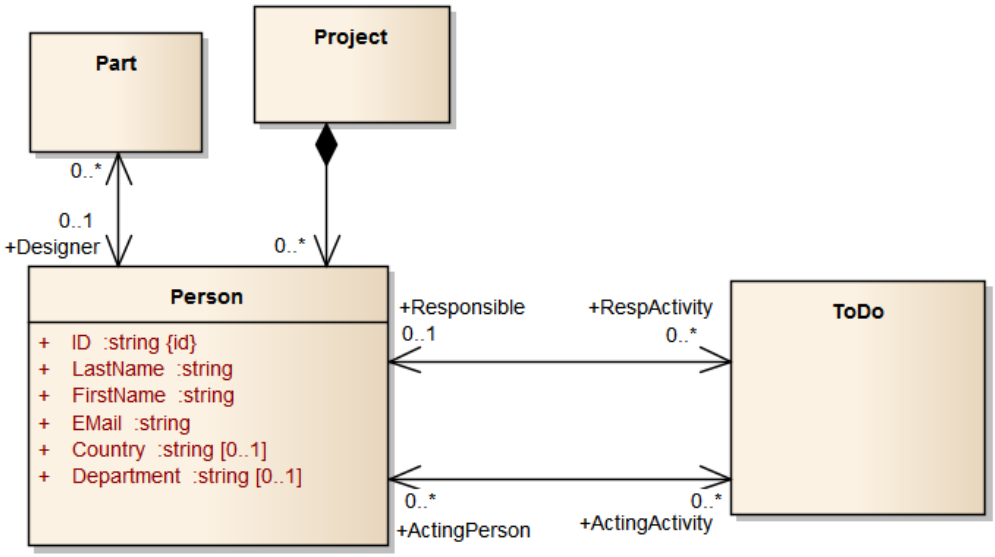


Abbildung 24: Klasse Person mit Assoziationen

4.3.8 Klasse Location

In der Klasse *Location* werden alle an einem Projekt teilhabenden Standorte aufgenommen. Mit Standorte sind sowohl die KOSTAL Standorte, als auch externe Produktionsstandorte, Standorte von Zulieferunternehmen und andere an einem Projekt teilhabende Standorte gemeint. Der Identifier eines Standorts ist dessen Name, aufgenommen in dem Attribut *Name* des Datentyps String. Zur Visualisierung wird auch in der Location-Klasse über das optionale Attribut *Picture* eine URL aufgenommen, die auf eine PNG des jeweiligen KOSTAL Standortes oder des Zulieferunternehmens verweist, sofern das Unternehmen als offizieller Zulieferer im Intranet samt PNG eingetragen ist. Neben den bereits bekannten Assoziationen zu den Klassen *Resin* (siehe Abschnitt 4.3.3) und *Part* (siehe Abschnitt 4.3.6), besitzt die Location-Klasse weitere bidirektionale Assoziationen zu den Klassen *Mould*, *ToDo* und *Machine*. Bei der Beziehung zur Mould-Klasse gilt das gleiche Prinzip wie schon bei der Beziehung zur Part-Klasse (Abschnitt 4.3.7). Einem Standort können beliebig viele Werkzeuge (Objekte der Klasse *Mould*) zugeordnet werden. Ein Werkzeug kann aber nur maximal einem Standort gleichzeitig zugeordnet sein. Die Assoziation zur ToDo-Klasse ist an beiden Enden optional beliebig. Einem Standort können mehrere ToDos zugeordnet sein und ein ToDo kann an mehreren Standorten gleichzeitig stattfinden. In Bezug zur Machine-Klasse können einem Standort optional beliebig viele Maschinen (Objekte der Klasse *Machine*) zugeordnet sein. Eine Maschine kann allerdings nur maximal einem Standort zugeordnet sein.

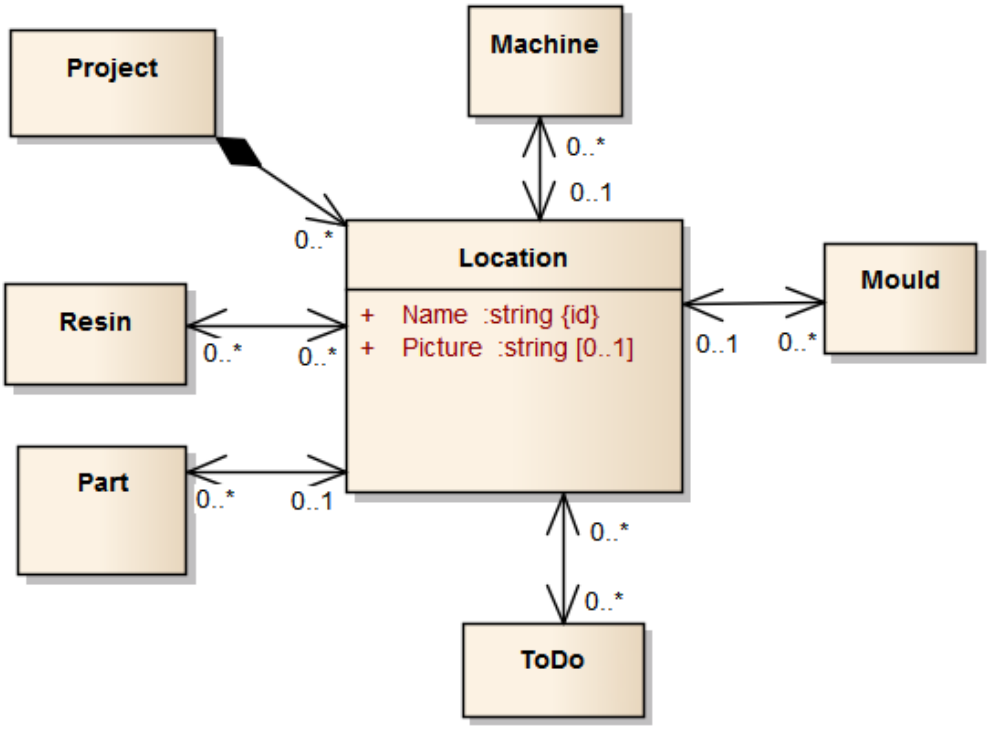


Abbildung 25: Klasse Location mit Assoziationen

4.3.9 Klasse Mould

Alle in einem Projekt anzufertigen Werkzeuge werden als Objekte der Klasse *Mould* (KOSTAL-interne Bezeichnung für Werkzeuge) aufgenommen. Jedes Werkzeug wird durch eine eindeutige interne Nummer identifiziert. Diese Nummer wird als Identifier im Attribut *EquipNumber* des Datentyps Integer aufgenommen. Darüber hinaus kann es vorkommen, dass ein extern hergestelltes Werkzeug, neben der ihm eindeutig zugeordneten internen Nummer, eine weitere externe Bezeichnung besitzt. Diese wird dann in dem optionalen Attribut *EqNoSupplier* des Datentyps String aufgenommen. Wichtig ist dabei, dass jedes extern hergestellte Werkzeug immer eine interne Werkzeugnummer besitzt, die es eindeutig identifiziert. Neben den bekannten Assoziationen zu den Klassen *Part* (siehe Abschnitt 4.3.6) und *Location* (siehe Abschnitt 4.3.8), besitzt die Mould-Klasse weitere bidirektionale Assoziationen zu den Klassen *Machine* und *ToDo*. Die Assoziation zur Machine-Klasse ist aufgrund der

guten Werkzeugstandardisierung bei KOSTAL an beiden Enden optional beliebig. Die meisten Werkzeuge harmonisieren auf mehreren Maschinen (Objekte der Klasse *Machine*) und eine Maschine kann dementsprechend meistens mehrere Werkzeuge benutzen. Besonders anspruchsvolle Werkzeuge bilden eine Ausnahme. Die Assoziation zur ToDo-Klasse ist ebenfalls an beiden Enden optional beliebig. Einem Werkzeug können mehrere ToDos (Objekte Der Klasse *ToDo*) zugeordnet werden und ein ToDo kann sich auf mehrere Werkzeuge beziehen.

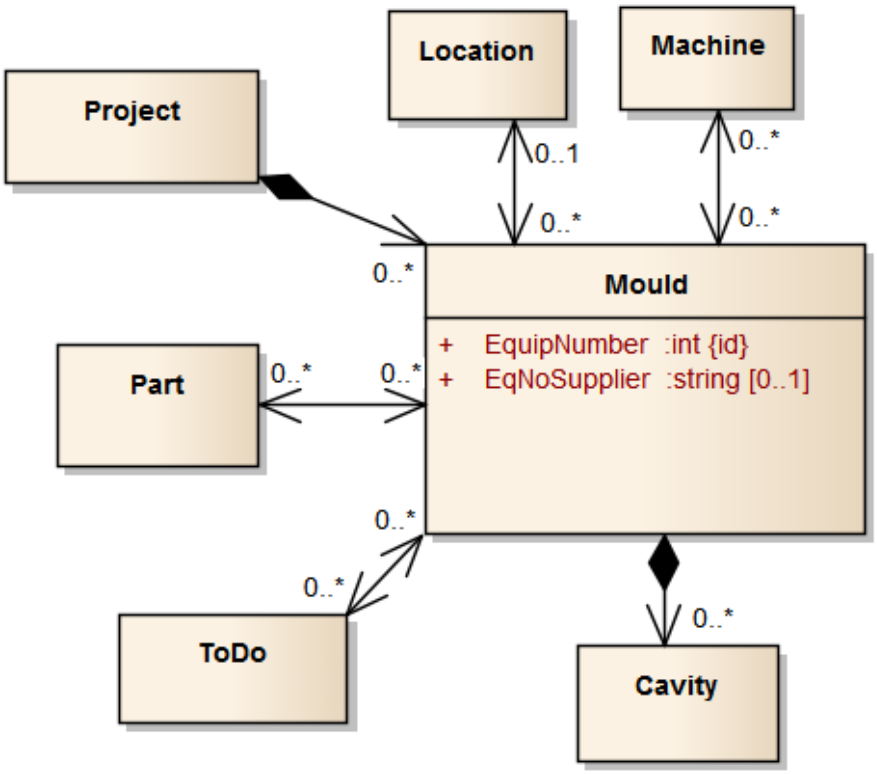


Abbildung 26: Klasse Mould mit Assoziationen

4.3.10 Klasse Cavity

Jedes Werkzeug (Objekt der Klasse *Mould*) hat eine bestimmte Anzahl an Kavitäten. Kavitäten beschreiben die Anzahl der Formnester in einem Spritzgießwerkzeug (vgl. Dangel 2015: S.68). In dieser Klasse *Cavity* wird jede einzelne Kavität eines Werkzeuges explizit als eigenes Objekt

der Klasse aufgenommen. Da die Klasse *Cavity* über eine Komposition mit der Mould-Klasse verbunden ist und somit jede Kavität einem Werkzeug eindeutig zugeordnet ist, reicht die simple einstellige Nummer der Kavität als Identifier aus. Dementsprechend wird diese Nummer im Attribut *Number* des Datentyps Integer als Identifier aufgenommen. Weiterhin wird das optionale Attribut *CorrectionStatus* des Datentyps String definiert. Sollte eine Kavität fehlerhaft sein, kann in diesem Attribut der aktuelle Korrekturstatus der jeweiligen Kavität aufgenommen werden.

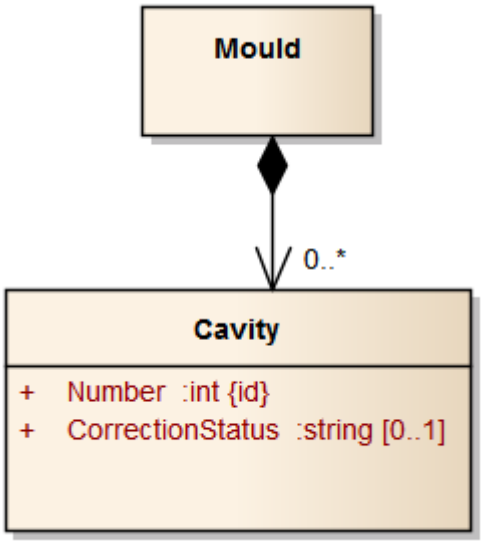


Abbildung 27: Klasse Cavity mit Aggregatklasse

4.3.11 Klasse Machine

In der Klasse *Machine* werden die eingesetzten Maschinen in einem Projekt als Objekte dieser Klasse aufgenommen. Dabei handelt es sich ausschließlich um jene Spritzgussmaschinen, auf denen die bereits bekannten Werkzeuge (Objekte der Klasse *Mould*) zum Einsatz kommen. Da die Maschinen nicht zum Bereich der Projektkoordination gehören (siehe Abschnitt 4.2), werden diese mit keinen weiteren ToDos (Objekte der Klasse *ToDo*) verknüpft. Sie werden lediglich in Beziehung mit dem jeweiligen Standort (Abschnitt 4.3.8), an dem sie sich befinden, und zu den jeweiligen Werkzeugen (Abschnitt 4.3.9), die auf ihnen benutzt werden können,

gesetzt. Die Attribute der Machine-Klasse sind dementsprechend an die essentiell benötigten Informationen, die von den Projektkoordinatoren in Bezug auf die Maschinen benötigt werden, angepasst. Nach diesem Prinzip werden die drei Attribute *Brand*, *Tonnage* und *ScrewBarrelSize* des Datentyps String definiert. Da alle drei Attribute essentiell für die erfolgreiche Zuordnung zu einem Werkzeug sind, werden sie auch alle als Identifier definiert. Das Attribut *Brand* nimmt die Marke der Maschine auf. Im Attribut *Tonnage* wird die maximale Schließkraft der Maschine in Tonnen aufgenommen und das Attribut *ScrewBarrelSize* nimmt den Durchmesser der Plastifizierschnecke der Maschine auf.

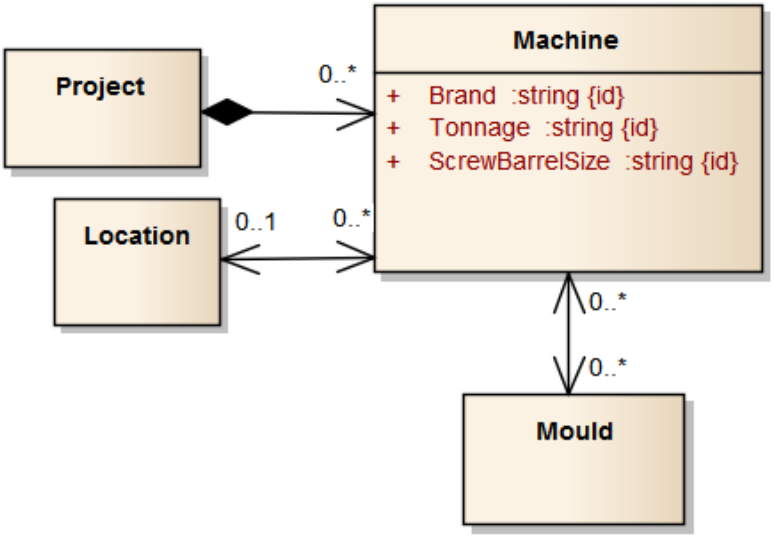


Abbildung 28: Klasse Machine mit Assoziationen

4.3.12 Klasse ToDo

Die Klasse ToDo ist die komplexeste Klasse dieses Datenmodells. Generell dient diese Klasse zur Aufnahme aller Arbeitsaufgaben oder auch ToDos genannt, die in einem Projekt entstehen können. Da die ToDo-Klasse eine Teilklasse der Project-Klasse ist, sind alle ToDos ein Teil des Projektes. ToDos beziehen sich aber nur in seltenen, übergeordneten Fällen ausschlielich auf das Projekt an sich. Der Großteil aller ToDos wird

nämlich entweder einem Einzelteil (Objekt der Klasse *Part*) oder einem Werkzeug (Objekt der Klasse *Mould*) zugeordnet. Des Weiteren können ToDos darüber hinaus einer oder mehreren Personen und/ oder einem oder mehreren Orten zugeordnet werden. Dementsprechend ist das eindeutige Identifizieren eines ToDos komplexer. Zunächst wird das Attribut *ToDoType* des Aufzählungstyps *ToDoTypeDef* als Identifier definiert. Die enumeration *ToDoTypeDef* beinhaltet alle Arbeitsaufgabentypen, die zu einem Einzelteil oder zu einem Werkzeug anfallen können. Für den Ausnahmefall, dass ein ToDo keinem der 20 Aufgabentypen zuzuordnen ist, wird das Attribut *ToDoType* per Default auf den ersten Wert *NotDefined* der enumeration gesetzt. Dies ermöglicht im Sonderfall auch ToDos einzulesen, die nicht einzelteil- oder werkzeugspezifisch sind. Das Attribut *ToDoType* als Aufzählunsgstyp und Identifier ist insofern wichtig, da es die einzelnen ToDos kategorisiert. Über den Sinn dahinter wird an späterer Stelle dieses Abschnittes genauer gesprochen.

Zur eindeutigen Identifizierung wird weiterhin das Attribut *ID* des Datentyps String definiert. Für den Großteil aller ToDos, die einem Einzelteil oder einem Werkzeug zuzuordnen sind, wird in diesem Attribut der eindeutige Identifizierungsschlüssel zu einem ToDo zum Einzelteil oder zum Werkzeug eingelesen. Die Datensysteme, aus denen die verschiedenen ToDos zu den Einzelteilen und Werkzeugen aufgenommen werden, legen automatisch zu jedem neuen ToDo eines ToDoTypes einen eindeutigen Identifizierungsschlüssel an. Dieser im Attribut *ID* aufgenommene Identifizierungsschlüssel, in Kombination mit dem jeweiligen ToDoType (notwendig, da der Idenitifizierungsschlüssel von einem Einzelteil bzw. einem Werkzeug aus unterschiedlichen Systemen kommt und gegenfalls gleich sein kann), identifiziert das ToDo im Datenmodell eindeutig. Im Sonderfall, dass ein ToDo keinem der ToDoTypes zuzuordnen ist und dadurch per Default als NotDefined eingeordnet wird, muss zumindest eine eindeutige ID definiert sein, die im Attribut *ID* aufgenommen werden kann.

Die Objekte der Klasse *ToDo*, sprich die einzelnen ToDos, können durch eine bidirektionale rekursive Assoziation (vgl. Abschnitt 3.3.5) miteinander in Verbindung gebracht werden. Das Attribut *ToDoType* als Identifier in Verbindung zu einem Einzelteil oder einem Werkzeug kategorisiert die ToDos des jeweiligen Einzelteils oder Werkzeugs. Die einzelnen Kategorien der ToDos eines Einzelteils oder eines Werkzeugs, ausgedrückt durch die diskreten Werte der enumeration *ToDoTypeDef*, stehen in einer bekannten chronologischen Reihenfolge zueinander. Anhand der rekursiven Assoziation in Verbindung mit den eindeutig definierten ToDoTypes für ein bestimmtes Einzelteil oder Werkzeug kann auch die chronologische Reihenfolge der ToDos berücksichtigt werden. Die Rollen *Successor* und *Precessor* an den Enden der rekursiven Assoziation mit optional beliebieger Multiplizität drücken aus, ob in Verbindung gebrachte ToDos als Vorgänger oder als Nachfolger zum aktuellen ausgewählten ToDo eingeordnet werden sollen.

Die Klasse *ToDo* besitzt zudem noch eine weitere rekursive Assoziation, in Form einer Komposition. Anhand dieser rekursiven Komposition kann in der ToDo-Klasse eine beliebige Baumstruktur realisiert werden. Dies ist für alle werkzeugspezifischen ToDos von Bedeutung, da diese immer in einer bestimmten Korrekturschleife stattfinden. Eine Korrekturschleife besagt, wie oft das Werkzeug schon durch Korrekturen verändert wurde. Komplizierte Werkzeuge können mehrere Korrekturschleifen durchlaufen, in denen auch wieder jeweils mehrere ToDos durchlaufen werden müssen. In einem ToDo mit dem spezifischen *ToDoType*: „CorrectionLoop“ wird im Attribut *ID* dementsprechend die Nummer der Korrekturschleife eingelesen. Theoretisch müssten also alle werkzeugspezifischen ToDos immer einem ToDo mit dem *ToDoType*: „CorrectionLoop“ und der jeweiligen Nummer der Korrekturschleife untergeordnet sein. Die Korrekturschleifen als solche können bereits als ToDo eingelesen werden, die dazugehörigen ToDos können aber (noch) nicht eindeutig ihrer jeweiligen Korrekturschleife zugeordnet werden. Die rekursive Komposition bietet dennoch die Möglichkeit, bei einer

verbesserten Aufnahme der Zugehörigkeiten von ToDos zu einer Korrekturschleife diese Struktur auch umzusetzen. In Bezug auf Einzelteile sind Korrekturschleifen aufgrund der Möglichkeit von Indexänderungen über die Indizes der Einzelteile obsolet.

In dem optionalen Attribut *Description* des Datentyps String wird die Beschreibung des jeweiligen ToDos eingelesen. Das optionale Attribut *Comment* des Datentyps String dient zur Aufnahme des aktuellen Kommentars zu einem ToDo, der immer von einer bestimmten Person in einem System hinterlegt ist. In den optionalen Attributen *JiraId* und *CdbDocNo* des Datentyps String werden zu einem ToDo zugehörige Dokumentennummern aufgenommen, anhand denen detailiertere Dokumente zu einem ToDo in anderen unternehmensinternen Datensystemen aufgerufen werden können. Das optionale Attribut *TrackingId* des Datentyps String nimmt bei Einzelteil oder Werkzeug - verlagerungen (*ToDoType*: „Transfer_18“) die Sendungs-verfolgungsnummer auf.

Die optionalen Attribute *PlannedStart*, *PlannendEnd*, *Started*, *Finished* und *Deadline* haben den Datentyp Date und nehmen daher den zeitlichen Verlauf eines ToDos auf. Sobald ein neues ToDo aufgenommen wird sollten auch direkt die Beiden Attribute *PlannedStart* und *PlannedStart* aufgenommen werden können. Diese definieren nach dem erstmaligen Anlegen eines ToDos den geplanten zeitlichen Rahmen. Muss ein ToDo zu einem bestimmten Zeitpunkt definitiv erledigt sein, wird eine Deadline definiert, die im gleichnamigen Attribut *Deadline* eingelesen wird. Neben den geplanten Start- und Endterminen werden über die Attribute *Started* und *Finished* ebenfalls die tatsächlichen Start- und Endtermine aufgenommen. Das Attribut *Finished* ist außerdem ausschlaggebend, ob ein ToDo über die in Abschnitt 4.3.4 beschriebende unidirketionale Assoziation zu einem Projekt (Objekt der Klasse *Project*) als *OpenToDo* zugeordnet wird oder nicht. Sobald einem ToDo ein Endtermin im Attribut *Finished* zugeordnet wurde, soll dieses nicht mehr als *OpenToDo* zur Klasse Project in Beziehung gesetzt

werden. Der gesamte Bearbeitungszeitraum eines ToDos vom Starttermin bis zum Endtermin wird im optionalen Attribut *DurationDays* des Datentyps Integer aufgenommen. Die eingelesene Zahl beschreibt die Dauer eines ToDos in Tagen.

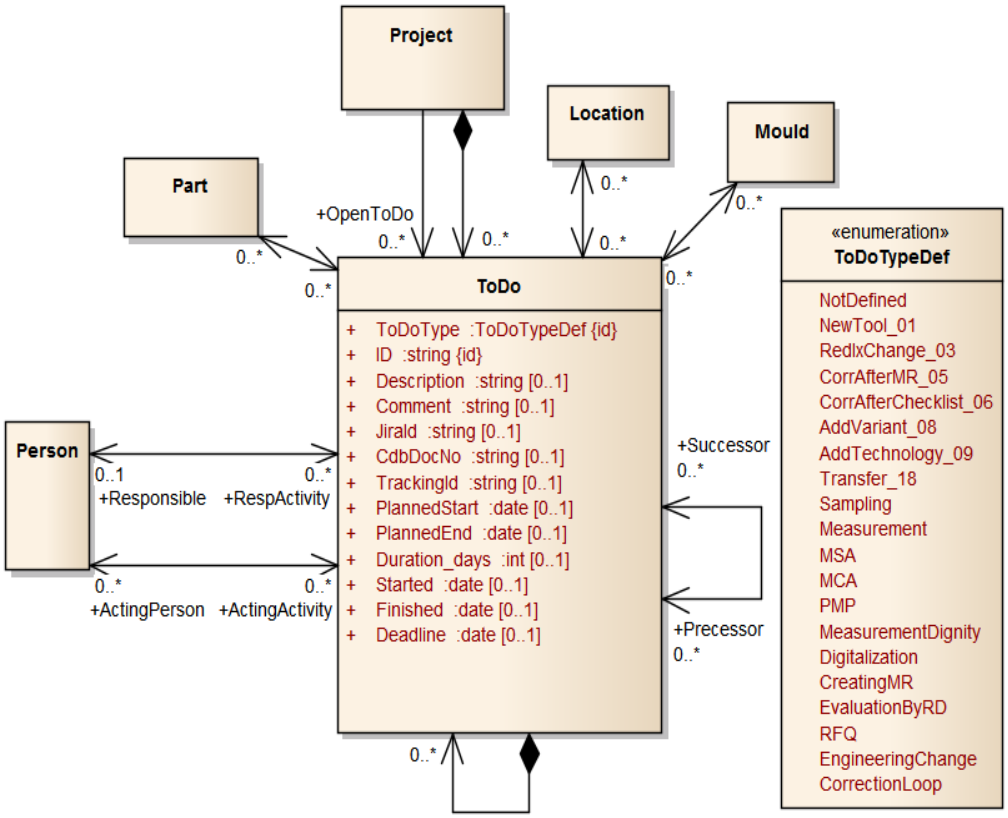


Abbildung 29: Klasse ToDo mit Assoziationen und enumeration

5 Erstellung eines Referenzmodells

5.1 Konstellation

Die Erstellung eines objektorientierten Referenzmodells mittels der UML hat in dieser Thesis einen besonderen Charakter. Als wirtschaftlichster und erfolgversprechendster Weg zur Identifizierung von Klassen eines Referenz-Klassenmodells wird die Analyse von bereits vorhandenen Modellen genannt (vgl. Abschnitt 3.4.3). Da im Kontext der (globalen) Koordination von Einzelteilen und Werkzeugen allerdings zum jetzigen Zeitpunkt keine Modelle zur Analyse vorliegen, wurde entgegen der in Abbildung 16 vorgestellten Reihenfolge zur Erstellung eines Referenzmodells und eines (unternehmens-)spezifischen Modells zunächst im vorherigen Fallbeispiel ein unternehmensspezifisches Datenmodell entwickelt. Das Datenmodell gepaart mit weiteren Informationsquellen bildet dann die Grundlage, anhand derer ein erstes grobes Referenzmodell für die Aufnahme und Verwaltung von Daten im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen erstellt werden kann. Unter Berücksichtigung der Vorgehensweise zur Erstellung von Referenzmodellen (vgl. Abschnitt 3.4.2) und dem vorherigen Fallbeispiel wird zur Referenzmodellerstellung ein objektorientiertes Referenz-Klassenmodell erstellt. Das Modell soll als Basismodell gesehen werden, mit zusätzlichen Empfehlungen für die zukünftige Erstellung von Erweiterungsmodellen. Aufgrund der eingeschränkten Informationsquellen im Rahmen dieser Referenzmodellierung und des Umfangs dieser Thesis wird zudem auf die Modellierung von Verhaltensvarianten und von zusätzlichen Erweiterungsmodellen unter dem Basismodell verzichtet. Der Fokus liegt dementsprechend auf der Abbildung von grundlegenden Strukturvarianten für die Aufnahme und Verwaltung von anfallenden Daten im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen. In Analogie zum unternehmensspezifischen Modell wird daher auch im Referenzmodell auf die Definition von Operationen verzichtet.

Für ein besseres Verständnis wird in Abbildung 30 das an die Situation dieser Thesis angepasste Vorgehensmodell zur Erstellung eines objektorientierten Referenzmodells gezeigt. Der grau hinterlegte Bereich beschreibt dabei die Schritte der Referenzmodellierung, die tatsächlich durchgeführt werden. Der restliche weiß hinterlegte Bereich beschreibt die Schritte, die auf der Basis zusätzlicher Informationsquellen und einem umfangreicheren Arbeitsrahmen noch zur Komplettierung dieses Referenzmodell durchgeführt werden müssten.

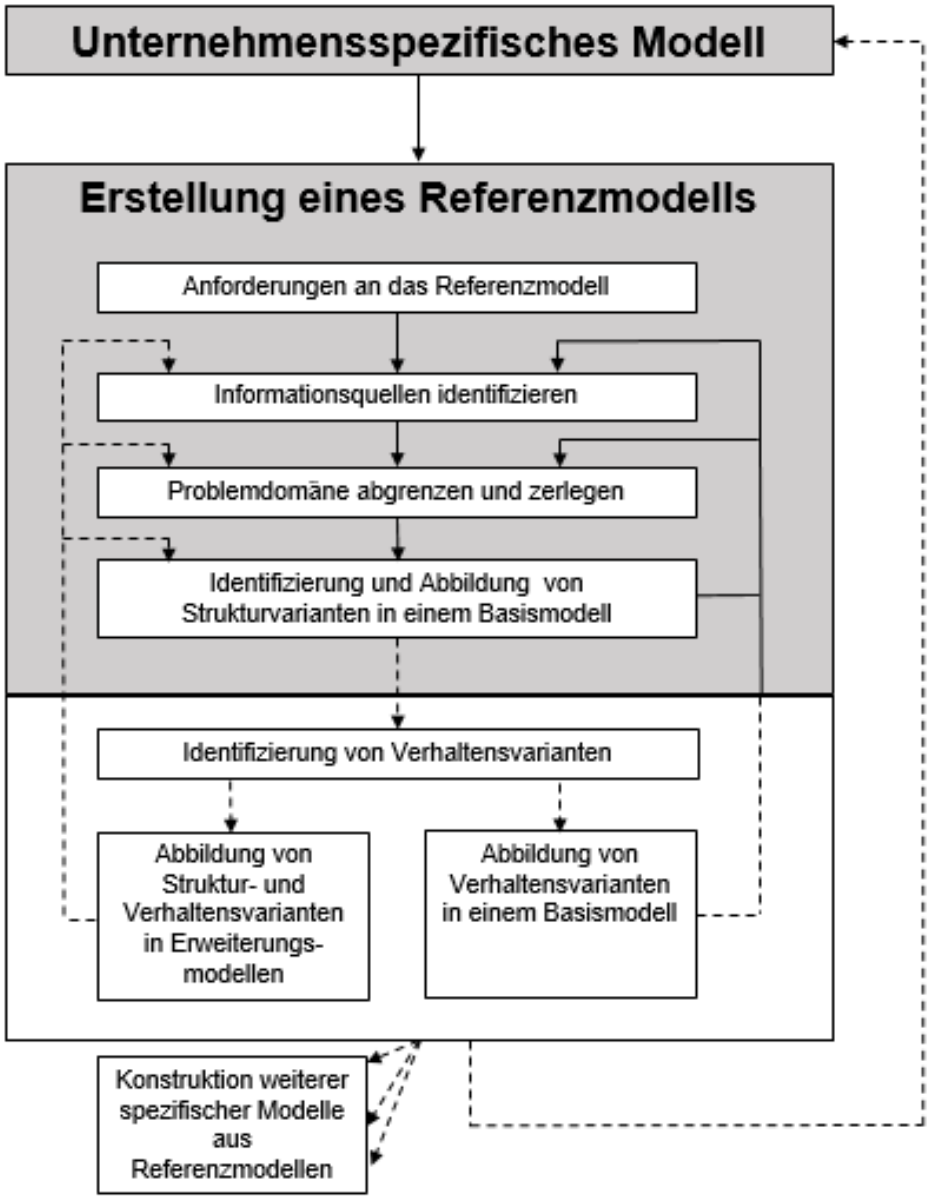


Abbildung 30: Angepasstes Vorgehensmodell zur Erstellung und Verwendung von objektorientierten Referenzmodellen (nach Schwegmann 1999)

5.2 Vorgehensweise

Für die *Anforderungen an das Referenzmodell* sei auf die in Abschnitt 3.4.1 in Tabelle 4 beschriebenen elementaren Anforderungen an Referenzmodelle verwiesen. Bei einer Spezifikation der Anforderungen in Bezug auf das zu modellierende Referenzmodell kann allerdings der Anforderung des modularen Aufbaus nur teilweise nachgekommen werden. Die Modellierung eines Referenz-Klassenmodells bietet einerseits eine sehr gute modulare Struktur. Um dieser Anforderung komplett nachzukommen, müsste das Basismodell jedoch noch in weitere Teilmodelle zerlegt werden. Die Anforderung nach Allgemeingültigkeit verlangt überdies die Sichtweisen unterschiedlicher potentieller Verwender zu berücksichtigen. Dieser Anforderung kann auch nur zum Teil genüge getragen werden, da auf keine bereits vorhandenen spezifischen Modelle zurückgegriffen werden kann.

Als *verwendete Informationsquellen* wurden neben dem in Kapitel vier entwickelten Fallbeispiel auch bereits existierende Referenzmodelle im erweiterten Kontext der Referenzmodellierung in Produktion und Logistik herangezogen. Damit sind insbesondere das objektorientierte Referenzmodell für die Lagerverwaltung von Schwegmann (vgl. 1999: S.185ff.), das nicht objektorientierte Referenzmodell maschinennaher Simulationskomponenten von Schlögl (vgl. 2000: S.151ff.) und das nicht objektorientierte Referenzmodell für Umschlag und Kommissionierung von Scholz (vgl. 2000: S.211ff.) gemeint. Des Weiteren wurden während der Erstellung des Referenzmodells kontinuierlich Experten mit langjähriger Erfahrung im Bereich der globalen Koordination von Einzelteilen und Werkzeugen miteinbezogen, um verschiedene Meinungen berücksichtigen zu können und so eine höhere allgemeine Akzeptanz zu schaffen.

Die *Abgrenzung und Zerlegung der Problemdomäne* erfolgt nach der aus Abschnitt 3.4.2 bekannten Vorgehensempfehlungen. Dementsprechend wird eine Checkliste zur Problembereichsabgrenzung (vgl. Tabelle 6) angelegt, in welcher die verschiedenen zu berücksichtigenden Aspekte des Referenzmodells zur Aufnahme und Verwaltung von Daten im Rahmen

der globalen Koordination von Einzelteilen und Werkzeugen aufgeführt und gewichtet werden.

Tabelle 6: Checkliste zur Problembereichsabgrenzung

Aspekt	Beschreibung	Gewichtung	Status
Abbildung von Einzelteilen	Aufnahme der spezifischen Daten von Einzelteilen	5	Berücksichtigt
Abbildung von Werkzeugen	Aufnahme der spezifischen Daten von Werkzeugen	5	Berücksichtigt
Abbildung von Koordinationsschritten	Aufnahme von koordinations-spezifischen Daten	5	Berücksichtigt
Abbildung von Aufträgen	Aufnahme der spezifischen Daten von übergeordneten Aufträgen	5	Berücksichtigt
Abbildung von Maschinen	Aufnahme von maschinenspezifischen Daten	4	Berücksichtigt
Abbildung von Standorten	Aufnahme von spezifischen Daten eines Standortes	4	Berücksichtigt
Abbildung von Materialien	Aufnahme der Materialdaten von Einzeilteilen, Werkzeugen oder Maschinen	3	Berücksichtigt

Abbildung von Geometrien	Aufnahme der Geometriedaten von Einzelteilen oder Werkezugen	3	Berücksichtigt
Abbildung von Personen	Aufnahme von personenspezifischen Daten involvierter Personen	3	Nicht berücksichtigt
Abbildung von Kunden	Aufnahme von kundenspezifischen Daten	2	Nicht berücksichtgt
Abbildung von Kavitäten	Aufnahme der werkzeugspezifischen Kavitäten	2	Nicht berücksichtigt
Abbildung von Kostenfaktoren	Aufnahme von Daten bezüglich Budgets, Ausgaben, etc.	2	Nicht berücksichtigt
Abbildung von Transporteinrichtungen	Aufnahme der Daten von Transporteinrichtungen (Transporthilfsmittel, Fördermittel, etc.)	1	Nicht berücksichtigt

5.3 Referenzmodell

Die *Identifizierung und Abbildung von Strukturvarianten in einem Basismodell* bilden den letzten Schritt zur Erstellung des Referenzmodells. Da nachfolgend ein objektorientiertes Referenz-Klassenmodell erstellt wird, wird zur Identifizierung und Abbildung von Strukturvarianten auf das Prinzip der Modellierung von Klassen objektorientierter Referenz-Klassenmodelle zurückgegriffen (vgl. Abschnitt 3.4.2). Als Grundlage zur Identifizierung dieser Klassen sei nochmal auf die im vorherigen Abschnitt 5.2 vorgestellten Anforderungen, Informationsquellen und auf die Checkliste

zur Problembereichsabgrenzung verwiesen. Die Abbildung der Klassen im Referenz-Klassenmodell findet unter Berücksichtigung der Gestaltungsempfehlungen zur Abbildung von Klassen in Referenz-Klassenmodelle statt (vgl. Abschnitt 3.4.3). Aufgrund der bekannten zunehmenden Globalisierung in der Koordination von Einzelteilen und Werkzeugen (vgl. Kapitel 1) wird auch das Referenzmodell in englischer Sprache modelliert. Alle nicht explizit beschriebenen Multiplizitäten an Assoziationen und Aggregationen werden als optional beliebig [0..*] verstanden. Das komplette Referenzmodell ist im Anhang (Abbildung 32) wiederzufinden.

Anders als im unternehmensspezifischen Modell besitzt das Referenzmodell keine Root-Klasse, da es als nicht sinnvoll erachtet wird eine Klasse zu modellieren, die nicht ohne weiteres selbsterklärend ist. Um aber weiterhin die Vorteile einer baumartigen Datenstruktur nutzen zu können (vgl. Abschnitt 4.3.1) schließt die Klasse *Order* als Startpunkt zumindest die primären Klassen des Referenzmodells über Aggregationen in sich ein. Die primären oder relevantesten Klassen dieses Modells sind an ihrer hohen Gewichtung (>3) in der Checkliste zu identifizieren.

Die Klasse ***Order*** (auf Deutsch Auftrag) kann als allgemeingültiges Pendant der Project-Klasse aus dem unternehmensspezifischen Modell gesehen werden, da in Abhängigkeit von der Art des Auftrags nicht zwangsläufig in Projekten gedacht wird. Für eine genauere Bestimmung (Spezialisierung) des jeweiligen Auftrags könnten in Erweiterungsmodellen zusätzliche Varianten zur Order-Klasse definiert werden. Einen guten Ansatzpunkt dafür bietet Scholz (vgl. Wenzel 2000: S.219) in seinem Referenzmodell für Umschlag und Kommissionierung, in welchem dieser eine hierarchische Anordnung der Auftragsklassen darstellt und unter anderem in externe, interne, kunden- und lieferantspezifische Aufträge unterscheidet. In diesem Zusammenhang könnten dann auch kundenspezifische Daten (z.B. Kundenname etc.) aufgenommen werden, die im Basismodell nicht berücksichtigt werden. Im Attribut *OrderID* der Order-Klasse werden die Daten aufgenommen, die den Auftrag beschreiben.

Aufgrund der Anforderung nach Allgemeingültigkeit und der Gestaltungsempfehlungen für die objektorientierte Referenzmodellierung wird dieses Attribut und auch alle anderen Attribute im Referenzmodell nicht weiter spezifiziert (vgl. Abschnitt 3.4.3). Es wird aber aus Gründen der Datenqualität dringend empfohlen, Attribute im unternehmensspezifischen Modell (abgeleitet aus dem Referenzmodell) in Bezug auf die aufzunehmenden Daten anhand der Charakteristika aus Tabelle 3 genauer zu modellieren. Dazu soll an dieser Stelle auch nochmal an das Modellieren von Identifier {id] erinnert werden (vgl. Abschnitt 4.3), die im Kontext einer guten Datenqualität (vgl. Abschnitt 2.3) für eine redundanzfreie Aufnahme und Verwaltung der Daten sorgen. Der Vorteil dieser anfangs hohen Allgemeingültigkeit ist, dass als Identifier eines Auftrags sowohl Nummern als auch interne Bezeichnungen oder jede andere Art von Daten zur Identifikation eines Auftrages aufgenommen werden können. Besteht die Notwendigkeit zur Beschreibung eines Auftrages anhand von mehreren unterschiedlichen Daten, müsste unter dem Aspekt der sauberen Datenaufnahme im unternehmensspezifischen Modell ggf. das Attribut OrderID in mehrere Attribute unterteilt werden, sofern dies im zugehörigen Erweiterungsmodell noch nicht geschehen ist. Es ist in jedem Fall aus Gründen der Datenqualität davon abzusehen, mehrere Daten in einem Attribut aufzunehmen.

Als nächstes wird als zentrales Element dieses Referenzmodells die Klasse ***Part*** modelliert, die zur Aufnahme der einzelteilspezifischen Daten gedacht ist. Zur vollständigen Beschreibung der Einzelteile werden hier die selbsterklärenden Attribute *PartNumber*, *PartIndex* und *PartName* modelliert. In Abhängigkeit des Konkretisierungsgrads der Beschreibung eines Einzelteils können je nach Referenzmodellverwender unter Berücksichtigung der Eindeutigkeit von Objekten in Klassen, ein Attribut oder mehrere Attribute im unternehmensspezifischen Modell als optional definiert oder entfernt werden. Die Part-Klasse ist über eine Aggregation mit der Order-Klasse und über Assoziationen mit den Klassen *Tool*, *Material*, *Geometry*, und *Location* verbunden. Insbesondere durch die Klassen *Material* und *Geometry* lässt sich ein Einzelteil relativ genau

beschreiben, sodass zusätzliche Erweiterungsmodelle der Part-Klasse nicht notwendig sind. Sollte ein Auftrag sowohl eigens angefertigte Einzelteile (Make-Parts) als auch extern hergestellte Kaufteile (Buy-Parts) beinhalten, reicht es aus diese beiden Arten anhand einer enumeration (sofern das Konzept der enumeration bekannt ist) in der Klasse *Part* zu unterscheiden. Da sich an den Attributen der Klasse *Part* ansonsten nichts ändert, sollte davon abgesehen werden an dieser Stelle in Erweiterungsmodelle zu unterscheiden.

Die Klasse ***Geometry*** ist aus Gründen der Allgemeingültigkeit nicht mehr wie zuvor im unternehmensspezifischen Modell Oberklasse der Part-Klasse. So wird verhindert, dass der Modellverwender vor jedem Einzelteil auch Daten zu dessen Geometrie aufnehmen muss. Aus rein logischer Betrachtung besitzt jedes Einzelteil auch eine Geometrie. Allerdings kann es speziell bei extern hergestellten Kaufteilen vorkommen, dass die Aufnahme von geometrischen Daten zu einem Einzelteil nicht zwangsläufig notwendig ist. Würde die Geometry-Klasse als Oberklasse der Part-Klasse modelliert werden, wäre es demnach nicht möglich ein Einzelteil auch ohne seine Geometrien aufzunehmen. Dementsprechend werden an dieser Stelle die Anforderungen nach Allgemeingültigkeit und Wiederverwendbarkeit der logischen Modellierung vorgezogen. Nichtsdestotrotz können und sollten so auch Daten zu den Geometrien und Werkstoffen (Objekte der Klasse *Material*) der Einzelteile aufgenommen werden, da sich Einzelteile insbesondere über ihre Geometrien definieren. In der Klasse *Geometrie* werden die selbsterklärenden Attribute *GeometryNumber*, *GeometryIndex* (nicht zwangsläufig vorhanden) und Geometry*Name* modelliert. In Erweiterungsmodellen zur Geometrie-Klasse könnten die Varianten der Geometriedaten (z.B. 3D-CAD-Daten, 2D-Zeichnungsdaten, etc.) spezialisiert werden.

In der Klasse ***Material*** wird zusätzlich die Möglichkeit geboten, Werkstoffdaten zu einem Einzelteil oder zu einem Werkzeug aufzunehmen. Da Werkstoffe in Abhängigkeit ihrer Art über Nummern oder Namen iden-

tifiziert werden können, wird in dieser Klasse das Attribut *MaterialID* modelliert. Ähnlich wie in der Order-Klasse kann das Attribut *MaterialID* im unternehmensspezifischen Modell bei Notwendigkeit weiter unterteilt werden.

Da auch dieses Referenzmodell im Rahmen einer globalen Koordination von Einzelteilen und Werkzeugen modelliert wird, wird bereits im Basismodell die Klasse ***Location*** mitaufgenommen. Hier wird im Attribut *Name* der Name des Standortes aufgenommen. In Abhängigkeit des Umfangs des zu koordinierenden Auftrags und der teilhabenden Standorte können zusätzliche Attribute zur Aufnahme standortspezifischer Daten modelliert werden. Die unterschiedlichen Standortvarianten (Produktionsstandort, Kundenstandort, Lieferentenstandort, etc.) könnten in Erweiterungsmodellen genauer definiert werden. Aus Gründen der sauberen Datenhaltung sollte auch im Basismodell ein Einzelteil (Objekt der Klasse *Part*) immer nur maximal einem Standort [0..1] gleichzeitig zugeordnet werden.

Werkzeugspezifische Daten werden in der Klasse ***Tool*** aufgenommen. Zur Beschreibung eines Werkzeugs werden die Attribute *ToolNumber* und *ToolName* modelliert. Ähnlich wie schon in der Part-Klasse kann auch hier in Abhängigkeit des Konkretisierungsgrads der Beschreibung eines Werkzeugs im unternehmensspezifischen Modell ein Attribut auf optional gesetzt oder entfernt werden. Um der Anforderung nach Allgemeingültigkeit nachzukommen werden im Basismodell keine Kavitäten zu einem Werkzeug modelliert, da ein Werkzeug in Abhängigkeit seines Zwecks nicht zwangsläufig Kavitäten besitzt. An dieser Stelle müsste in Erweiterungsmodellen auf die Art und den Zweck der zu koordinierenden Werkzeuge genauer eingegangen werden. In Bezug auf das Fallbeispiel müsste dann in einem spezifischen Erweiterungsmodell zu Spritzgießwerkzeugen auch die Cavity-Klasse wiederzufinden sein. Die Tool-Klasse ist über eine Aggregation mit der Order-Klasse verbunden und hat Assoziationen zu den Klassen *Part*, *Geometry*, *Machine*, *Material* und *Location*. Ein Werkzeug kann dabei maximal einem Standort (Objekt der Klasse *Location*) gleichzeitig zugeordnet werden.

Die Klasse ***Machine*** müsste streng genommen Oberklasse der Klasse *Tool* sein, bzw. müsste dazwischen sogar noch eine Klasse *Baugruppe* modelliert werden. Genaueres dazu ist dem Produktmodell des Referenzmodells maschinennaher Simulationskomponenten von Schlögl zu entnehmen (vgl. Wenzel 2000: S.156). Im Rahmen der Koordination von Einzelteilen und Werkzeugen wird diese Struktur allerdings nicht als sinnvoll erachtet, da Werkzeuge auch ohne Bezug zu Maschinen oder Baugruppen aufgenommen werden können müssen. Dementsprechend ist die Klasse *Machine* als Aggregation der Order-Klasse modelliert und nur über eine Assoziation mit der Tool-Klasse verbunden. Die zur Koordination von Einzelteilen und Werkzeugen aufzunehmenden Maschinendaten haben einen sehr schwer zu erfassenden Rahmen. Daher werden in der Machine-Klasse im Basismodell keine weiteren Attribute modelliert. Erweiterungsmodelle der Machine-Klasse müssten stark in Abhängigkeit zu den Erweiterungsmodellen der Tool-Klasse modelliert werden, da die Art der Maschine und damit auch die in dieser Klasse aufzunehmenden Daten primär von der Art des Werkzeugs abhängig sind. Im Zweifelsfall obliegt es dem Referenzmodellverwender im unternehmensspezifischen Modell die für seine Zwecke notwendigen Attribute zur Aufnahme der maschinenspezifischen Daten zu modellieren. Eine Maschine kann darüber hinaus maximal einem Standort gleichzeitig zugeordnet werden.

In der Klasse ***CoordinationStep*** werden die koordinationsspezifischen Daten in Bezug auf Einzelteile und Werkzeuge aufgenommen. Wie genau solche Koordinationsschritte bzw. koordinationsspezifischen Daten Aussehen ist primär unternehmensabhängig. Im Basismodell wird im Attribut CoordinationStep*ID* die eindeutige Bezeichnung des Koordinationsschritts/ der Koordinationsaufgabe aufgenommen. Hiermit sind eindeutige Identifikationsnummern bzw. Identifikationsbezeichnungen von Koordinationsschritten gemeint, die für eine saubere Zuordnung angelegt und aufgenommen werden müssen. Um Unklarheiten bei der Aufnahme und Verwaltung von Koordinationsschritten zu vermeiden, wird zu dem CoordinationStepID-Attribut das Attribut *Description* modelliert, in dem

die Beschreibung bzw. die Erklärung des Koordinationsschrittes aufgenommen wird. Dass dieses Attribut schon im Basismodell mitaufgenommen wird hängt damit zusammen, dass die Klasse *CoordinationStep* aufgrund ihrer vielseitigen Interpretationsmöglichkeiten eine genauere Definition ihrer einzelnen Objekte verlangt, die durch die reine Idenfikationsnummer/ -bezeichnung (Attribut CoordinationStep*ID*) in der Regel nicht gegeben ist. Über die Assoziation zur Klasse *Location* können einem Koordinationsschritt zusätzlich ein oder mehrere Standorte zugeordnet werden.

In Erweiterungsmodellen könnte die CoordinationStep-Klasse grundsätzlich in Koordinationsschritte für Einzelteile und Koordinationsschritte für Werkzeuge unterteilt werden. Allerdings ist auch hier die Definition von allgemeingültigen Attributen ohne genaueres Kontextwissen über die zu koordinierenden Daten im Unternehmen kaum möglich. Die detaillierte Modellierung von Attributen der CoordinationStep-Klasse obliegt daher dem Referenzmodellverwender im unternehmensspezifischen Modell. Das weitere Spezifizieren der Koordinationsschritte im Basismodell mit zusätzlichen Attributen ist aus Gründen der Allgemeingültigkeit und Wiederverwendbarkeit nicht sinnvoll. Dennoch werden dem Referenzmodellverwender an dieser Stelle einige Anregungen zur Modellierung der Klasse *CoordinationStep* im unternehmensspezifischen Modell mitgegeben.

Sofern der Referenzmodellverwender mit dem Konzept der enumeration vertraut ist, empfiehlt es sich aus Übersichtlichkeitsgründen im unternehmensspezifischen Modell eine enumeration mit allen Koordinationsschritten zu modellieren, sofern deren Anzahl handhabbar ist (vgl. Abschnitt 4.3.12). Die enumeration beinhaltet dann nicht die Identifikationsnummern bzw. die Identifikationsbezeichnungen der Koordinationsschritte, sondern eine übergeordnete Bezeichnung des Koordinationsschrittes, um diesen im Gesamtkontext einordnen zu können. Für UML-versierte Referenzmodellverwender sei nochmal auf die Struktur der Klasse *ToDo*

im Fallbeispiel verwiesen (vgl. Abschnitt 4.3.12), die es ermöglicht einzelne Objekte auch in einer Baumstruktur untereinander anzulegen und durch Vorgänger und Nachfolger eine Vorwärtsplanung zu realisieren. Darüber hinaus kann in der CoordinationStep-Klasse durch Modellieren von Attributen des Datentyps *Date* auch der zeitlichen Rahmen eines Koordinationsschritts aufgenommen werden. Da im Basismodell keine personenspezifischen Daten aufgenommen werden, kann bei Bedarf in dieser Klasse über ein zusätzliches Attribut (z.B. *ResponsiblePerson*) eine für den Koordinationsschritt verantwortliche Person aufgenommen werden. Sollte die Notwendigkeit bestehen mehrere Personen aufzunehmen, müsste aus Gründen einer sauberen Datenhaltung eine zusätzliche Klasse *Person* definiert werden (vgl. Abschnitt 4.3.7).

6 Kritische Würdigung

Die vorliegende Arbeit leistet einen Beitrag, um den Grundstein für ein intelligentes Datenmanagement im Unternehmen zu legen. Hierfür wurde im Bereich der produktionsübergreifenden globalen Koordination von Einzelteilen und Werkzeugen der Leopold Kostal GmbH & Co. KG ein objektorientiertes Datenmodell entworfen. Auf der Basis dieses Datenmodells besteht zukünftig die Möglichkeit, das veraltete Konzept zur Aufnahme und Verwaltung aller in der Projektkoordination anfallenden Daten durch eine intelligentere Lösung zu ersetzen. Auf diese Weise kann vor allem die Datenqualität in der Projektkoordination nachhaltig verbessert werden. Das Datenmanagement endet allerdings nicht mit der Datenmodellierung. Um langfristig die Datenqualität im Unternehmen zu verbessern und der Forderung nach einem intelligenten Datenmanagement nachzukommen, müssen auch die Bereiche der Datenadministration und Datenbankadministration berücksichtigt werden (vgl. Abschnitt 2.3). Eine saubere Datenmodellierung unter Miteinbeziehung der involvierten Personen bzw. der Datenverwender bietet dafür die perfekte Grundlage. Dennoch generiert erst das Zusammenspiel aller drei Funktionen des Datenmanagements einen effektiven Mehrwert für das Unternehmen und rechtfertigt dann auch den vorher geleisteten Aufwand zur Datenmodellierung aus wirtschaftlicher Perspektive.

Die Erstellung eines Referenzmodells im Bereich der Koordination von Einzelteilen und Werkzeugen ist durchaus kritischer zu betrachten. Aufgrund der Nichtexistenz bzw. der nicht möglichen Einsicht von Datenmodellen in diesem Bereich, wurde das Vorgehensmodell zur Erstellung eines Referenzmodells (vgl. Abschnitt 3.4.2) inhaltlich umgedreht. Dieser Schritt war notwendig, da die Analyse spezifischer Modelle der erfolgversprechendste Weg zur Klassenidentifikation für die Referenzmodellerstellung ist (vgl. Abschnitt 3.4.3). Aufgrund des fehlenden Bezugs zu anderen unternehmensspezifischen Modellen kann den Anforderungen

nach Allgemeingültigkeit, Wiederverwendbarkeit und Akzeptanz in diesem Referenzmodell nur in einem eingeschränkten Rahmen nachgekommen werden. Für ein vollständiges Referenzmodell müssten zudem die im Basismodell genannten Empfehlungen für zusätzliche Varianten in Erweiterungsmodellen genauer spezialisiert werden. Hierfür müssten zwangsläufig weitere Informationsquellen hinzugezogen werden. Angesichts des eingeschränkten Umfangs dieser Bachelorthesis wurde zudem im Referenzmodell und im unternehmensspezifischen Modell auf die Modellierung von Operationen verzichtet. Diese müssten zur Vollständigkeit der Modelle zwangsläufig noch ergänzt werden. Das erstellte Referenz-Klassenmodell zur Koordination von Einzelteilen und Werkzeugen versteht sich dementsprechend nicht als vollständiges Referenzmodell. Dennoch kann der Prozess der Erstellung eines unternehmensspezifischen Datenmodells im Bereich der globalen Koordination von Einzelteilen und Werkzeugen unter Berücksichtigung des Referenzmodells beschleunigt und verbessert werden. Zuletzt soll an dieser Stelle nochmal darauf hingewiesen werden, dass auch ein vollständiges Referenzmodell niemals eigene Modellierungsüberlegungen ersetzen kann.

7 Zusammenfassung und Ausblick

Die vorliegende Arbeit beschäftigt sich mit dem Bereich des Datenmanagements in der Produktion. Aufgrund des eingeschränkten Umfangs einer Bachelorthesis wurde primär die Datenmodellierung als eine der klassischen Funktionen des Datenmanagements untersucht. Da die Arbeit in Kooperation mit der Projektkoordination für Einzelteile der Leopold Kostal GmbH & Co. KG entstanden ist, wurde der Bereich des Datenmanagements zu Beginn der Arbeit zunächst im Umfeld der Automobilproduktion durchleuchtet. Hier wurde eine deutliche Komplexitätssteigerung in Bezug auf die Menge und Varietät der Daten im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen festgestellt, die sich auch in der Praxis bestätigte.

Ziel der Arbeit war daher, anhand einer Datenmodellierung eine zeitgemäße Lösung zur Aufnahme und Verwaltung aller in der Projektkoordination anfallenden Daten zu entwickeln. Diesbezüglich wurde auf objektorientierte Modellierungstechniken zurückgegriffen. Als Resultat wurde dementsprechend ein objektorientiertes Datenmodell entworfen, welches die Aufnahme und Verwaltung aller anfallenden Daten im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen ermöglicht. Zur Modellierung des Datenmodells wurde im speziellen die UML benutzt. Die UML als objektorientierte Modellierungssprache konnte durch ihre vielseitigen und detaillierten Darstellungsmöglichkeiten zur Aufnahme und Verwaltung von Daten überzeugen. Gleichzeitig konnte durch das objektorientierte Paradigma eine realitätsnahe Modellierung erfolgen. Die dadurch hervorgerufene hohe Verständlichkeit erlaubte eine permanente Miteinbeziehung der Datenverwender bei der Erstellung des Datenmodells, obgleich diese nicht mit dem Prinzip der Datenmodellierung vertraut waren. Als Ergebnis wurde ein Klassendiagramm modelliert. Die komplexen Strukturen in der Projektkoordination in Bezug auf alle aufzunehmenden Daten wurden zum besseren Verständnis gesondert in den einzelnen Klassen des Klassendiagramms erläutert (vgl.

Kapitel 4.3.1 bis 4.3.12). Das unternehmensspezifische Datenmodell als Hauptziel der Arbeit bildet den Grundstein, um langfristig ein intelligentes Datenmanagement im Unternehmen implementieren zu können.

Auf der Basis des erstellten Datenmodells im Unternehmen wurde zusätzlich ein erstes grobes Referenzmodell im Rahmen der globalen Koordination von Einzelteilen und Werkzeugen erstellt. Dass hier entgegen der Literatur zuerst das unternehmensspezifische Datenmodell erstellt wurde, ist der Nichtexistenz bzw. der Unauffindbarkeit anderer unternehmensspezifischer Modelle im Bereich der globalen Koordination von Einzelteilen und Werkzeugen geschuldet. Auf diese Weise konnte zumindest das vorher erstellte Datenmodell als direkte Informationsquelle hinzugezogen werden. Im Bereich der Referenzmodellierung wurde grundsätzlich festgestellt, dass auch zur Referenzmodellerstellung objektorientierte Modellierungsansätze existieren. Die Erstellung des Referenzmodells fand demzufolge anhand der Vorgehensweise zur Erstellung objektorientierter Referenzmodelle nach Schwegmann (vgl. 1999: S.165ff.) statt. Hierzu wurden allgemeine Anforderungen an das Referenzmodell formuliert, zusätzliche Informationsquellen neben dem zuvor erstellten Datenmodell identifiziert und eine Checkliste zur Problembereichsabgrenzung erstellt. Auf dieser Basis konnte dann als Ergebnis der Referenzmodellierung ein Referenz-Klassenmodell erstellt werden, das als Basismodell zur Aufnahme und Verwaltung der in der Koordination von Einzelteilen und Werkzeugen anzufallenden Daten zu sehen ist.

Die Trends in der Automobilproduktion deuten auf eine fortschreitende Komplexitätssteigerung in Bezug auf die Anzahl und Varietät der Daten im Umfeld der Automobilzulieferer hin. Dabei stellen Daten als Fundament für Informationen für Unternehmen im Umfeld der Automobilproduktion im Generellen und für die globale Koordination von Einzelteilen und Werkzeugen im Speziellen eine kostbare Ressource dar. Ohne eine Veränderung im Hinblick auf die Art und Weise des Datenmanagements der Projektkoordination für Einzelteile der Leopold Kostal GmbH & Co.

KG wird die Qualität der Daten zwangsläufig auf ein nicht mehr hinnehmbares Niveau zusteuern. Als Konsequenz daraus müssten wieder suboptimale Übergangslösungen eingeführt werden, um zumindest das Alltagsgeschäft am Laufen zu halten. Das entwickelte Datenmodell soll daher als Anstoß gesehen werden, um das Datenmanagement in der Projektkoordination langfristig wieder in eine erfolgversprechendere Richtung zu führen. Mit dem Datenmodell als Grundstein und unter Berücksichtigung der im vorherigen Kapitel genannten Empfehlungen für nachfolgend durchzuführende Schritte kann das gegenwärtige Datenmanagement in der Projektkoordination inkrementell durch eine intelligentere und damit zukunftsfähige Version abgelöst werden.

Literaturverzeichnis

- Abts, D.; Mülder, W. (2017) Grundkurs Wirtschaftsinformatik. Eine kompakte und praxisorientierte Einführung. 9.Auflage. Wiesbaden: Springer Vieweg

- Apel, D.; Behme, W.; Eberlein, R.; Merighi, C. (2015) Datenqualität erfolgreich steuern. Praxislösungen für Business-Intelligence-Projekte. 3. Auflage. Heidelberg: dpunkt. Verlag.

- Balzert, H. (2007) UML 2 in 5 Tagen. Der schnelle Einstieg in die Objektorientierung. Herdecke: W3L-Verlag

- Becker, J.; Delfmann, P. (2004) Referenzmodellierung. Grundlagen, Techniken und domänenbezogene Anwendung. Heidelberg: Physica-Verlag

- Becker, J.; Probandt, W.; Vering, O. (2012) Grundsätze ordnungsmäßiger Modellierung. Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement. Berlin: Springer Gabler

- Bratzel, S.; Retterath, G.; Hauke, N. (2015) Automobilzulieferer in Bewegung. Strategische Herausforderungen für mittelständische Unternehmen in einem turbulenten Umfeld. 1.Auflage. Baden-Baden: Nomos edition sigma

- Dangel, R. (2015) Spritzgießwerkzeuge für Einsteiger. München: Hanser

- Dippold, R.; Meier, A.; Schnider, W.; Schwinn, K. (2005) Unternehmensweites Datenmanagement. Von der Datenbankadministration bis zum Informationsmanagement. 4.Auflage. Wiesbaden: Stephen Fedtke

- Dobing, B.; Parsons, J. (2006) How UML IS USED. In: COMMUNICATIONS of THE ACM Vol.49, No. 5, S.109-113

- Dudenredaktion (o.J.) „Intelligenz“ auf Duden online. <https://www.duden.de/rechtschreibung/Intelligenz> Abrufdatum: 19.06.2018

- Feiks, M. (2016) Datenerhebung mit Excel. Eine Anleitung zur Umsetzung von Inhaltsanalysen und Befragungen. Wiesbaden: Springer VS

- Gutenschwager, K.; Rabe, M.; Spieckermann, S.; Wenzel, S. (2017) Simulation in Produktion und Logistik. Grundlagen und Anwendungen. Berlin: Springer Vieweg

- Hab, G.; Wagner, R. (2017) Projektmanagement in der Automobilindustrie. Effizientes Management von Fahrzeugprojekten entlang der Wertschöpfungskette. 5.Auflage. Wiesbaden: Springer Gabler

- Helfert, M.; Herrmann, C.; Strauch, B. (2001) Datenqualitätsmanagement. Universität St. Gallen, Institut für Wirtschaftsinformatik, Bericht Nr.: BE HSG/CC DW2/02

- Kaya, D. (2007) Nutzung betrieblicher E-Mail- und Intranet-Systeme für gewerkschaftliche Zwecke. In: Schriften zum Sozial- und Arbeitsrecht, Band 263. Berlin: Duncker & Humblot

- Kinkel, S.; Zanker, C. (2007) Globale Produktionsstrategien in der Automobilzuliefererindustrie. Erfolgsmuster und zukunftsorientierte Methoden zur Standortbewertung. Berlin, Heidelberg: Springer-Verlag

- Krcmar, H. (2010) Informationsmanagement. 5. Auflage. Berlin: Springer

- Mertens, P. (2013) Integrierte Informationsverarbeitung 1. Operative Systeme in der Industrie. 18. Auflage. Wiesbaden: Springer Gabler

- Mertens, P.; Bodendorf, F.; König, W. (2017) Grundzüge der Wirtschaftsinformatik. 12., grundlegend überarbeitete Auflage. Berlin: Springer Gabler

- Rahm, E.; Saake, G.; Sattler, K.-U. (2015) Verteiltes und paralleles Datenmanagement. Von verteilten Datenbanken zu Big Data und Cloud. Berling: Springer Vieweg

- Rupp, C.; Queins, S. (2012) UML2 glasklar. Praxiswissen für die UML-Modellierung. 4. Auflage. München: Hanser

- SAP (o.J.) SAP Glossar. <https://help.sap.com/doc/saphelp_di46c2/4.6C2/de-DE/35/26b2fcafab52b9e10000009b38f974/content.htm?no_cache=true> Abrufdatum: 19.06.2018

- Seemann, J.; von Gudenberg, J. W. (2006) Software-Entwurf mit UML 2. Objektorientierte Modellierung mit Beispielen in Java. 2. Auflage. Berlin, Heidelberg: Springer-Verlag

- Schlagheck, B. (2000) Objektorientierte Referenzmodelle für das Prozess- und Projektcontrolling. Grundlagen - Konstruktion – Anwendungsmöglichkeiten. Dissertation, Universität Münster. Wiesbaden: Deutscher Universitätsverlag

- Schlögl, W. (2000) Referenzmodell maschinennaher Simulationskomponenten. In: Wenzel, Sigrid (Eds.): Referenzmodelle für die Simulation in Produktion und Logistik. Ghent: SCS – The Society for Computer Simulation International S.151-170

- Scholz, I. (2000) Referenzmodell für Umschlag & Kommissionierung. In: Wenzel, Sigrid (Eds.): Referenzmodelle für die Simulation in Produktion und Logistik. Ghent: SCS – The Society for Computer Simulation International S.211-232

- Schütte, R. (1998) Grundsätze ordnungsmäßiger Referenzmodellierung. Konstruktion konfigurations- und anpassungsorientierter Modelle. Dissertation, Universität Münster. Wiesbaden: Gabler Verlag

- Schwegmann, A. (1999) Objektorientierte Referenzmodellierung. Theoretische Grundlagen und praktische Anwendung. Dissertation, Universität Münster. Wiesbaden: Deutscher Universitätsverlag

- Stachowiak, H. (1973) Allgemeine Modelltheorie. Wien: Springer-Verlag

- Stahl, R.; Staab, P. (2017) Die Vermessung des Datenuniversums. Berlin: Springer Vieweg

- Staud, J. L. (2010) Unternehmensmodellierung. Objektorientierte Theorie und Praxis mit UML 2.0. Berlin, Heidelberg: Springer-Verlag

- Verband der Automobilindustrie (2017) Tatsachen und Zahlen. Berlin: VDA

- Vieweg, I.; Werner, C.; Wagner, K.-P.; Hüttl, T.; Backin, D. (2012) Einführung Wirtschaftsinformatik. IT-Grundwissen für Studium und Praxis. Wiesbaden: Springer Gabler

- Vom Brocke, J. (2002) Referenzmodellierung. Gestaltung und Verteilung von Konstruktionsprozessen. Dissertation, Universität Münster. Berlin: Logos Verlag

- Würthele, V. G. (2003) Datenqualitätsmetrik für Informationsprozesse. Datenqualitätsmanagement mittels ganzheitlicher Messung der Datenqualität. unv. Dissertation, ETH Zürich

Anhang

A1 Fallbeispiel aus der Praxis

A1.1 Unternehmensspezifisches Datenmodell

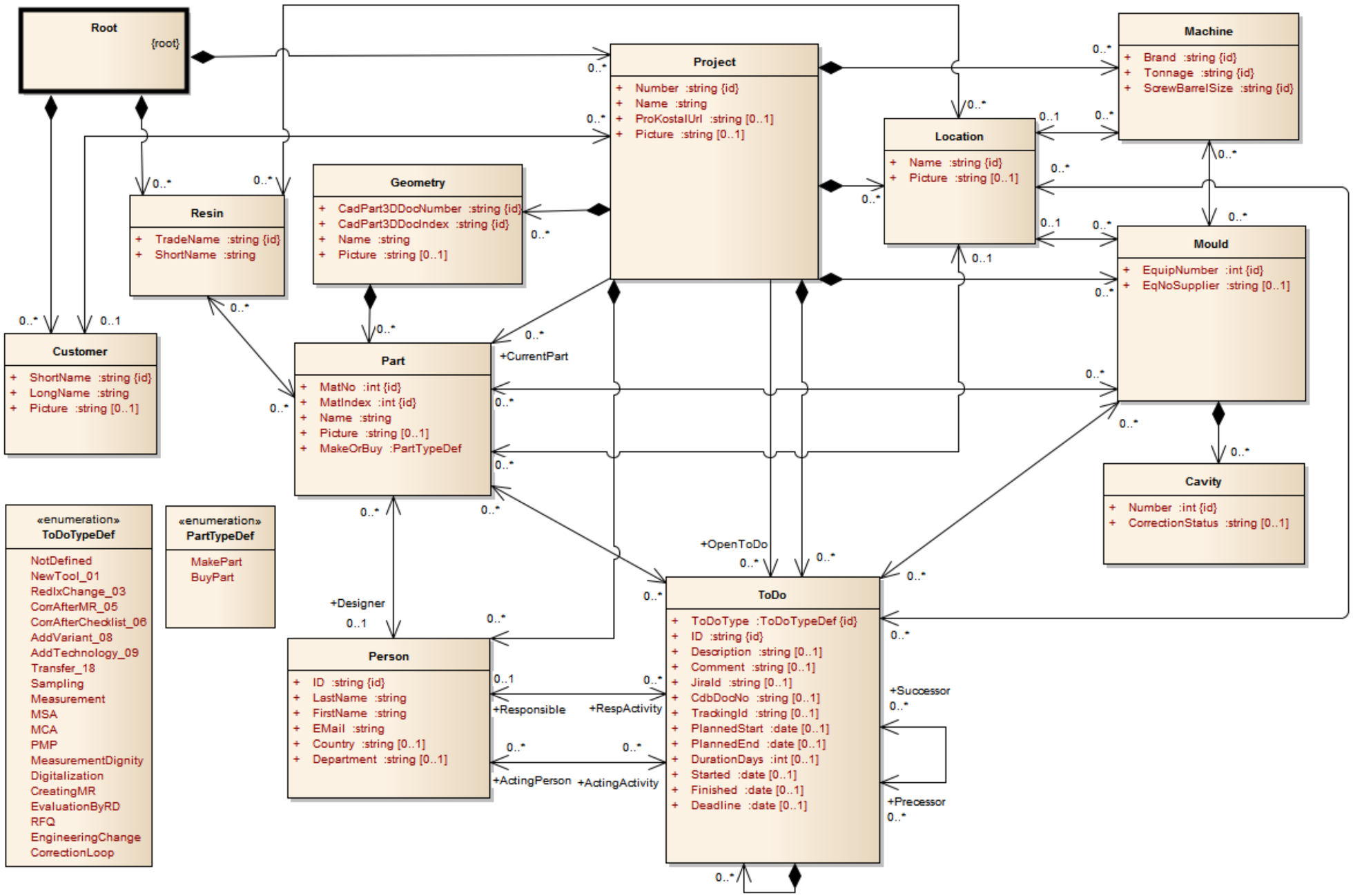


Abbildung 31: Gesamtes unternehmensspezifisches Datenmodell

A2 Erstellung eines Referenzmodells

A2.1 Referenz-Klassenmodell

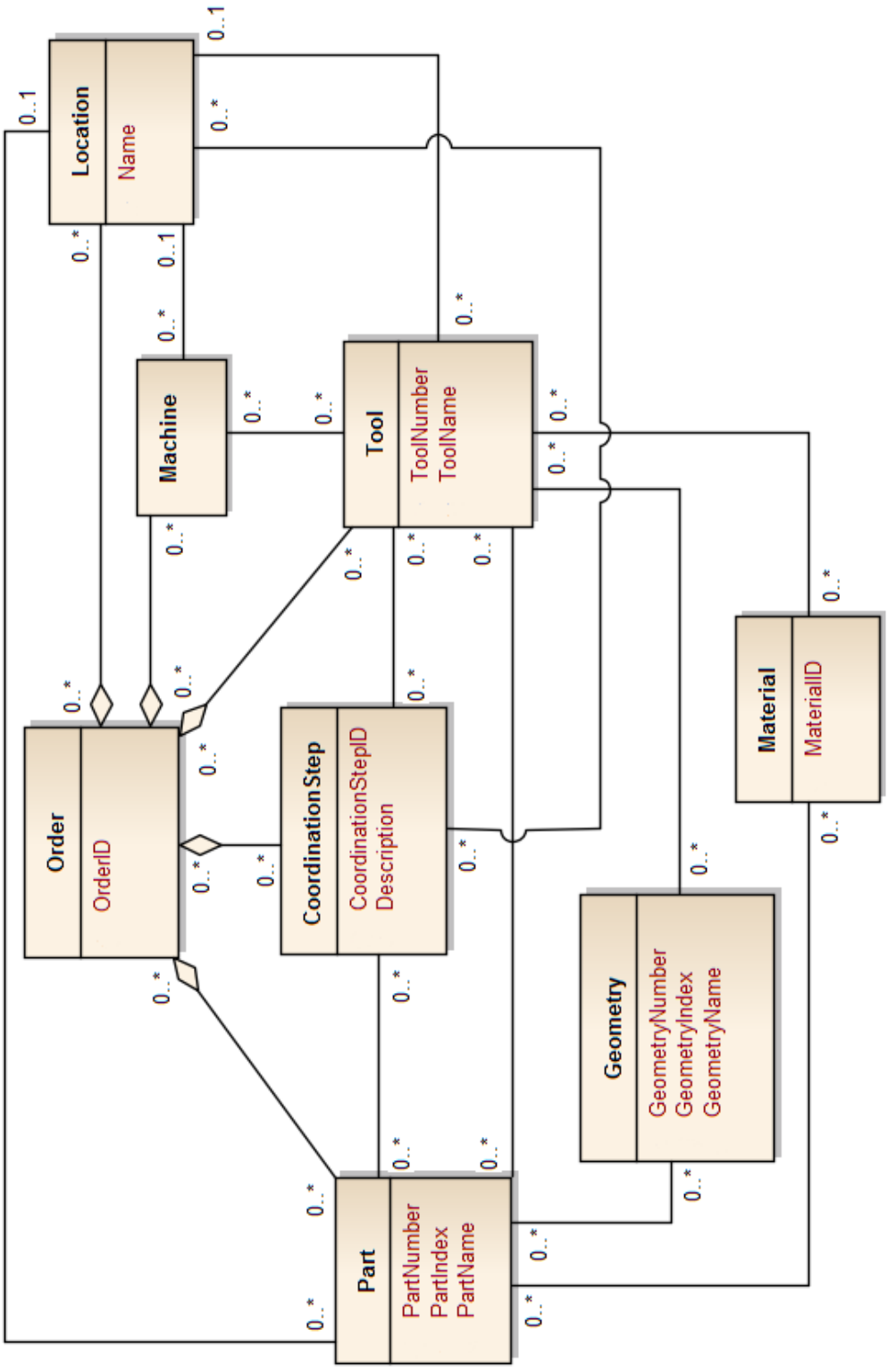


Abbildung 32: Referenz-Klassenmodell

Eidesstattliche Versicherung
(Affidavit)

Rosian, Marvin

Name, Vorname
(Last name, first name)

149966

Matrikelnr.
(Enrollment number)

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

I declare in lieu of oath that I have completed the present Bachelor's/Master's* thesis with the following title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution.

Titel der Bachelor-/Masterarbeit*:
(Title of the Bachelor's/ Master's* thesis):

Intelligentes Datenmanagement in der Produktion - Ein objektorientierter Modellierungsansatz

*Nichtzutreffendes bitte streichen
(Please choose the appropriate)

Dortmund, 02.07.2018

Ort, Datum
(Place, date)

Unterschrift
(Signature)

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:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to €50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, section 63, subsection 5 of the North Rhine-Westphalia Higher Education Act (*Hochschulgesetz*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:**

Dortmund, 02.07.2018

Ort, Datum
(Place, date)

Unterschrift
(Signature)

**Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.