Konzeptuelle Modellierung für modellgetriebene Decision Support Systeme

Konzeption und prototypische Implementierung

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Wirtschaftswissenschaften
des Fachbereichs Wirtschaftswissenschaften
der Universität Osnabrück

vorgelegt
von

Christian Schultewolter

aus
Gronau/Westf.

Osnabrück,
September 2012

Dekan des Fachbereichs:	Prof. Dr. Thomas Gaube
Referenten:	Prof. Dr.-Ing. Bodo Rieger
	Prof. Dr. Frank Teuteberg
Tag der Disputation:	20. Juni 2013

Vorwort

Die vorliegende Dissertation entstand während meiner fünfjährigen Tätigkeit als wissenschaftlicher Mitarbeiter am Fachgebiet BWL / Management Support und Wirtschaftsinformatik des Instituts für Informationsmanagement und Unternehmensführung der Universität Osnabrück. Ihr Gegenstand ist die Entwicklung eines konzeptuellen Modellierungsansatzes für modellgetriebene Decision Support Systeme.

Der Modellierung kommt in modellgetriebenen Decision Support Systemen eine zentrale Rolle zu, da deren Funktionsspektrum maßgeblich durch die Modellkomponente determiniert wird. Aufgrund der Tatsache, dass solche Systeme häufig mittels sogenannter Spreadsheet-Werkzeuge implementiert werden, muss das Problem einer empirisch beobachteten Fehleranfälligkeit von Spreadsheet-Modellen berücksichtigt werden. Im verwandten Themenbereich der Datenmodellierung konnte die Modellqualität durch den Einsatz eines konzeptuellen Modellierungsansatzes nach Chen positiv beeinflusst werden (Entity-Relationship-Model). Die vorliegende Arbeit zielt daher auf die Adaption dieser Vorgehensweise und erarbeitet einen konzeptuellen Modellierungsansatz für modellgetriebene Decision Support Systeme.

Diese Arbeit konnte nur durch die großartige Unterstützung vieler Personen fertiggestellt werden. Ich bin für jegliche Beiträge, Ideen und alle Diskussionen auf dem Weg von der ersten Idee bis zur finalen Fertigstellung sehr dankbar. Zu ganz besonderem Dank fühle ich mich meinem Doktorvater, Herrn Prof. Dr.-Ing. Bodo Rieger, verpflichtet. Kontinuierliche Diskussionen, ständige Bereitschaft zum wissenschaftlichen Diskurs und das Einräumen der nötigen Freiräume für Forschungstätigkeiten waren ein entscheidender Faktor für das Gelingen meines Promotionsvorhabens. Danken möchte ich auch Herrn Prof. Dr. Frank Teuteberg für die Übernahme des Korreferats sowie das Feedback in Doktorandenkolloquien.

Ein besonderer Dank gilt außerdem Daniel Pöppelmann, der mir in unzähligen Diskussionen Ideen und Anregungsvorschläge gab, welche mir bei der Umsetzung und Fertig-

stellung der Arbeit sehr hilfreich waren. Danken möchte ich auch allen weiteren Kolleginnen und Kollegen, insbesondere Dr. Markus Gelhoet, Christopher Harb, Sonja Schulze, Jutta Stelter, Boris Werner, Nick Oldenburger, Jan-Frederik Marx, Dr. Nicolas Meseth, Dr. Patrick Kirchhof und Dr. Stefan Schlangen.

Meiner Freundin Dr. Bettina Gleichauf gilt ein ganz besonderer Dank. In allen Phasen meines Dissertationsprojekts konnte ich auf ihren uneingeschränkten Zuspruch bauen und Verständnis für jegliche Einschränkung erwarten, die eine solche Arbeit mit sich bringt. Ebenfalls möchte ich ihrer Familie für das Interesse an der Arbeit sowie die moralische Unterstützung danken.

Schließlich möchte ich meinen Eltern, Hermann und Elisabeth Schultewolter, sowie meinen Geschwistern André und Daniel Schultewolter danken. Ich bin mir bewusst, dass das Ermöglichen meiner akademischen Laufbahn, ihr Interesse sowie jedwede Unterstützung meines Vorhabens etwas ganz Besonderes waren. Aus diesem Grund sei ihnen allen diese Arbeit gewidmet.

Christian Schultewolter

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADAPT	Application Design for Analytical Processing Technologies
AHP	Analytischer Hierarchie Prozess
AIS	Association for Information Systems
BI	Business Intelligence
DBMS	Datenbankmanagementsystem
DDM	Dialog, Daten und Modellierung
DFM	Dimensional Fact Model
DTD	Document Type Definition
DSS	Decision Support Systeme
DWH	Data Warehouse
EIS	Executive Information Systems
EuSpRIG	European Spreadsheet Risk Interest Group
JDOM	Java Document Object Model
MBMS	Modell-/Methodenbankmanagementsystem
MD	Multidimensional Data Model
MDSS	Modellgetriebene Decision Support Systeme
MIS	Management Information Systems
MML	Multidimensional Modeling Language
MOC	maximum ontological completeness
MOO	minimal ontological overlap
MOD^{2}DSS	Modeling to Decision Support Systems

MSS	Management Support Systems
M/ER Model	Multidimensional Entity-Relationship Model
OFFIS	Oldenburger Forschungs- und Entwicklungsinstitut für Informatik
OLAP	Online Analytical Processing
OOMD	Object oriented Multidimensional Data Model
OR	Operations Research
SDWM	Semantisches Data Warehouse Modell
SIGDSS	Special Interest Group on Decision Support, Knowledge and Data Management Systems
UML	Unified Modeling Language
VB	Visual Basic
VBA	Visual Basic for Applications
XML	eXtensible Markup Language

1 Einleitung

1.1 Motivation und Zielsetzung der Arbeit

George Gorry und Michael Scott Morton ebneten 1971 mit der Kombination unterschiedlicher Management Perspektiven nach Anthony (Anthony, 1965) und dem Strukturiertheitsgrad von Entscheidungssituationen nach Simon (Simon, 1947, 1977) den Weg für die ersten Überlegungen zur Unterstützungsleistung von Computern in Entscheidungssituationen für Manager. Die Differenzierung von strukturierten, semi-strukturierten und unstrukturierten Situationen nimmt in diesem Zusammenhang eine zentrale Position ein. Peter Keen und Michael Scott Morton beschreiben 1978 die Aufgabe von Decision Support Systemen (DSS) als Unterstützung des Entscheidungsträgers in semi-strukturierten Situationen. Der Entscheidungsträger soll ausdrücklich nicht ersetzt, sondern im Prozess der Entscheidungsfindung adäquat unterstützt werden. Ziel eines DSS ist es, die Effektivität und nicht die Effizienz der Entscheidungsfindung positiv zu beeinflussen (vgl. Keen und Scott Morton, 1978, S. 1). Alter wählt 1980 eine sehr ähnliche Definition (vgl. Alter, 1980, S. 1) und auch Bonczek et al. stellen die Unterscheidung des Strukturiertheitsgrads sowie das Ziel den Entscheidungsträger zu unterstützen und ihn nicht zu ersetzen, heraus (vgl. Bonczek et al., 1981, S. 18f.).

Im Laufe der Zeit wurden viele unterschiedliche Konzepte entwickelt, die dem Bereich der DSS zuzuordnen sind. Zur genaueren Abgrenzung der verschiedenen Systeme finden sich in der Literatur zahlreiche Klassifizierungsvorschläge, die unterschiedliche Charakteristika der Systeme für deren Differenzierung heranziehen. Die Klassifizierung der *Special Interest Group on Decision Support, Knowledge and Data Management Systems* (SIGDSS) der *Association for Information Systems* (AIS) (vgl. Turban et al., 2011a, S. 79ff.) sowie der Vorschlag von Holsapple und Whinston (vgl. Holsapple und Whinston, 1996, S. 178ff.) ähneln sich sehr stark und haben sich im wissenschaftlichen Umfeld etabliert. Sie prägen unter anderem die Klasse der modellgetriebenen Decision Support Systeme (MDSS) bzw. der Spreadsheet-orientierten DSS, der in dieser Arbeit eine zentrale Bedeutung zukommt.

Die Modellierung spielt in MDSS eine besonders wichtige Rolle, da das Funktionsspektrum solcher Systeme maßgeblich durch eine Modellkomponente determiniert wird. Bonczek et al. erwähnen bereits Anfang der 1980er Jahre die generelle, hohe Relevanz von Modellen sowie den Nutzen umfangreicher, aber vor allem intuitiv nutzbarer Modellie-

rungssprachen (vgl. Bonczek et al., 1981, S. 18). Im Rahmen dieser Arbeit ist dies von besonderer Bedeutung, da die adressierten Systeme Modelle nutzen, die explizit multidimensionale Strukturen unterstützen. Daraus folgt, dass ein Modell kein flaches Gebilde darstellt, sondern einen mehrdimensionalen Raum aufspannt, in welchem Modellvariablen uneingeschränkt miteinander in Beziehung stehen und so dem Modell eine logische Komponente verleihen. Dies erhöht die Anforderungen an eine intuitive Bedienung der Modellierungssprache.

Spreadsheet-Werkzeuge wie Microsoft Excel (kurz: Excel) waren 1981 noch nicht populär und kaum am Markt vertreten. Sie bilden heute die bekannteste und meist genutzte Entwicklungsumgebung für die Erstellung von MDSS. Excel bietet Schnittstellen zu Datenbanken, unterstützt Lineare Programmierung und stellt dem Anwender zahlreiche statistische und weitere Funktionen zur Verfügung (vgl. Turban et al., 2011a, S. 80). Obgleich der Nutzen solcher Spreadsheet-Werkzeuge unumstritten ist, sieht sich der Benutzer auch mit Problemen konfrontiert. Empirisch wurde eine nicht zu vernachlässigende Fehleranfälligkeit von Spreadsheet-Modellen nachgewiesen. Diese beschränkt die Aussagekraft und Interpretierbarkeit von MDSS, deren Modellkomponente mittels eines Spreadsheet-Werkzeugs erstellt wurde.

Ein sehr ähnliches Problem wurde im Bereich der Datenmodellierung in den 1970er Jahren durch eine konzeptuelle Modellierungsebene gelöst. Der Vorschlag des Entity-Relationship Modells nach Chen (vgl. Chen, 1976) wird auch heute noch genutzt. Durch diese unabhängige Ebene ergeben sich nicht nur positive Effekte im Hinblick auf die Fehleranfälligkeit der erstellten datengetriebenen Systeme, es bietet sich auch die Möglichkeit, konzeptuelle Datenmodelle in proprietäre Modellinstanzen verschiedener Datenbank-Applikationen zu übersetzen. Vom Anwender wird auf diesem Weg weniger Werkzeugspezifisches Wissen gefordert.

Das Gestaltungsziel der vorliegenden Arbeit liegt analog auf der Konzeption und prototypischen Implementierung einer konzeptuellen Modellierungsebene für MDSS. Der Anwender soll in die Lage versetzt werden unter Anwendung des Konzepts ein generisches, multidimensionales Modell zu definieren und dieses automatisiert in ein Spreadsheet-Modell zu überführen. Während des Modellierungsprozesses wird der Anwender mit Informationen bezüglich der Konsistenz der modellierten Zusammenhänge versorgt. Auf diese Weise soll die empirisch beobachtete Fehleranfälligkeit positiv beeinflusst werden,

so dass das resultierende Modell für aussagekräftige Analysen aus dem Bereich der Entscheidungsunterstützung genutzt werden kann.

1.2 Vorgehen und Aufbau

Der ausgeprägte Anwendungsbezug der Arbeit impliziert das Ziel, die Ergebnisse dieser Forschungsarbeit gewinnbringend in der Anwendungspraxis einzusetzen. Diesbezüglich stellt die Konstruktion der konzeptuellen Modellierungsebene ein Artefakt dar, das aus den Teilen *Modellierungssprache, Modell* und einer *prototypischen Implementierung* besteht. Das Erstellen und Evaluieren von Prototypen ist die dominante Forschungsmethodik im Bereich der deutschsprachigen Wirtschaftsinformatik (vgl. Frank, 2006, S. 1). Forschungsmethodisch entspricht dieses Vorgehen dem konstruktionsorientierten Ansatz auf Grundlage von Deduktion. Aus wissenschaftstheoretischer Perspektive bietet sich die Konstruktion eines Artefakts an, sofern Anforderungen spezifiziert werden können und die Lösung eines Problems oder einer Klasse von Problemen in Aussicht stehen (vgl. Frank, 2007, S. 179). Hinsichtlich einer konzeptuellen Modellierungsebene ist dies gegeben.

Ausgehend von der Beobachtung der Fehleranfälligkeit von Spreadsheet-Systemen werden die Anforderungen an eine konzeptuelle Modellierungsebene auf Basis der charakteristischen Modelleigenschaften erarbeitet. Auf Basis dieses explizierten Anforderungskatalogs erfolgt die Entwicklung der prototypischen Anwendung, mittels derer eine Evaluation im Sinne eines *Proof of Concept* durchgeführt wird (vgl. Wilde und Hess, 2006, S. 6). Neue Erkenntnisse können bei dieser Methodik sowohl in der Erstellung des Prototyps als auch durch die Evaluation des Konzepts anhand der Anwendung bzw. des Artefakts gewonnen werden (vgl. Wilde und Hess, 2007, S. 282). Die Evaluation sollte daher nicht als alleinige Explikation der Erkenntnisse betrachtet werden. Eine gesamthafte Betrachtung von Anforderungen, Konzeptualisierung, Prototyp und Evaluation erscheint sinnvoll. Des Weiteren müssen die Limitationen des Prototyps bei der Interpretation der Ergebnisse berücksichtigt werden. Einerseits beeinflusst die Unvollständigkeit der Applikation Evaluationsergebnisse auf negative Art und Weise, andererseits ermöglicht ein Prototyp mit diesen charakteristischen Einschränkungen erst diese Art des Erkenntnisgewinns (vgl. Naumann und Jenkins, 1982, S. 32).

Inhaltlich ist die Arbeit in drei Teile gegliedert. Teil 1 - *Modellierung und modellgetriebene Decision Support Systeme* - befasst sich in den Kapiteln 2 und 3 mit der theoretischen Einordnung und Abgrenzung der Forschungsarbeit. Des Weiteren wird die Problemstel-

lung konkretisiert sowie der State-of-the-Art formuliert. Teil 2 - *Konzept eines multidimensionalen Modellierungsansatzes* - erarbeitet auf Basis von Teil 1 in Kapitel 4 das Konzept einer konzeptuellen Modellierungsebene für MDSS und legt in Kapitel 5 gesonderten Wert auf die Darstellung der konzeptuellen Modellierungssprache. Abschließend wird in Teil 3 - *Prototypische Implementierung und Evaluation des konzeptuellen Modellierungsansatzes* - die prototypische Applikation innerhalb des sechsten Kapitels erläutert und eine Evaluation des Konzepts bzw. des Prototyps in Kapitel 7 vorgenommen.

Im Detail befasst sich Kapitel 2 mit der inhaltlichen Definition und Abgrenzung des zugrunde gelegten Verständnisses des Themenbereichs MDSS. Hierzu werden Klassifizierungsvorschläge für DSS erläutert und die Architekturkomponenten eines typischen DSS beschrieben. Der Terminus *Modellierung* nimmt in der vorliegenden Arbeit eine zentrale Rolle ein. Aus diesem Grund wird eine theoretische Herleitung des vorausgesetzten Modellverständnisses auf Basis des axiomatischen, abbildungsorientierten, konstruktionsorientierten und allgemeinen Modellbegriffs unterbreitet. Abschließend wird aus diesen Grundlagen die Klasse modellgetriebene DSS subsummiert sowie eine Abgrenzung adressierter MDSS vorgenommen.

Das dritte Kapitel beschäftigt sich mit der Definition des Forschungsbedarfs. Konkret werden die Anforderungen konzeptueller Modellierung hinsichtlich multidimensionaler Modelle im Bereich DSS erläutert. Dieser fußt auf einer empirisch beobachteten Fehlerrate in Modellen der betrachteten MDSS und ist eng mit der charakteristischen Multidimensionalität der Modelle verbunden. Nachdem Schwierigkeiten und Herausforderungen der Modellierung in diesem Umfeld erläutert wurden, werden bekannte Lösungsansätze und Methoden aus angrenzenden Themengebieten auf die Möglichkeit der Adaption für die vorliegende Forschungsarbeit untersucht.

Basierend auf dem Forschungsbedarf sowie der theoretischen Abgrenzung wird in Kapitel 4 eine konzeptuelle Modellierungsebene als Lösungsvorschlag entworfen. Ein Überblick und eine Erläuterung der Architekturkomponenten verdeutlichen zu Beginn den Bezugsrahmen des Konzepts. Im Anschluss werden mögliche, charakteristische Modellierungsfälle eingehend betrachtet. Es wird eine Klassifizierung vorgeschlagen, mittels derer die Modellierung im multidimensionalen Kontext hinsichtlich der Kriterien *Grad*, *Homogenität* und *Simultanität* der beteiligten Dimensionen in einer Formel unterschieden werden kann. Mithilfe dieser Differenzierung lassen sich Informationen generieren, die dem Anwender im Modellierungsprozess unterstützend zur Verfügung gestellt werden können.

Nachdem das Konzept im Gesamten dargestellt wurde, präsentiert Kapitel 5 den Entwurf der konzeptuellen Modellierungssprache, die im Gesamtkonzept eine Grammatik für die modellierten Inhalte vorgibt. Durch die Sprachkonzepte werden die Konstrukte und Elemente definiert, aus denen Instanzen der konzeptuellen Modelle aufgebaut sind. Anhand dieser Elemente und deren Eigenschaften müssen im Folgenden sämtliche Informationen abgebildet werden können, die den Inhalt, Aufbau und die Logik eines Modells determinieren.

Das sechste Kapitel widmet sich der prototypischen Realisation des entwickelten Konzepts. Nach einem kurzen Einblick in die technischen Gegebenheiten der Entwicklungsumgebung werden die Komponenten des Prototyps vorgestellt. Der Gliederung des Konzepts entsprechend besteht der Prototyp ebenfalls aus mehreren Teilen. Den Erläuterungen zur Umsetzung des Modellierungseditors für die Definition grundlegender Modellstrukturen folgt die Darstellung einer Modelltransformationskomponente. Diese übernimmt die Aufgabe, Überschneidungen von Formeln innerhalb von Zellen eines Modells zu entfernen und somit eine eindeutig definierte Modellinstanz zur Verfügung zu stellen. abschließend werden die Limitationen des Prototyps erläutert. Diese fallen gegenüber dem detaillierten Konzept in der Regel sehr umfangreich aus, da die Applikation im Sinne eines Proof of Concept auf viele Details verzichtet, die sich sehr Entwicklungs-intensiv gestalten würden, ohne den Zweck des Nachweises der Nützlichkeit nennenswert zu befördern.

Kapitel 7 stellt den inhaltlichen Abschluss dieser Arbeit in Form einer Evaluation des Konzepts dar. Anhand eines Modellierungsvergleichs zwischen dem Prototyp und Excel erfolgt eine Beurteilung des erarbeiteten Konzepts. Der Vergleich basiert auf einer Fehlermetrik. Diese begründet für die Analyse und Auswertung der Ergebnisse, welche Konstellationen als Fehler gewertet werden und geht dabei auf Besonderheiten bezüglich der genutzten Modellierungswerkzeuge ein.

Abschließend fasst Kapitel 8 die Arbeit kurz zusammen, unterbreitet eine kritische Würdigung des vorgeschlagenen Konzepts und gibt einen Ausblick über weitere Forschungsaktivitäten im Umfeld dieser Arbeit.

Teil I

Modellierung und modellgetriebene Decision Support Systeme

2 Modellgetriebene Decision Support Systeme

Decision Support bzw. Entscheidungsunterstützung beschreibt einen Bereich, über welchen in der Literatur keine feststehende, allgemein akzeptierte Definition zu finden ist. Dieses Kapitel widmet sich der Erklärung und Abgrenzung alternativer Begriffsverständnisse und geht dabei insbesondere auf den Teilbereich *Modellgetriebene Decision Support Systeme* ein, welche einen zentralen Aspekt dieser Arbeit darstellen. Des Weiteren wird der Begriff *Modellierung* in diesem Kontext definiert sowie eine Übersicht unterschiedlicher Arten von Modellen gegeben, die im Umfeld von MDSS relevant sind. Abschließend wird das für diese Arbeit relevante Verständnis von MDSS abgeleitet.

2.1 Decision Support Systeme

2.1.1 Definition

Diese Arbeit folgt der allgemein anerkannten Einordnung von Decision Support Systemen als Teilbereich von Management Support Systemen (MSS) bzw. Managementunterstützungssystemen. MSS etablierte sich als Sammelbegriff für Informations- und Kommunikationssysteme in den 1980er Jahren (vgl. Holten, 1999, S. 29; Kemper et al., 2006, S. 1). Scott Morton spielte bei der Entstehung des Begriffs eine zentrale Rolle und definierte ihn als „the use of information technologies to support management“ (Scott Morton, 1983, S. 2). Seit Mitte der 1990er Jahre entwickelte sich aus der unternehmerischen Praxis heraus mit *Business Intelligence* (BI) ein neuer Begriff. Im Laufe der Zeit hat der Begriff auch in der Wissenschaft Anklang gefunden und wird häufig synonym zu MSS verwendet. Klassische Ausprägungen von BI bzw. MSS sind neben DSS Management Information Systems (MIS) und Executive Information Systems (EIS) (vgl. Gluchowski et al., 2008, S. 15; Turban et al., 2011a, S. 19; Winter, 2010, S. 95).

Alter beschreibt 1996 Decision Support Systeme als interaktive Systeme, welche den Benutzer beim Entscheiden und Beurteilen unterstützen, wenn über das Einsatzgebiet wenig bekannt ist und keine oder wenige Erfahrungswerte vorliegen (vgl. Alter, 1996, S. 225). DSS entfalten ihr Potenzial insbesondere in semi- und unstrukturierten Entscheidungssituationen und generieren hauptsächlich in der Such- bzw. Auswahlphase eines Entscheidungsprozesses durch das Bereitstellen von Modellen, Methoden und problembezogenen Daten einen Nutzen für den Entscheidungsträger (vgl. Bonczek et al., 1981, S. 18; Keen

und Scott Morton, 1978, S. 1; Sprague und Carlson, 1982, S. 4f.; Turban et al., 2011a, S. 45ff.). Die Unterscheidung von unstrukturierten, semi-strukturierten und strukturierten Entscheidungssituationen sowie die Unterscheidung einzelner Phasen einer Entscheidung gehen dabei auf Simon zurück, der Überlegungen bezüglich der Automatisierbarkeit von Entscheidungen formuliert (vgl. Simon, 1977, S. 40ff.). Er unterscheidet vier Phasen eines Entscheidungsprozesses: *Intelligence*, die Suche nach Problemsituationen, die eine Entscheidung erfordern, *Design*, die Entwicklung von alternativen Vorgehensweisen, *Choice*, die Auswahl einer Alternative und *Review/Implementation*, die Umsetzung der Entscheidung und der damit verbundenen Handlungen (vgl. Simon, 1977, S. 40ff.; Turban et al., 2011a, S. 12). Er definiert strukturierte Entscheidungen (*programed decisions*) als routinemäßig wiederholte Entscheidungen, denen ein konkreter Maßnahmenkatalog bezüglich durchzuführender Handlungen zu Grunde liegt. Unstrukturierte Entscheidungen verlangen hingegen eine individuelle Vorgehensweise, weil bisher keine Methode bekannt ist oder die Umgebungsvariablen so komplex und wenig präzise abzubilden sind, dass eine Methode zur Handhabung nicht in Frage kommt (vgl. Simon, 1977, S. 46f.).

Gorry und Scott Morton entwickelten darauf aufbauend ein Framework für die Computer-basierte Entscheidungsunterstützung, das eine Basis für die Entwicklung des allgemeinen DSS-Konzepts darstellt (vgl. Turban et al., 2011a, S. 11). Das Rahmenwerk stellt die Klassifikation der drei Managementperspektiven *strategic planning*, *management control* und *operational control* nach Anthony dem Strukturiertheitsgrad von Entscheidungen nach Simon gegenüber (vgl. Anthony, 1965, S. 15ff.; Gorry und Scott Morton, 1971, S. 56ff.; Turban et al., 2011a, S. 11ff.) [1]. *Strategic planning* umfasst langfristige Aktivitäten in einem Unternehmen, wie z.B. Ressourcen-Allokation und den Entwurf von Richtlinien, unter *management control* fällt die Akquise und effiziente Nutzung von Ressourcen zur Umsetzung der langfristigen Unternehmensziele und *operational control* umfasst die effiziente und effektive Ausführung spezieller Arbeitsprozesse auf unterster Unternehmensebene (Anthony, 1965, S. 15ff.; Turban et al., 2011a, S. 12f.).

[1] Einzelheiten zum Framework und dem Entwicklungsprozess finden sich in (Gorry und Scott Morton, 1971)

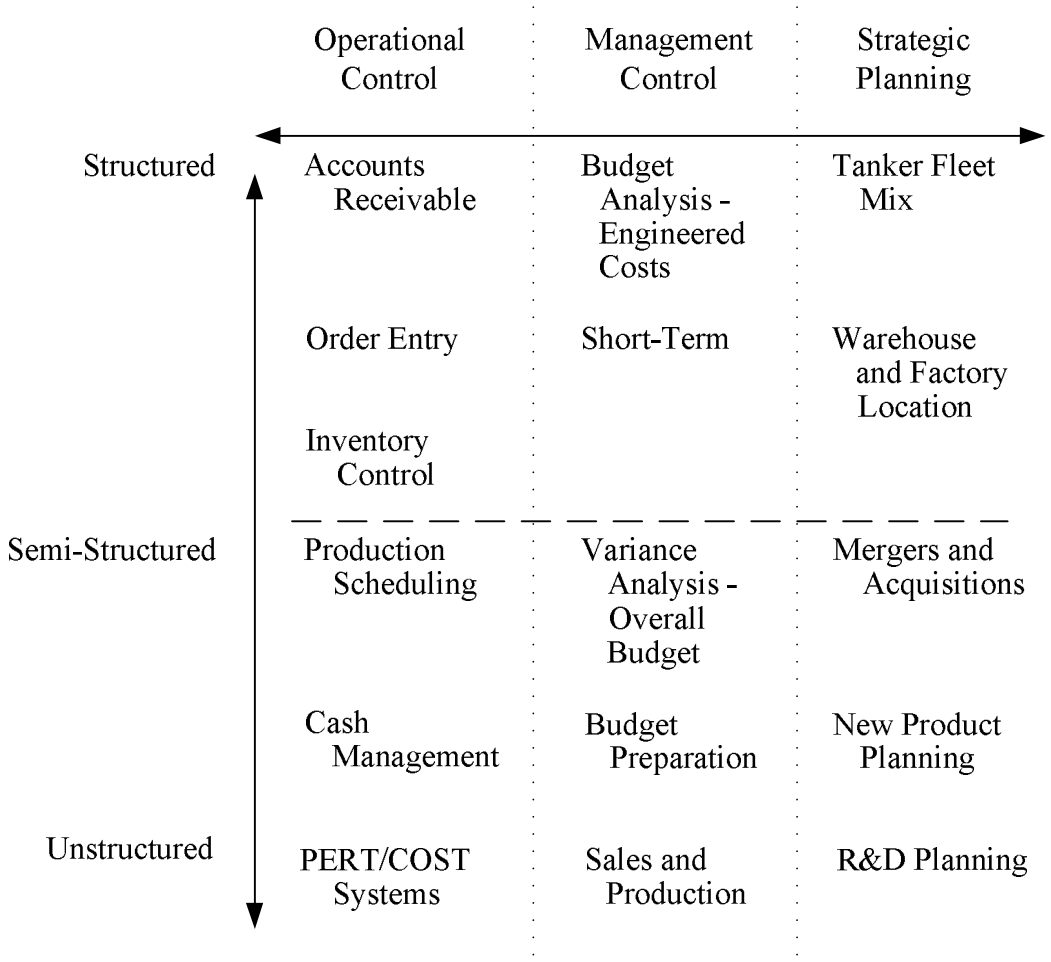


Abbildung 2-1: Framework nach Gorry und Scott Morton

(Gorry und Scott Morton, 1971, S. 62)

Abbildung 2-1 zeigt das Framework nach Gorry und Scott Morton. Das Ziel von Decision Support Systemen ist vor allem die Unterstützung des Entscheiders in semi- und unstrukturierten Situationen (vgl. Gorry und Scott Morton, 1971, S. 61; Keen und Scott Morton, 1978, S. 1), auf den Ebenen management control und strategic planning (vgl. Gorry und Scott Morton, 1971, S. 65). Entscheidungssituationen in diesem Bereich zeichnen sich dadurch aus, dass der Informationsbedarf des Entscheiders nicht objektiv bestimmt werden kann. Intuition sowie Urteilsvermögen des Entscheiders spielen eine wichtige Rolle in solchen Situationen (vgl. Turban et al., 2011a, S. 15). Der Entscheidungsträger versucht eine geeignete Lösungsstrategie für ein Problem zu finden, die eine sinnvolle Entscheidung nach sich zieht. Je mehr situativ verwertbare Informationen dem Entscheidungsträger zur Verfügung gestellt werden können, desto positiver kann man dessen Fähigkeit zur objektiven Beurteilung von Entscheidungssituationen beeinflussen. Kernaufgabe eines DSS ist daher die Steigerung der Effektivität von Entscheidungsfindungen (vgl. Alter, 1980, S. 95ff.; Keen und Scott Morton, 1978, S. 7ff.). Eine Steigerung der Effizienz würde bedeu-

ten, dass eine Entscheidung so gut wie möglich hinsichtlich eines zuvor definierten Kriteriums getroffen wird (vgl. Keen und Scott Morton, 1978, S. 7).

Das Ziel, dem Benutzer *verwertbare* Informationen bereitzustellen, wird in DSS durch Anwendung unterschiedlicher Modelle und Methoden erreicht. Dies wiederum schlägt sich in unterschiedlichen Klassifizierungen von DSS nieder, denen sich der nächste Abschnitt widmet.

2.1.2 Klassifizierungen von DSS

In der Literatur werden unterschiedlichste Kriterien zur Klassifizierung von DSS herangezogen. Hackathorn und Keen beispielsweise differenzieren DSS nach den vorgesehenen Benutzern (vgl. Hackathorn und Keen, 1981), während Donovan und Madnick nach der wiederkehrenden Verwendung eines DSS lediglich ad-hoc und institutionalisierte DSS unterscheiden (vgl. Donovan und Madnick, 1977, S. 81). Im Folgenden werden vier Ansätze vorgestellt. Diese sind im Rahmen dieser Arbeit relevant, da sie explizit Modelle berücksichtigen.

2.1.2.1 Klassifizierung nach SIGDSS

Die *Special Interest Group on Decision Support, Knowledge and Data Management Systems* (SIGDSS) der *Association for Information Systems* differenziert DSS in kommunikations-, daten-, dokumenten-, wissens- und modellgetriebene Systeme. Des Weiteren gibt es eine Klasse *compound*, die Systeme umfasst, welche Eigenschaften von mindestens zwei der erwähnten Klassen in sich vereint. Tabelle 1 gibt einen Überblick über Unterstützungsleistungen der einzelnen Klassen:

Tabelle 1: DSS-Klassifizierung nach SIGDSS (vgl. Turban u. a., 2011, S. 79ff.)

Systemklasse	**Unterstützungsgebiete / Methoden**
Kommunikationsgetriebene DSS	- jegliche Systeme, die Unterstützung von Besprechungen, Kommunikation, Kollaboration oder anderer Gruppentätigkeiten bieten
Datengetriebene DSS	- On-Line Analytical Processing (OLAP) - Data Mining - Data Warehouse (DWH) / Large Databases - teilweise mathematische Modelle
Dokumentengetriebene DSS	- Wissensmanagementsysteme - teilweise mathematische Modelle
Wissensgetriebene DSS	- Wissensmanagementsysteme - sämtliche Methoden der Künstlichen Intelligenz - Experten-Systeme
Modellgetriebene DSS	- mathematische Modelle - Optimierungsrechnungen - Simulation - Operations Research (OR)
Compound	- je nach Konfiguration Teilbereiche aller anderen Klassen

Diese Klassifizierung wird im weiteren Verlauf der Arbeit zugrunde gelegt, da sie die fokussierte Klasse MDSS explizit enthält. Für DSS, die aus hybriden Komponenten bestehen und daher nur schwer eindeutig einer Klasse zugewiesen werden können, ist mit *Compound* eine eigene Klasse enthalten.

Die folgenden Ansätze adressieren ebenfalls Modelle und können einen Beitrag für das Verständnis der Klasse MDSS leisten.

2.1.2.2 Klassifizierung nach Alter

Alter schlägt basierend auf einer Analyse von 56 Systemen sieben Klassen von DSS vor (vgl. Alter, 1980, S. 74ff.). Entscheidendes Kriterium für die Klassifizierung ist die allgemeine bzw. generische Aufgabe, die ein System erledigt. Es besteht keine Abhängigkeit der Klassen zum Strukturiertheitsgrad einer Entscheidung nach Simon, dem funktionalen Umfeld, in welchem die Entscheidung zu treffen ist (Abteilung eines Unternehmens, z.B. Marketing oder Produktion) oder der Managementperspektive nach Anthony, der die Entscheidung zugeordnet werden kann. In diesem Zusammenhang unterscheidet Alter nicht zwischen den Termini *System* und *Modell*, welches grundsätzlich nur einen Teil eines DSS darstellt. *File drawer systems* ermöglichen einen unmittelbaren Zugriff auf Daten, *data analysis systems* unterstützen die Datenmanipulation, *analysis information systems* ermöglichen Zugriff auf kleine Modelle sowie Datenbanken, *accounting models* berechnen die Konsequenzen möglicher Aktionen auf Basis belastbarer Daten (z.B. Bilanzierungsdaten), *representation models* schätzen die Konsequenzen auf Basis vermuteter Zusammenhänge, *optimization models* unterbreiten Handlungsalternativen, die hinsichtlich formalisierter Restriktionen ein optimales Ergebnis nach sich ziehen und *suggestion models* schlagen Handlungsalternativen vor, die in strukturierten Situationen möglichst optimale Ergebnisse versprechen lassen (vgl. Alter, 1980, S. 74ff.).

Alters Klassifizierungsvorschlag ist insbesondere für die genauere Abgrenzung von MDSS interessant, da unterschiedliche Modelltypen charakterisiert werden. Dieses Kriterium ist für ein Verständnis der Klasse MDSS interessant und in dieser Form in keinem weiteren Klassifizierungsvorschlag zu finden.

2.1.2.3 Klassifizierung nach Power

Power führt 2002 an, dass Alters Klassifizierung nicht mehr zeitgemäß sei, da DSS im Laufe der Zeit für immer speziellere Anforderungen entwickelt werden und facettenreicher sind. Er identifiziert die von Alter entwickelten Klassen als nach wie vor relevant, schlägt jedoch aus Gründen der Übersicht eine Zusammenfassung in die drei Klassen *datengetriebene*, *modellgetriebene* und *wissensgetriebene* DSS vor. Diesen drei Klassen fügt er fünf weitere Klassen mit aktuellem Bezug hinzu (vgl. Power, 2002, S. 12ff.). *Dokumentengetriebene* DSS unterstützen einen Entscheidungsträger beim Organisieren von Dokumenten jeglicher Art. Eine weitere Klasse identifiziert Power als *Kommunikationsgetriebene und Gruppen* DSS. Systeme dieser Klasse unterstützen Entscheidungsträger bei der Arbeit innerhalb eines Teams und liefern diesbezüglich z.B. Funktionen für die Organisation von

Meetings. *Intraorganisationale* DSS werden ausschließlich innerhalb eines Unternehmens genutzt, während *interorganisationale* DSS z.B. Stakeholdern Zugang zum Unternehmensinternen Intranet ermöglichen, um dort benötigte Informationen abzurufen. Des Weiteren schlägt er vor, DSS nach Art der Verwendung zu unterscheiden. *Funktions-* bzw. *Branchen-spezifische* DSS werden für Aufgaben entwickelt, die wiederholt und weitestgehend standardisiert zu lösen sind, z.B. die Planung von Flugbegleitern bei einer Fluggesellschaft. Demgegenüber stehen DSS, die für einen allgemeinen Zweck entwickelt werden, wie z.B. DSS für das Projektmanagement. Das letzte Merkmal bezieht sich auf die Technologie, mittels derer ein DSS umgesetzt ist und lautet *Web-basierte* DSS. In diesen Systemen werden Internet und andere Webtechnologien bei der Implementierung genutzt.

Powers Klassifizierungsvorschlag berücksichtigt explizit den Technologieaspekt. Unter der Berücksichtigung, dass DSS in den 1980er und Anfang der 1990er Jahre aufgrund beschränkter technologischer Innovationen keine kollaborativen und kommunikativen Funktionen umfassten und sich auch die technische Umsetzung im Laufe der Zeit durch neue Technologien geändert hat, ist dieser Aspekt für die Beurteilung aktueller Systeme relevant. Die fünf von Power hinzugefügten Differenzierungsmerkmale ermöglichen daher eine bessere Abgrenzung der drei Klassen datengetriebene, modellgetriebene und wissensgetriebene DSS.

2.1.2.4 Klassifizierung nach Holsapple und Whinston

Die Klassifizierung nach Holsapple und Whinston (vgl. Holsapple und Whinston, 1996, S. 178ff.) ähnelt der Klassifizierung der SIGDSS sehr stark und besteht ebenfalls aus sechs Klassen (inklusive *compound*) (vgl. Turban et al., 2011a, S. 82). Die Klasse *Text-orientierte* DSS kann *dokumentengetriebenen* DSS nach SIGDSS zugeordnet werden, *Datenbank-orientierte* DSS entsprechen *datengetriebenen* und *Solver-orientierte* DSS entsprechen *modellgetriebenen* DSS. Des Weiteren sind *Spreadsheet-orientierte* DSS eine Form *modellgetriebener* DSS, jedoch mit einem Unterschied: Aufgrund der Tatsache, dass Spreadsheet-Werkzeuge wie z.B. Excel häufig in der Lage sind kleinere Datenbankapplikationen zu nutzen bzw. zu verwalten, kann mit ihnen beschreibendes Wissen, z.B. in Form von Meta-Daten[2], manipuliert werden (vgl. Turban et al., 2011a, S. 82). Da eine solche Funktion nach SIGDSS unter *datengetriebene* DSS fällt, wird hier keine Zuordnung zur Klasse *modellgetriebene* DSS vorgenommen, sondern eine eigene Klasse berücksich-

[2] Meta-Daten stellen eine Beschreibung hinsichtlich Bedeutung und Struktur von Daten dar. Sie geben des Weiteren Auskunft über das Entstehen von Daten, wie Daten zugegriffen und wie sie genutzt werden. *Meta* gibt in diesem Zusammenhang an, dass es sich um „Daten über Daten“ handelt (vgl. Devlin, 1997, S. 52).

tigt. *Regel-orientierte* DSS bilden die fünfte Klasse. Sie schließen die meisten wissensgetriebenen DSS ein, jedoch ergänzt um Data-Mining und Experten-Systeme. *Compound* DSS entsprechen wiederum der gleichnamigen Klasse nach AIS SIGDSS.

Aufgrund der großen Ähnlichkeit zur Klassifizierung nach SIGDSS wird der Vorschlag nach Holsapple und Whinston in großen Teilen analog berücksichtigt. Die Abgrenzung der Klasse Spreadsheet-orientierte DSS ist im Rahmen der Arbeit besonders interessant, da Spreadsheet-Werkzeuge häufig zur Umsetzung der adressierten MDSS herangezogen werden. Die Berücksichtigung von Regel-orientierten Systemen ist ebenfalls interessant, da diese eine genauere Charakterisierung von Formeln und Verknüpfungen in den adressierten MDSS ermöglichen.

Der Begriff *modellgetrieben* im Term MDSS impliziert, dass Modellen in solchen Systemen eine zentrale Rolle zukommt. Um eine konkrete Vorstellung der Komponenten eines DSS sowie deren Zusammenspiel zu erhalten, stellt der nächste Abschnitt die Architektur und den strukturellen Aufbau eines DSS dar.

2.1.3 Komponenten eines DSS

Während es viele unterschiedliche Klassifizierungen von DSS gibt, herrscht in Wissenschaft und Praxis weitestgehend Einigkeit über die Komponenten und die Architektur eines solchen Systems. Sprague und Carlson stellen fest, dass jedes DSS ein Technologiespektrum der drei Gebiete *Dialog*, *Daten* und *Modellierung* abdecken muss (DDM Paradigma) (vgl. Sprague und Carlson, 1982, S. 23ff.). In unternehmensweiten DSS-Applikationen sind die Ressourcen aus den drei Gebieten idealerweise etwa gleichwertig berücksichtigt, weil somit die Anforderung an eine möglichst einfache Bedienbarkeit, die Möglichkeit viele Daten zu berücksichtigen und ein möglichst großes Analyse- sowie Modellierungsspektrum erreicht werden (vgl. Sprague und Watson, 1996, S. 13). Von einem Entscheidungsträger wird dadurch kein oder nur sehr wenig technisches Wissen verlangt. Ein DSS besteht demzufolge aus drei Komponenten, einem *Dialogsystem*, einer *Modell-* bzw. *Methodenbank* sowie einer *Datenbank*. Abbildung 2-2 stellt die Komponenten eines DSS schematisch dar:

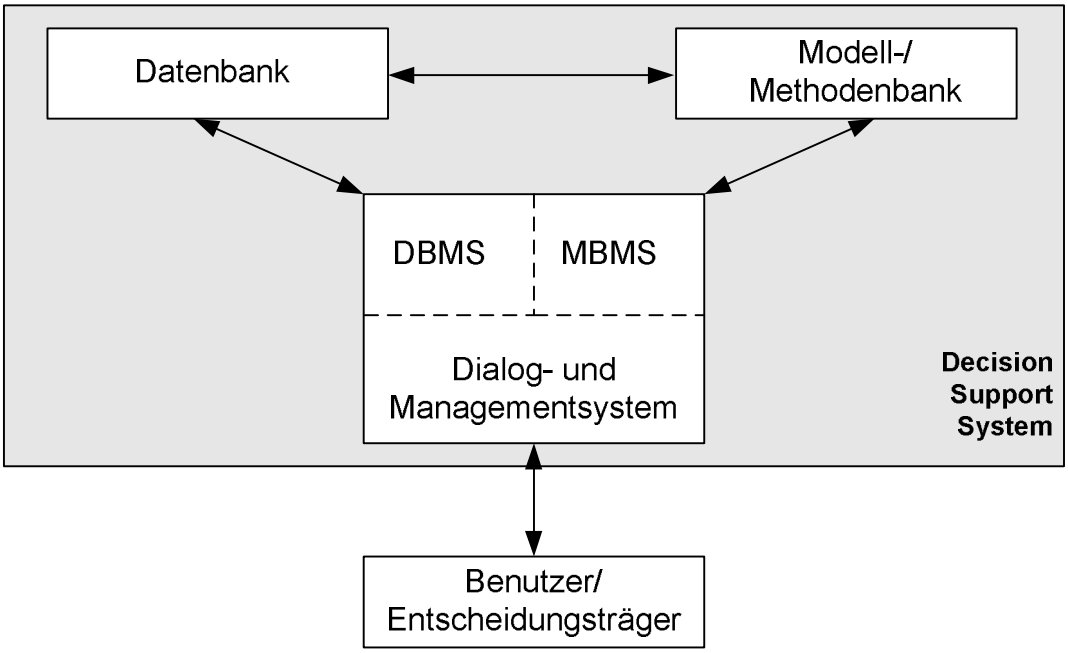


Abbildung 2-2: Komponenten eines DSS

(vgl. Sprague und Carlson, 1982, S. 29)

Die Datenbank übernimmt die Aufgabe, relevante Daten in Bezug auf eine konkrete Entscheidungssituation zur Verfügung zu stellen. Die Datenbasis, die häufig in Form eines Data Warehouse vorliegt (vgl. Sprague und Watson, 1993, S. 104ff., 1996, S. 108ff.), muss nicht die volle Bandbreite operativer Datenquellen umfassen, sondern muss vielmehr in der Lage sein, einen relevanten Datenbestand in Bezug auf eine explizite Entscheidungssituation derart integriert darzustellen, dass er kompatibel zum genutzten Modell und dessen Parametern ist. Dies umfasst häufig persönliche Daten von Entscheidungsträgern, manuell zu erfassende Planungsdaten als Resultat vorhergehender Modellrechnungen und andere externe Quellen (vgl. Gluchowski et al., 2008, S. 70). Des Weiteren entscheidet eine sorgfältige Dokumentation der Datenstruktur darüber, ob der Benutzer die vorliegende Datenbasis adäquat beurteilen und nötige Änderungen spezifizieren kann (vgl. Sprague und Carlson, 1982, S. 32).

Die Modell- bzw. Methodenbank übernimmt die Aufgabe der Erstellung, Speicherung, Katalogisierung sowie die Bereitstellung der unter Umständen zahlreichen verschiedenen Modelle und Methoden (vgl. Gluchowski et al., 2008, S. 68). Häufig werden Entscheidungssituationen in Teilprobleme unterteilt und diese Teilprobleme wiederum durch einzelne Modelle repräsentiert (sog. *building blocks*), die miteinander interagieren (vgl. Sprague und Carlson, 1982, S. 32f.). Je nach Charakteristik der behandelten Entschei-

dungssituationen kann eine Modellbank unterschiedliche Modelltypen enthalten (vgl. Sprague und Watson, 1996, S. 112).

Um ein Modell zur Beantwortung einer Fragestellung bzw. zur Unterstützung in einer Entscheidungssituation nutzen zu können, existieren in einer Methodenbank unterschiedliche algorithmische Verfahren (vgl. Kemper et al., 2006, S. 103). Da Methoden in Form von Algorithmen und Modellen sehr oft eng miteinander verbunden sind und auf Systemebene nicht scharf abgegrenzt werden können, werden sie oft nicht getrennt behandelt, sondern als zusammengehörig aufgefasst (vgl. Gluchowski et al., 2008, S. 67) (siehe Abbildung 2-2). Abhängig vom Anwendungsbereich eines DSS werden von einfachen Methoden der Konsolidierungs- und Aggregationsrechnung bis hin zu komplexen Finanzrechnungen, Korrelations-, Regressions- und Zeitreihenanalysen unterschiedlichste Verfahren eingesetzt (vgl. Gluchowski et al., 2008, S. 69f.). Die bekanntesten Analyseformen sind *What-If-*, *Sensitivitäts-* und *Goal-Seeking-*Analysen. Bei einer What-If Analyse werden die Werte von Modellvariablen, einzelne Parameter oder ganze Formeln variiert und die Auswirkungen auf eine abhängige Zielgröße beobachtet (vgl. Power, 2002, S. 160; Turban et al., 2011a, S. 159). Für diese Form von Analyse stellen DSS idealerweise entsprechende Funktionen zur Beeinflussung der jeweiligen Werte und zum Vergleich der unterschiedlichen Szenarien oder Versionen bereit. Die Sensitivitätsanalyse ähnelt der What-If Analyse sehr stark und wird oft auch unter dieser subsumiert (vgl. Rieger, 1994, S. 128). In diesem Fall stehen jedoch nicht resultierende Wertänderungen abhängiger Modellvariablen im Vordergrund, sondern der Grad der Beeinflussung, die eine Wertänderung einer Modellvariable auf eine abhängige andere Variable ausübt (vgl. Turban et al., 2011a, S. 158f., 2011b, S. 154). Während sowohl bei einer What-If, als auch bei einer Sensitivitätsanalyse stets vorwärts orientiert analysiert wird, d.h. auf Änderungen von Variablenwerten folgen Änderungen beobachteter Zielgrößen, wird dieser Zusammenhang in einer Goal-Seeking-Analyse (auch how-to-achieve Rechnung genannt) umgekehrt. Diese wird genutzt, um geeignete Maßnahmen zur Erreichung eines definierten, quantitativen Unternehmensziels zu eruieren (vgl. Mertens, 2003, S. 62). Eine Break-Even Rechnung ist ein typisches Beispiel einer Goal-Seeking-Analyse (vgl. Sharda et al., 1988, S. 149). Leistungsfähige DSS offerieren dem Benutzer bei einer Abhängigkeit der beobachteten Zielvariablen (engl.: goal variable) von mehreren anderen Modellvariablen mathematische Verteilungen anzuwenden, um den Grad der Beteiligung beeinflussender Modellvariablen beeinflussen zu können. Des Weiteren ermöglichen sie das Sperren von Variablen, so dass sie konstante Werte annehmen und im Berechnungsprozess nicht variiert werden können.

Modell-/Methodenbank und Datenbank verfügen über keinerlei Elemente, mittels derer eine Interaktion mit dem Benutzer ermöglicht werden könnte. Die Präsentation jeglicher Inhalte und die Verwaltung aller Benutzereingaben zur Bedienung des DSS erfolgt daher über ein Dialogsystem. Diese Komponente muss so gestaltet sein, dass der Anwender die angebotenen Funktionen möglichst intuitiv verstehen und aufrufen kann. Bei der Bedienung sollte des Weiteren auf starre hierarchische Befehlsketten durch entsprechende Menüabfolgen verzichtet werden, da das menschliche Problemlösungsverhalten oft sprunghaft und wenig strukturiert ist (vgl. Gluchowski et al., 2008, S. 68). Um in diesem unstrukturierten Prozess Plausibilität und Sinnhaftigkeit der Benutzereingaben sicherstellen zu können, übernimmt das Dialogsystem diese Überprüfung sowie die inhaltliche Anleitung des Benutzers durch kontextsensitive Hilfefunktionen und Warnhinweise beim Feststellen von Fehlern im Bedienungsprozess (vgl. Gluchowski et al., 2008, S. 68; Kemper et al., 2006, S. 104). Neben diesen Anforderungen muss ein Dialogsystem ein umfassendes Funktionsspektrum zur Administration von Daten- und Modell-/Methodenbank bereitstellen. In Abbildung 2-2 sind daher innerhalb des Dialogsystems die beiden Komponenten DBMS (Datenbankmanagementsystem) und MBMS (Modell-/Methodenbankmanagementsystem) gesondert berücksichtigt. Das MBMS muss in diesem Zusammenhang mit der Bereitstellung einer Modellierungsschnittstelle eine zentrale Funktion im gesamten DSS zur Verfügung stellen. Über eine möglichst eindeutige, logische Modellierungssprache werden Modelle erstellt, gepflegt und überarbeitet (vgl. Sprague und Carlson, 1982, S. 33). Das Spektrum angebotener Operatoren für die Modellierung logischer Zusammenhänge innerhalb eines Modells determiniert die Mächtigkeit der Modellierungssprache.

Neben den geschilderten Aufgaben übernimmt das Dialogsystem (siehe Abbildung 2-2: Dialog- und Managementsystem) aber auch Koordinationsaufgaben zwischen Datenbank und Modell-/Methodenbank, die miteinander interagieren. Einzelne building blocks innerhalb der Modellbank sind häufig zur Umsetzung von Teilproblemen einer übergeordneten Entscheidungssituation zuständig und müssen zur ganzheitlichen Betrachtung miteinander in Beziehung gesetzt werden. Die Korrektheit dieser Inter-Modell-Beziehungen gewährleistet das MBMS, indem es sämtlichen Modellen eine einheitliche Datenbasis anbietet. Dies stellt Konsistenz sicher und bietet gleichzeitig die Möglichkeit, einzelne Modelle über das Nutzen derselben Datenobjekte unkompliziert miteinander zu verbinden (vgl. Sprague und Carlson, 1982, S. 256ff.).

Zusätzlich zu Dialogsystem, Modell- bzw. Methodenbank und Datenbank wird in anderen Vorschlägen häufig gesondert eine *Wissensbank* berücksichtigt. Diese sorgt für die Integra-

tion unternehmensweiter Wissensbanken und Methoden der künstlichen Intelligenz in das DSS (vgl. Turban et al., 2011b, S. 86). Im vorliegenden Ansatz sind diese nicht gesondert berücksichtigt, sondern in den Komponenten Modell- und Methoden- sowie Datenbank enthalten.

Die drei Komponenten (Dialogsystem, Modell- bzw. Methodenbank und Datenbank) unterstützen die Zuordnung von DSS zu den unterschiedlichen Klassen nach SIGDSS. Die einzelnen Komponenten eines DSS sind je nach Klasse unterschiedlich stark ausgeprägt (vgl. Power, 2002, S. 16f.). Dokumentengetriebene DSS verfügen beispielsweise über eine ausgeprägte Datenbank-Komponente, da sie den Entscheidungsträger beim Organisieren und Auffinden unstrukturierter Daten unterstützen. Dies wird der Komponente Datenbank nach Sprague und Carlson zugeordnet (vgl. Power, 2002, S. 18). Die Modell-Komponente beschränkt sich bei diesen Systemen auf Such-Methoden für das Auffinden relevanter Dokumente. Die fokussierten MDSS greifen hauptsächlich auf *formalisierte Modelle* zurück. Wie die Bezeichnung vermuten lässt, dominiert in diesen Systemen die Modellbank. Anhand der Modelle wird versucht die Eigenschaften und Zusammenhänge einer konkreten, realen Entscheidungssituation möglichst detailliert abzubilden (vgl. Power, 2002, S. 158). Prozedural betrachtet wird hierfür das Wissen des Entscheidungsträgers über die reale Situation in ein explizites Modell überführt, wobei Annahmen getroffen und situationsbedingte Vereinfachungen vorgenommen werden (vgl. Balzert, 2000, S. 100; Gluchowski et al., 2008, S. 64). Dies geschieht anhand einer durch das MBMS zur Verfügung gestellten Modellierungssprache. Das explizite Modell wird mittels einer geeigneten Methode im nächsten Schritt gelöst und abschließend wird das Ergebnis der Modellanwendung in die Realität zurück transformiert (vgl. Gluchowski et al., 2008, S. 64).

Ein Entscheidungsträger kann in komplexen Entscheidungssituationen mithilfe eines oder mehrerer Modelle relevante Zusammenhänge, Variablen und Parameter leichter abschätzen und sich einen besseren Überblick über etwaige Handlungsalternativen verschaffen. Aufgrund der Tatsache, dass Modelle die zentrale Komponente von MDSS sind und einen zentralen Bestandteil dieser Arbeit ausmachen, werden im folgenden Kapitel 2.2 das dieser Arbeit zugrunde liegende Modellverständnis und der Begriff der konzeptionellen Modellierung in diesem Kontext dargestellt.

2.2 Modellierung

Der Begriff Modellierung drückt gemeinhin die Schaffung und den Aufbau eines Modells aus (vgl. Sinz, 1997, S. 271). Die vorliegende Arbeit betrachtet die Erstellung solcher Modelle, die von einem MDSS für die Ausführung von Analysen genutzt werden können. Eine weitere Spezialisierung bezüglich des hier diskutierten Verständnisses von Modellierung und berücksichtigten Modellen wird durch den *konzeptionellen Charakter* dieser ausgedrückt. Abschnitt 2.2.2 widmet sich dieser Thematik eingehend.

2.2.1 Modellbegriffe

Im Folgenden werden vier in der Wirtschaftsinformatik gebräuchliche Modellbegriffe auf Basis der Arbeiten von Thomas und vom Brocke erläutert (vom Brocke, 2003; Thomas, 2005). Der allgemeine Modellbegriff nach Stachowiak sowie der axiomatische Modellbegriff bilden allgemein anerkannt eine Art formales Fundament für den Bereich Modellierung und weisen einen eher grundlegenden Charakter auf. Die Auffassungen des abbildungs- und des konstruktionsorientierten Modellbegriffs werden hingegen kontrovers diskutiert.

2.2.1.1 Allgemeiner Modellbegriff nach Stachowiak

Stachowiak expliziert im allgemeinen Modellbegriff die drei Merkmale *Abbildung*, *Verkürzung* sowie *Pragmatismus*. Ein Modell repräsentiert demzufolge immer ein natürliches oder ein künstliches Original, wobei Modelle selbst wiederum ein Original darstellen können (vgl. Stachowiak, 1973, S. 131). Eine Verkürzung bezeichnet das Weglassen einer subjektiv aus Sicht des Modellierers und auf Basis dessen Modellierungszwecks (vgl. Balzert, 2000, S. 100; Frank, 1997, S. 99; Schwarzer und Krcmar, 2010, S. 96) irrelevant erachteten Eigenschaft des Originals (vgl. Stachowiak, 1973, S. 132). Das Merkmal Pragmatismus drückt den Modellierungszweck von Modellen aus. Ein Modell ist per se nicht allein die autonome Abbildung eines Originals, sondern vielmehr die Abbildung eines Originals für eine bestimmte Person zu einer gegebenen Zeit und für einen gewissen Zweck (vgl. Stachowiak, 1973, S. 132f.).

2.2.1.2 Axiomatischer Modellbegriff

Der axiomatische Modellbegriff (oft auch als mathematischer Modellbegriff bezeichnet) geht in seinen Ursprüngen auf das Teilgebiet Logik in der Mathematik zurück (vgl. Ort-

lieb, 2008, S. 55). Modelle werden mittels Zeichen[3] formuliert, die als Abbildungsmittel zu interpretieren sind. Zeichen sind dabei elementare informative Bausteine eines Modells, die sich mittels der drei Dimensionen *Syntax*, *Semantik* und *Pragmatismus* unterscheiden lassen (vgl. de Molière, 1984, S. 42f.; Stegmüller, 1974, S. 30ff.). Syntax bezeichnet in diesem Zusammenhang den Aufbau und die formalen sowie logischen, strukturellen Beziehungen von Zeichen untereinander, während Semantik die Beziehung zwischen Zeichen derart charakterisiert, dass die an sich inhaltsleeren Zeichen mit einer Bedeutung[4] versehen werden. Die Sicht des Modells als Gesamtheit, Syntax und Semantik einschließend, wird durch die Dimension des Pragmatismus repräsentiert (vgl. de Molière, 1984, S. 43). Insbesondere im Bereich des Operations Research, in dem mathematisch geprägte Modelle eine wichtige Rolle einnehmen, erfährt die Dimension des Pragmatismus im Ziel der Ermittlung einer mathematisch optimalen Lösung eine hohe Relevanz.

2.2.1.3 Abbildungsorientierter Modellbegriff

Kosiol bezeichnet ein Modell als „adäquates Abbild der betrachteten Wirklichkeit“ (Kosiol, 1961, S. 321, zitiert nach Thomas, 2005, S. 14 (ohne Hervorhebungen)). Adäquat bedeutet in diesem Zusammenhang, dass die subjektiv relevanten Merkmale oder Eigenschaften eines repräsentierten realen Originals strukturerhaltend abgebildet werden (vgl. Witte et al., 1975, S. 7f.). Der grundlegende Unterschied zum allgemeinen Modellbegriff nach Stachowiak ist ein zwingend vorhandenes reales Problem (vgl. Thomas, 2005, S. 14). Der betrachtete, verkürzte Ausschnitt der Realwelt wird auch als *Diskurssystem* oder *Diskurswelt* bezeichnet (vgl. Ferstl und Sinz, 2008, S. 5), wobei der subjektiv interpretierte Ausschnitt dieses Diskurssystems wiederum das *Objektsystem* bildet (vgl. Thomas, 2005, S. 15). Die subjektive Abbildung dieses Objektsystems schließlich wird als *Modellsystem* bezeichnet. Es wird auf Grundlage eines Metamodells konstruiert.

An einem Modellierungsprozess sind demnach das in Bezug auf die Realwelt verkürzte und subjektiv interpretierte Objektsystem sowie ein Metamodell beteiligt. Das Objektsystem wird dabei anhand von Sprachkonstrukten formalisiert in einem Modellsystem abgebildet, wobei ein Metamodell diese Sprachkonstrukte zur Verfügung stellt.

---

[3] Im Vergleich zur allgemein üblichen Bedeutung des Begriffs Zeichen wird in Bezug auf die mathematische Modellierung die Bedeutung insofern eingeschränkt, als dass ein Zeichen lediglich als Modellbestandteil zu interpretieren ist. Eine sonst übliche Auffassung von Zeichen als Modell selbst wird nicht verfolgt (vgl. de Molière, 1984, S. 42).

[4] Eine Bedeutung ist in diesem Zusammenhang als Zuordnung von Zeichen zu einem bestimmten Objektbereich zu interpretieren (vgl. de Molière, 1984, S. 43).

2.2.1.4 Konstruktionsorientierter Modellbegriff

Der konstruktionsorientierte Modellbegriff verfolgt die Subjektivität der Wahrnehmung (vgl. vom Brocke, 2003, S. 12). Eine Modellbildung ist immer Ergebnis einer Konstruktion, die nicht einer vorgegebenen Realität zu entsprechen hat, sondern für die Zwecke einer Person brauchbar ist (vgl. Lehner et al., 1995, S. 25). Neben dieser Akzentuierung des Konstruktionsaspekts kommt einem Problembezug eine zentrale Rolle zu (vgl. Rieper, 1992, S. 25). Ein reales Problem ist hier die subjektiv wahrgenommene Differenz zwischen einem tatsächlich erreichten Ist- und einem Soll-Zustand (vgl. Rieper, 1992, S. 19), „verbunden mit einem ursprünglichen Mangel an Wissen über Möglichkeiten, diese Lücke zu schließen“ (Bretzke, 1980, S. 34). „Das Wesen eines wirklichen Problems besteht […] in einem Mangel an Struktur“ (Bretzke, 1980, S. 34).

Dieser Sachverhalt wird auch in der Definition von Schütte deutlich, die in der Wirtschaftsinformatik allgemein anerkannt ist: „Ein Modell ist das *Ergebnis einer Konstruktion* eines Modellierers, der für *Modellnutzer* eine Repräsentation eines Originals zu einer *Zeit* als relevant mit Hilfe einer *Sprache* deklariert“ (Schütte, 1998, S. 59, zitiert nach vom Brocke, 2003, S. 12). Da für die Konstruktion eines Modells im Sinne eines Modellnutzers durch einen Modellierer ein hoher Abstimmungsbedarf zwischen Modellierer und Benutzer unterstellt werden kann, ist eine strukturierte Darstellung der Situation im Sinne des Benutzers zwingend erforderlich (vgl. vom Brocke, 2003, S. 12).

2.2.1.5 Subsumtion des angewandten Modellverständnisses

Im weiteren Verlauf dieser Arbeit wird hauptsächlich die Auffassung des konstruktionsorientierten Modellbegriffs verfolgt, da bei der Erstellung eines Modells unter Nutzung der konzeptuellen Modellierungsebene für MDSS in jedem Fall ein klarer Problembezug besteht. Die betrachteten Modelle werden immer für die Zwecke bzw. für das Erreichen konkreter Ziele eines Modellbenutzers erstellt und dienen der Analyse einer Entscheidungssituation. Auf diesem Weg soll der Entscheidungsträger mit relevanten Informationen versorgt werden, die dem Aufzeigen von Entscheidungsalternativen dienen. Ein Modell stellt daher eine strukturierte Darstellung der Problemsituation im Sinne eines Modellbenutzers dar. Entsprechend dieser strukturierten Darstellungen werden Modellelemente unter Verwendung einer Modellierungssprache miteinander in Relation gesetzt. Bei der Interpretation von Modellen ist jedoch nicht ausschließlich das konstruktionsorientierte Modellverständnis zu berücksichtigen. Syntax und Semantik aus dem Verständnis des Mathematischen Modellbegriffs müssen ebenfalls beachtet werden, da gerade diesen Elemen-

ten für die Erstellung einer konzeptuellen Modellierungsebene für MDSS eine tragende Rolle zukommt. Die Syntax definiert diesbezüglich die Regeln zur Verwendung der Elemente einer Modellierungssprache und die Semantik ermöglicht Aussagen über die Interpretation der zusammengestellten Elemente zu treffen. Eine ausschließliche Akzentuierung der formalen Aspekte (im Sinne von Zeichen) und damit ein zugrunde gelegtes Verständnis des mathematischen Modellbegriffs kann jedoch für diese Arbeit nicht als zielführend erachtet werden. Vielmehr soll ein Problem für einen gewissen Zweck unter Einhaltung formaler Aspekte adäquat repräsentiert werden können.

2.2.2 Konzeptuelle Modellierung

Ein Konzept ist in der Wirtschaftsinformatik zu verstehen als „das Ergebnis eines Gedankenexperiments […] oder einer gedanklichen Konstruktion, das Vorbild für die Wirklichkeit ist“ (Heinrich et al., 2004, S. 379). Vom Wortstamm aus betrachtet unterscheiden sich *konzeptuelle* und *konzeptionelle* Modelle dahingehend, dass ein konzeptuelles Modell ein Konzept aufweist, d.h. eine gedankliche Konstruktion in Form eines Modells ausdrückt und ein konzeptionelles Modell ein Konzept betrifft, d.h. eine gedankliche Konstruktion adressiert. In der Wirtschaftsinformatik werden beide Begriffe jedoch häufig synonym verwendet, z.B. von Frank (vgl. Frank, 2000, S. 341).

Konzeptuelle Modelle werden als Versuch betrachtet, ein Konzept bzw. gedankliches Konstrukt mittels eines Modells in der Realität nutzbar zu machen. Sinn und Zweck der konzeptuellen Modellierung ist die Erschaffung eines Rahmenwerkes für die formalisierte Repräsentation eines realen Originals in einem gegebenen Kontext, um die Möglichkeit zur Evaluation eines Modells mittels Validierung und Verifikation[5] zu erhalten (vgl. Lehner et al., 1995, S. 136f.). Im thematisch an diese Arbeit angrenzenden Bereich der Datenmodellierung werden mithilfe konzeptueller Modellierung Qualitätsziele für die zu entwerfende Datenbankapplikation verfolgt (vgl. Rauh und Stickel, 1997, S. 30ff.). Ein konzeptuelles Datenmodell schafft die wesentlichen Voraussetzungen für eine Datenunabhängigkeit in Anwendungsprogrammen, die auf einer Datenbank aufsetzen (vgl. Gabriel und Röhrs, 1994, S. 271; Vetter, 1991, S. 85). Das Ziel der Unabhängigkeit von Anwendungsprogrammen lässt sich auf den Bereich MDSS übertragen.

[5] Validierung beschreibt die Korrektheit einer Repräsentation in Bezug auf den gewählten Repräsentationsformalismus, während Verifikation die Korrektheit in Bezug auf eine Problemstellung darstellt (vgl. Lehner et al., 1995, S. 137).

Um reale, sehr detaillierte Situationen abbilden zu können wird in Modellen neben der Verkürzung häufig abstrahiert. Während eine Verkürzung von der Betrachtung *eines* konkreten Einzelfalls abgeleitet wird, wird bei einer Abstraktion von *mehreren* Einzelfällen auf eine für all diese Instanzen gültige, stellvertretende Betrachtung bzw. Repräsentation geschlossen (vgl. Vetter, 1991, S. 8).

Die zur Verfügung stehenden Elemente zur Abbildung realer Zusammenhänge werden durch eine Modellierungssprache definiert und das Konzept gibt vor, welche Elemente zu welchem Zweck genutzt werden. Wand und Weber unterbreiten ein Framework für die konzeptuelle Modellierung, das neben einer konzeptuellen Modellierungssprache und einem Konzept (konzeptuelle Modellierungsmethode) ein konzeptuelles Modellierungsskript sowie den Kontext eines Modells betrachtet (vgl. Wand und Weber, 2002, S. 363ff.). Die konzeptuelle Modellierungssprache definiert die zur Verfügung stehenden Konstrukte und reguliert deren Kombination bei der Abbildung von Ausschnitten der Real-Welt. Die Modellierungsmethode macht in diesem Zusammenhang eine Aussage darüber, welche Objekte der realen Welt in einem Modell auf welche Weise abgebildet werden. Das konzeptuelle Modellierungsskript ist das entstehende Produkt dieses Modellierungsprozesses. Jedes Skript (konzeptuelles Modell) ist demnach eine Darstellung, welche mittels der Modellierungssprache erzeugt wird. Der Kontext setzt sich aus mehreren Faktoren zusammen, die das Umfeld eines konzeptuellen Modells beschreiben und wird nicht näher betrachtet.

Konzeptuelle Modelle werden z.B. benutzt, um die Anforderungen einer Applikation zu definieren und so dem Anwender der Applikation das Verständnis gewisser Zusammenhänge zu erleichtern (vgl. Wand und Weber, 2002, S. 363). Anhand abstrahierter Zusammenhänge, Prozesse und Elemente fällt es dem Anwender leichter sich einen Überblick über eine Situation zu verschaffen, diese richtig einzuschätzen und richtige Schlussfolgerungen aus der Anwendung solcher Modelle zu treffen, wobei gerade der letzte Punkt stark von der Integration des Modells mit der Anwendungsumgebung (Kontext) abhängig ist (vgl. Davies et al., 2006, S. 371ff.).

Weber und Zhang schlagen für konzeptuelle Modellierungsansätze das Konstrukt *minimale ontologische Überschneidung* (MOO - minimal ontological overlap) vor (vgl. Weber und Zhang, 1996). MOO bezieht sich auf die zur Verfügung stehenden Modellierungselemente, die sich in ihrer Aussage und Struktur soweit wie möglich disjunkt gegenüber jeglichen weiteren Modellierungselementen verhalten sollen. Dies bedeutet auch, dass sie nicht durch eine Kombination weiterer Elemente konstruiert werden können. Green erarbeitet

auf dieser Basis das Konstrukt der *maximalen ontologischen Vollständigkeit* (MOC - maximum ontological completeness). MOC wird erreicht, sobald mittels der zur Verfügung stehenden Elemente der konzeptuellen Modellierungssprache alle realen Situationen abgebildet werden können (vgl. Green und Rosemann, 1999, S. 236; Green, 1997, S. 7). Die Qualität einer konzeptuellen Modellierungsebene für MDSS hängt eng mit der Umsetzung der beiden Konstrukte MOO und MOC zusammen. Je umfassender die Sprachkonstrukte disjunkt zueinander sind und je mehr reale Situationen mittels der Sprache abgebildet werden können, desto besser ist die Qualität der konzeptuellen Modelle zu beurteilen, die unter Verwendung der Sprache erstellt werden. Je weniger sich die Sprachkonstrukte überschneiden, desto leichter ist die Interpretation von Modellen und je mehr reale Sachverhalte sich mittels der Sprachkonstrukte modellieren lassen, desto weniger ist der Modellierer zu (ungewollten) Verkürzungen und Annahmen gezwungen.

Der folgende Abschnitt 2.3 gibt eine Übersicht über modellgetriebene Decision Support Systeme. Anhand einer Klassifizierung unterschiedlicher Modelltypen wird abschließend abgegrenzt, welche MDSS mit dem in der Arbeit beschriebenen konzeptuellen Modellierungsansatz adressiert werden.

2.3 Klassifizierung modellgetriebener Decision Support Systeme

Kapitel 2.1.2 hat vier allgemeine Klassifizierungsvorschläge für DSS auf Basis unterschiedlicher Kriterien und Merkmale vorgestellt. Die im Mittelpunkt dieser Arbeit stehende Klasse MDSS umfasst ein sehr breites Feld (vgl. Power, 2002, S. 157). Die Entwicklung neuer Methoden und immer leistungsfähigerer Softwareprodukte, die ein großes Spektrum verschiedener Analysemöglichkeiten und damit auch unterschiedlicher Modelltypen unterstützen, lässt eine weitere Untergliederung verschiedener MDSS sinnvoll erscheinen. Insbesondere der Klassifizierungsvorschlag von Alter, der unterschiedliche Modelltypen für den allgemeinen Bereich DSS unterscheidet, ist für eine detailliertere Differenzierung von MDSS relevant.

2.3.1 Klassifizierungskriterium

Als Kriterium zur Unterscheidung verschiedener MDSS-Typen wird im Folgenden der verwandte Modelltyp herangezogen. Die Art eines Modells konfrontiert den Anwender vor allem im Modellierungsprozess mit speziellen Anforderungen und Eigenschaften. Im Endeffekt führt dies zu Typ-spezifischen Modellierungswerkzeugen. Der Modelltyp bietet sich

daher im Rahmen dieser Arbeit als Klassifizierungskriterium für MDSS an, um eine konkrete Vorstellung der adressierten MDSS zu erlangen.

Auch die Berücksichtigung der Modellierung auf konzeptueller Ebene im Kontext der vorliegenden Arbeit beeinflusst die Klassifizierungsstrategie von MDSS. Konzeptuelle Modelle haben einen universelleren Charakter, als es bei Modellen der Fall ist, für welche die bestehenden Klassifizierungsansätze entworfen wurden. Gluchowski et al. sprechen in diesem Zusammenhang von logischen Modellen im engeren Sinne, die für das Aufzeigen und Bewerten alternativer Handlungsmöglichkeiten genutzt werden. Sie unterscheiden *deterministische* und *stochastische*, *statische* und *dynamische*, *lineare* und *nichtlineare*, *ein-* und *mehrkriterielle*, sowie *scharfe/exakte* und *unscharfe* sowie *optimierende* und *satisfizierende* Modelle (vgl. Gluchowski et al., 2008, S. 68f.). Für die vorliegende Arbeit ist diese Klassifizierung ungeeignet, da sie die mathematischen Eigenschaften von Modellen fokussiert und somit Aussagen über enthaltene Methoden oder geeignete Algorithmen für die Anwendung eines Modells ermöglicht. Des Weiteren beschreibt der Vorschlag lediglich binär ausgeprägte Eigenschaftspaare, von denen keine abgegrenzten Klassen für die Differenzierung von MDSS abgeleitet werden. Sprague und Watson schlagen eine deutlich gröbere Unterteilung nach dem *verfolgten Zweck*, dem *Umgang mit Zufall* und der *generellen Anwendbarkeit* vor (vgl. Sprague und Watson, 1996, S. 112). Auch diese Klassifizierung wird nicht weiter verfolgt, da das Kriterium Zweck dem Anspruch der Anwendungsunabhängigkeit einer konzeptuellen Modellierungsebene entgegensteht.

Der Klassifizierungsvorschlag für DSS nach Alter (vgl. Kapitel 2.1.2.2) unterscheidet unter anderem *Accounting-*, *Abbildungs-* (engl.: representational), *Optimierungs-* und *Empfehlungsmodelle* (vgl. Alter, 1980, S. 80ff.). Power unterbreitet mit *Accounting- und Finanz-*, *Entscheidungsanalyse-*, *Prognose-*, *Netzwerk- und Optimierungs-* und *Simulationsmodellen* einen sehr ähnlichen Vorschlag (vgl. Power, 2002, S. 161ff.). Beide Klassifizierungen charakterisieren die einzelnen Klassen anhand konkreter Eigenschaften und sind kompatibel zu den Anforderungen einer konzeptuellen Modellierung. Sie bilden die Grundlage der verfolgten Klassifizierung und werden unter den folgenden fünf Modelltypen zusammengefasst: Accounting- und Finanz-, Entscheidungsanalyse-, Prognose-, Optimierungs- und Simulationsmodelle.

2.3.2 Accounting- und Finanzmodelle

Accounting- und Finanzmodelle fokussieren betriebswirtschaftliche Kennzahlen. Zu dieser Klasse gehören Modelle für Break-Even Analysen, die i.d.R. einen schnellen Überblick

über Themengebiete wie Preise, Mengen und Profit geben, Budgetierungsmodelle sowie Modelle für Finanz- und Kennzahlenanalysen. Budgetierungsmodelle ermöglichen über Web-basierte Schnittstellen das kollaborative Erstellen eines unternehmensweiten Budgetplans. Für die strategische Ausrichtung eines Unternehmens können Finanz- und Kennzahlenanalysen sehr wichtig sein. Es ist das Ziel, zukünftige Umsätze und Profite auf Basis von Daten aus aktuellen und zurückliegenden Perioden sowie absehbaren Entwicklungen zu schätzen, damit bei adäquater Interpretation der Analyseergebnisse geeignete Maßnahmen zur Sicherung des unternehmerischen Erfolgs bereits frühzeitig eingeplant und vorgenommen werden können (vgl. Power, 2002, S. 161ff.). In Accounting- und Finanzmodellen werden Variablen durch grundlegende Rechenoperationen wie Addition und Multiplikation zu einem logischen Modell verknüpft, um so dem Entscheidungsträger relevante Informationen und Analysemöglichkeiten bereitstellen zu können.

2.3.3 Entscheidungsanalyse-Modelle

Modelle dieser Kategorie werden in neuen, unbekannten Entscheidungssituationen eingesetzt und unterstützen den Benutzer bei der Identifikation möglicher Alternativen (vgl. Power und Sharda, 2007, S. 1047). Sie ermöglichen Aussagen über Sensitivitäten bezüglich eines zuvor definierten Ziels und beschränken sich zumeist auf eine geringe Anzahl möglicher Vorgehensweisen. Das Kernziel einer Entscheidungsanalyse und damit auch eines Entscheidungsanalyse-Modells ist es, eine Problemsituation zu verstehen und eine Strukturierung durch ein hierarchisches Objektmodell vorzunehmen. Für unikriterielle Probleme kommen z.B. Entscheidungsbäume und für multikriterielle Probleme Analytische Hierarchie Prozesse (AHP)[6] oder Bayes'sche Netze[7] zum Einsatz (vgl. Power, 2002, S. 165f.). Entscheidungsanalyse-Modelle fokussieren die Struktur beteiligter Objekte

[6] „Der Analytic Hierarchy Process ist eine von Thomas L. Saaty entwickelte Methode zur Modellierung schlecht strukturierter, multikriterieller Entscheidungsaufgaben und zählt zu den hierarchisch additiven Gewichtungsverfahren (vgl. Saaty, 1980; Weber, 1993, S. 73ff.). Diese Verfahren zeichnen sich durch die Auffassung aus, dass komplexe Entscheidungen in Abhängigkeit der konkreten Problemsituation und den individuellen Präferenzen des Entscheidungsträgers durch eine Hierarchie von Zielgrößen strukturiert darstellbar sind (vgl. Woll, 2000, S. 25)" (Gelhoet, 2010, S. 137). Über die Analyse der Präferenzen des Entscheidungsträgers wird versucht Erkenntnisse über die Entscheidungssituation zu gewinnen.

[7] „Bayes'sche Netze, auch Belief-Netze oder probabilistische Netze genannt, sind graphische Modelle zur visuellen Repräsentation der Interaktion zwischen Variablen. Ein Bayes'sches Netz besteht aus Knoten und Kanten zwischen den Knoten. Jeder Knoten entspricht einer Zufallsvariable X und hat den Wert, welcher der Wahrscheinlichkeit der Zufallsvariablen P(X) entspricht. Wenn es eine direkte Kante von Knoten X zu Knoten Y gibt, so deutet dies an, dass X direkten Einfluss auf Y hat" (Alpaydin, 2008, S. 53). Die Auswertung der Verbindungen von Knoten ermöglicht Rückschlüsse auf die Einflussfaktoren einer Entscheidungssituation.

eines Entscheidungsproblems, um dieses genauer zu verstehen und dienen weniger einer Unterstützung bei der Wahl einer konkreten aus mehreren Alternativen.

2.3.4 Prognosemodelle

Die zentrale Aufgabe von Prognosemodellen besteht in der Vorhersage von Variablenwerten zu einem bestimmten zukünftigen Zeitpunkt. Angewandte Methoden sind z.B. Beurteilungsmethoden (engl.: judgment methods), die mittels Expertenwissen und unter Berücksichtigung externer Faktoren versuchen, Werte von Modellvariablen in zukünftigen Perioden zu bestimmen (vgl. Power, 2002, S. 168). Aufgrund der oftmals großen Ungenauigkeit solcher Modelle werden sie vorrangig in Problemsituationen eingesetzt, in denen das Aufzeigen eines Trends im Vordergrund steht und das Bestimmen exakter Werte eher zweitranging ist. Für die kurzfristige Vorhersage der Werte von Modellvariablen wird z.B. die Methode des gleitenden Durchschnitts eingesetzt. Auf Basis von Durchschnittswerten historischer Daten erfolgt eine Fortschreibung der betrachteten Variablenwerte einer oder weniger zukünftiger Perioden. Der Charakter von Prognosemodellen ist allgemein sehr formal und mathematisch geprägt. Es werden häufig unterschiedliche Methoden kombiniert und sequentiell angewandt.

2.3.5 Optimierungsmodelle

Optimierungsmodelle dienen der Identifikation und Analyse von Maximierungs- oder Minimierungselementen innerhalb eines Entscheidungsproblems, das mathematisch beschrieben werden kann. In solchen Modellen werden oft viele Objekte miteinander kombiniert, die jeweils speziellen Bedingungen unterliegen (vgl. Alter, 1980, S. 84f.). Die wohl bekannteste Methode zur Erstellung und Analyse von Optimierungsmodellen ist die Lineare Programmierung (vgl. Turban et al., 2011a, S. 152). Typische Anwendungsgebiete finden sich in der Ressourcenallokation und der Planung, wodurch ein direkter Hinweis auf den Charakter solcher Modelle gegeben ist. Optimierungsmodelle können nur in gut strukturierten Situationen, die mittels diskret formulierter Eigenschaften zur Beschreibung realer Begebenheiten repräsentiert werden können, sinnvoll angewandt werden. Häufig werden Optimierungsmodelle für Teilprobleme eines Entscheidungsproblems formuliert, für die genügend gut strukturiertes Wissen vorliegt (vgl. Alter, 1980, S. 85f.).

2.3.6 Simulationsmodelle

Simulation wird als Begriff in seiner weitesten Definition für jede Art von Modellbetrachtung gebraucht (vgl. Witte, 1973, S. 17). Im Kontext von MDSS beschreibt Simulation

jedoch einen deutlich engeren und stärker abgegrenzten Bereich. Während quantitative Modelle häufig ein verkürztes Abbild der Realität darstellen, besteht der Anspruch von Simulationsmodellen in der Nachahmung realer Begebenheiten mit deutlich weniger vereinfachenden Annahmen bzw. Verkürzungen (vgl. Power, 2002, S. 172). Die Modellbildung ist in diesem Zusammenhang ein zentrales Merkmal von Simulation (vgl. Law und Kelton, 2000, S. 4).

Simulationsmodelle werden in Situationen genutzt, die auf Grund großer Komplexität realer Begebenheiten und eines häufig damit einhergehenden hohen Anteils an Zufall nicht mittels anderer Modelltypen abgebildet werden können. Der Modellbildungsprozess ist häufig sehr aufwendig und spezifisch auf eine Problemsituation ausgerichtet, so dass die Wiederverwendbarkeit der Modelle gering ist (vgl. Turban et al., 2011a, S. 171ff.).

2.4 Adressierte modellgetriebene Decision Support Systeme

Das zentrale Ziel dieser Arbeit, der Entwurf einer konzeptuellen Modellierungsebene für MDSS, adressiert nicht alle Typen von MDSS. Anhand der fünf zuvor beschriebenen Modelltypen wird nun spezifiziert, für welche Typen von MDSS die konzeptuelle Modellierungsebene entworfen wird.

Das Modell als zentraler Bestandteil und Kernkomponente eines MDSS führt zwangsläufig zur Betrachtung des Modellerstellungsprozesses bzw. der Modellierung. Die adressierten MDSS zeichnen sich durch das Nutzen von Spreadsheet-Werkzeugen als Plattform für die Implementierung und Gestaltung der Modellerstellungskomponente aus. Aus technologischer Sicht werden in diesem Zusammenhang zumeist Spreadsheet-Pakete, wie z.B. Microsoft Excel, genutzt (vgl. Power und Sharda, 2007, S. 1050), da diese neben der Spreadsheet-Komponente eine Vielzahl weiterer Funktionen und Schnittstellen zu etablierten Softwareprodukten bieten, sodass eine Weiterverarbeitung gewonnener Informationen erleichtert wird.

Die konzeptuelle Modellierungsebene adressiert weder Simulations- noch Entscheidungsanalyse-Modelle. Simulationsmodelle werden nicht betrachtet, da sie nur äußerst selten anhand eines Spreadsheet-Werkzeugs umgesetzt werden und inhaltlich das Nachahmen der Realität eine untergeordnete Rolle spielt. Entscheidungsanalyse-Modelle werden aufgrund ihres Fokus‘ auf Objektstruktur und Hierarchiebildung nicht behandelt.

Modelle in MDSS, für welche die konzeptuelle Modellierungsebene entworfen wird, bestehen in der Regel aus weniger komplexen mathematischen Konstrukten. Der Fokus liegt

auf der Betrachtung der Modellstruktur und sie befassen sich eingehend mit Relationen und der logischen Formulierung realer Zusammenhänge und Begebenheiten, unter Berücksichtigung von Verkürzung und Abstraktion.

Accounting- sowie Prognosemodelle (vgl. Abschnitt 2.3.2 und 2.3.4) repräsentieren den größten durch diese Arbeit adressierten Teil von Modellen und damit auch MDSS. Bei der Modellierung liegt die Aufmerksamkeit hauptsächlich auf der logischen Modellstruktur und dem formalen Definieren von Wirkungszusammenhängen zwischen zwei oder mehreren Modellvariablen. Komplexe mathematisch geprägte Algorithmen, wie z.B. die Lineare- oder die gemischt-ganzzahlige Programmierung, werden nicht vorrangig betrachtet, gleichwohl solche Modelle sehr häufig mittels Spreadsheet-Paketen umgesetzt werden.

Multidimensionalität ist ein weiteres Merkmal, welches den Charakter von Modellen und damit den MDSS, für welche die konzeptuelle Modellierungsebene entworfen wird, maßgeblich prägt. Das Vorliegen multidimensionaler Strukturen ist essentieller Bestandteil der Betrachtung und wirkt sich insbesondere auf die Modellierung logischer Strukturen aus. Die Besonderheiten von Multidimensionalität bei der Definition der adressierten Modelle wird im Rahmen der Anforderungen in Kapitel 3 (besonders) betrachtet. Des Weiteren werden verwandte Modellierungsansätze vorgestellt, um deren Eignung zur Umsetzung der konzeptionellen Modellierungsebene zu prüfen.

Zusammenfassend lassen sich also folgende Eigenschaften benennen, die charakteristisch für die von der konzeptuellen Modellierungsebene adressierten MDSS sind:

- Spreadsheet-basierte Modelle
- multidimensionale Betrachtung einer Situation
- Betrachtung der logischen Zusammenhänge zwischen zwei oder mehreren Modellvariablen
- weniger komplexe mathematische Konstrukte

3 Analyse der Anforderungen und bestehende Modellierungsansätze

Kapitel 2.4 hat die charakteristischen Eigenschaften von MDSS dargestellt, für welche eine konzeptuelle Modellierungsebene entworfen werden soll. Aus diesen Eigenschaften der adressierten MDSS ergeben sich Anforderungen. Damit eine konzeptuelle Modellierungsebene die Qualität von Modellen und den Modellierungskomfort des Anwenders bei der Nutzung der adressierten MDSS nachhaltig positiv beeinflussen kann, müssen diese Eigenschaften und Anforderungen beim Konzeptentwurf berücksichtigt werden.

Kapitel 3.1 erläutert das Problem der Fehleranfälligkeit von Spreadsheet-Modellen, anhand derer die adressierten MDSS umgesetzt werden. Im folgenden Abschnitt 3.2 wird das Charakteristikum der Multidimensionalität der Modelle hinsichtlich der Modellelemente und des Modellierungsprozesses beschrieben. Abschnitt 3.3 stellt basierend auf den Elementen und Eigenschaften multidimensionaler Modelle die Anforderungen an eine konzeptuelle Modellierungsebene dar, die eine Modellierungskomponente für die positive Beeinflussung der Qualität von Modellen erfüllen muss. Kapitel 3.4 unterbreitet eine Analyse bestehender Ansätze multidimensionaler Modellierung und formuliert auf Basis der Anforderungen aus Abschnitt 3.3 eine Qualitätsbeurteilung für die Nutzung der Ansätze im Kontext dieser Arbeit. Kapitel 3.5 fasst die Betrachtung dieser Ansätze auf Basis des Anforderungskatalogs als State-of-the-Art zusammen.

3.1 Fehler in Spreadsheet-Modellen

Modelle fungieren als zentraler Bestandteil eines MDSS und verleihen diesem die typischen Charakteristika und Analysemöglichkeiten. Fehlerhafte Modelle gefährden die sinnvolle Anwendbarkeit eines MDSS und beeinträchtigen die Unterstützungsleistung, die einem Entscheidungsträger zur Verfügung gestellt werden kann. Fehlerhafte Modelle werden in verschiedenen Forschungsarbeiten adressiert. Aufgrund der Popularität von Spreadsheet-Werkzeugen in vielen Unternehmen werden fehlerhafte Spreadsheet-Modelle in der Wissenschaft vermehrt untersucht. Aufgrund der Tatsache, dass die betrachteten MDSS vorwiegend mittels Spreadsheet-Werkzeugen implementiert werden, werden im Folgenden Forschungsergebnisse zu Fehlern in Spreadsheet-Modellen vorgestellt. Diese müssen – wie alle weiteren Fehler, die die Qualität von Modellen beeinträchtigen – beim Entwurf der konzeptuellen Modellierungsebene berücksichtigt werden.

Fehler in Spreadsheets werden vermehrt seit Mitte der 1990er Jahre in zahlreichen Forschungsarbeiten untersucht. In Europa hat sich die *European Spreadsheet Risk Interest Group* (EuSpRIG) formiert, welche eine jährliche Konferenz zu diesem Themenkomplex veranstaltet.[8] Forschungsprojekte, welche sich der Häufigkeit widmen, mit denen Fehler auftreten, kommen größtenteils zu sehr ähnlichen Ergebnissen. Panko untersuchte 2008 sieben Studien und fasste diese zusammen. Er kommt dabei zu dem Ergebnis, dass ca. 88% aller Spreadsheet-Modelle Fehler enthalten (vgl. Panko, 2008). Die Ergebnisse der Fehlerrate pro Zelle divergieren hingegen sehr stark. So streuen die Werte in den ersten Studien zu diesem Thema von ca. 1,1 % fehlerhaften Zellen (vgl. Panko und Sprague, 1998, S. 347) bis hin zu einem Wert von 21 % fehlerhaften Zellen (vgl. Olson und Nilsen, 1987). Aktuelle Untersuchungen kommen zu dem Schluss, dass ein Wert von ca. 1-2% fehlerhaften Zellen in Spreadsheet-Modellen realistisch ist (vgl. Powell et al., 2009, S. 126). Galletta et al. weisen in ihrer Publikation darauf hin, dass die vielen unterschiedlichen Ergebnisse bezüglich der Häufigkeit von Fehlern auf Unterschiede im Vorgehen zum Aufdecken dieser zurückgehen. So ist es z.B. ausschlaggebend, ob ein Spreadsheet-Modell direkt auf dem Bildschirm oder ausgedruckt auf Papier vorliegt, während es auf Fehler untersucht wird (vgl. Galletta et al., 1997). Dieser Aspekt soll hier jedoch vernachlässigt werden, da er dem allgemeinen Tenor der Fehleranfälligkeit von Spreadsheet-Modellen nicht entgegensteht.

Weitere Forschungsaktivitäten befassen sich mit dem Verständnis und dem Differenzieren unterschiedlicher Fehlerarten, um mittels typischer Charakteristika unterschiedlicher Fehlerklassen Ansatzpunkte für Konzepte zur Fehlerreduktion zu gewinnen. Im Laufe der Zeit entstanden mehrere etablierte Klassifizierungsansätze[9]. Fehlerklassifizierungen können unter Berücksichtigung der verschiedensten Kriterien definiert werden. Die Ursache eines Fehlers zu betrachten stellt nur eine Möglichkeit dar. Weitere finden sich in dessen Auswirkungen, der Form oder dem Zeitpunkt, zu welchem dieser erkennbar wird. Derzeit existiert keine allgemein akzeptierte Klassifizierung (vgl. Powell et al., 2008, S. 129) und es ist strittig, ob dies überhaupt sinnvoll wäre, da es sehr viele Faktoren gibt, von denen die Beurteilung und Definition eines Fehlers abhängen kann. Eine Betrachtung der Auswirkungen hinsichtlich des finanziellen Schadens oder des ausgehenden Risikos eines Fehlers

[8] Auf der Internetseite (http://www.eusprig.org) finden sich z.B. Dokumentationen von Konsequenzen, die Fehler in Spreadsheet-Modellen unterschiedlicher Unternehmen nach sich zogen.

[9] Häufig herangezogene Fehlerklassifizierungen wurden z. B. in den folgenden Arbeiten thematisiert: (Cragg und King, 1993; Galletta et al., 1993; Panko und Halverson, 1996; Rajalingham, 2005; Rajalingham et al., 2000).

(vgl. Powell et al., 2008, S. 129) erscheint in entsprechenden Situationen sinnvoll, ist jedoch sehr schwierig zu bewerkstelligen. Als Konsens der Literatur kann festgestellt werden, dass eine Fehlerklassifizierung immer spezifizieren sollte, in welchem Kontext und aus welchem Grund sie entstanden ist. Dies ist im Rahmen der hier berücksichtigten Studien durch Beschreibung von Forschungsmethodik oder dem gewählten Vorgehen gegeben. Powell et al. explizieren 2008 diese Forderung und formulieren des Weiteren, dass jede Kategorie bzw. Klasse einer Klassifizierung genau definiert und voneinander abgegrenzt sein sollte. Außerdem sollte eine Klassifizierung zur Sicherstellung einer konsistenten Anwendung durch unterschiedliche Personen getestet werden (vgl. Powell et al., 2008, S. 129).

Es konnte festgestellt werden, dass Unternehmen oder Organisationen sich auch heute der Gefahr von fehlerhaften Spreadsheet-Modellen häufig nicht bewusst sind (vgl. McGill und Klobas, 2005; Powell et al., 2008, S. 128), obwohl selbst einzelne Fehler großen finanziellen Schaden verursachen können (vgl. Powell et al., 2009, S. 128ff.). Dementsprechend können signifikante Unterschiede in den von Seiten der Forschung empfohlenen und den praktisch implementierten Qualitätssicherungsmaßnahmen für Spreadsheet-Modelle festgestellt werden (vgl. Caulkins et al., 2007, S. 17).

Die Fehleranfälligkeit von Spreadsheet-Modellen muss unbedingt berücksichtigt werden, da Spreadsheet-Modelle für die Umsetzung der adressierten MDSS genutzt werden. Damit Fehler im Modell eines MDSS nicht auftreten, muss versucht werden ihr Entstehen im Modellierungsprozess zu verhindern. Kapitel 2.2.2 hat einen Überblick konzeptueller Modellierung gegeben und Ziele herausgearbeitet, die mit ihr verfolgt werden. Es fällt einem Anwender leichter Situationen zu überblicken, wenn ihm Zusammenhänge in Form von abstrahierten Informationen vorliegen. Konzeptuelle Modelle ermöglichen des Weiteren die Validierung und Verifikation der modellierten Situation und werden z.B. im Bereich der Datenmodellierung eingesetzt, um Qualitätsziele zu verfolgen (vgl. Rauh und Stickel, 1997, S. 30ff.). Die Trennung von konzeptuellem Datenmodell und Daten ist der zentrale Punkt dieses Ansatzes. Diese unabhängige Betrachtung lässt sich auf den Bereich der MDSS übertragen. Analog zur Datenmodellierung soll für die Modellierungskomponente der adressierten MDSS eine konzeptuelle Modellierungsebene entworfen werden, die eine Anwendungssystem-unabhängige Definition der logischen Zusammenhänge zwischen zwei oder mehr Modellvariablen ermöglicht. Dies bedeutet, dass bei der Modellierung eines konzeptuellen Modells jegliche Anforderungen mit Bezug auf eine Analyse und

proprietäre Werkzeug-Konzepte unberücksichtigt bleiben. Auf diese Weise soll die Anzahl von Fehlern in Spreadsheet-Modellen vermindert werden.

3.2 Analyse der Anforderungen multidimensionaler Modellierung

Multidimensionale Modelle konfrontieren den Benutzer und Modellersteller mit spezifischen Anforderungen. Im Folgenden werden daher die grundlegenden Eigenschaften von Multidimensionalität in Bezug auf die durch diese Arbeit adressierten MDSS betrachtet.

3.2.1 Dimensionen und Modellstruktur

Saylor et al. umschreiben den ursprünglichen Gedanken multidimensionaler Informationssysteme als Anordnen betriebswirtschaftlicher Kennzahlen entlang inhaltlich verbundener Beschreibungsobjekte (vgl. Gluchowski et al., 2008, S. 151; Saylor et al., 1997, S. 39). Diese zusammengehörigen Beschreibungsobjekte oder -gebiete werden unter dem Begriff *Dimension* zusammengefasst. Dimensionen sind aus einzelnen Elementen aufgebaut, die als *Dimensionsausprägungen* (im Folgenden auch kurz Ausprägung) bezeichnet werden und strukturelle sowie beschreibende Daten zur Attribuierung von Kennzahlen beinhalten. Strukturelle Daten ermöglichen das Bilden von Klassen und die Zuordnung einzelner Ausprägungen zu diesen (z.B. die Zuordnung eines Artikels zu einer Artikelgruppe), während beschreibende Daten z.B. die Farbe eines Artikels erläutern (vgl. Totok, 2000, S. 87). Innerhalb von Dimensionen werden häufig anhand von Taxonomien Hierarchien abgebildet, die eine Dimension vertikal in verschiedene Ebenen aufteilen. Ein Beispiel hierfür sind verdichtende Hierarchien (vgl. Böhnlein und Ulbrich-vom Ende, 2001, S. 34ff.; Oehler, 2006, S. 136f.; Totok, 2000, S. 92), die eine Kennzahl über mehrere Ebenen aufgespaltet darstellen und in einem Wert verdichten. Die Tiefe einer Dimension beschreibt wie viele Verdichtungsebenen existieren und wird auch als *Granularität* einer Dimension bezeichnet. Sie definiert die Anwendungsmöglichkeiten eines Modells dahingehend, dass sie für einen inhaltlichen Bereich, der durch eine Dimension repräsentiert wird, bestimmt, welche Intervalle zur Verfügung stehen und wie groß diese sind (vgl. Gluchowski et al., 2008, S. 151f.; Holthuis, 1999, S. 82ff.; Kemper et al., 2006, S. 167). In diesem Zusammenhang ist zu erwähnen, dass Kennzahlen als Dimensionen zu interpretieren sind und keine speziellen Modellelemente darstellen.

Die betrachteten Modelle bestehen aus mehreren Dimensionen. Die Kombination von genau einer Ausprägung je Modelldimension wird als *Zelle* bezeichnet. Im Umkehrschluss bedeutet dies, dass eine Dimensionsausprägung stellvertretend für die Menge von Zellen

betrachtet werden kann, an deren Definition sie durch Kombination mit Dimensionsausprägungen der restlichen Modelldimensionen beteiligt ist. Die Gesamtheit aller Zellen bildet wiederum ein Modell.

Die hierarchischen Zusammenhänge, die sich aus einer Taxonomie ergeben, bestimmen wiederum, wie die Werte der betroffenen Modellzellen berechnet werden. Für eine verdichtende Hierarchie – sinnvoll sind diese nur in Zusammenhang mit einer Kennzahl zu betrachten – wäre es z.B. denkbar, dass die Ausprägungen *a*, *b* und *c* einer Dimension in einer Dimensionsausprägung *d* als Summe verdichtet werden. Die eigentliche Definition der Summe als Additionsformel $d = a + b + c$ ist hiervon unabhängig und kann z.B. bei der Hierarchiedefinition mittels eines Summenzeichens (Σ) erfolgen.

Existiert in einem Modell jede Zelle, die durch Kombination von genau einer Ausprägung je vorhandener Modelldimension denkbar ist, wird ein Modell als homogen bezeichnet. Ohne explizite Definition von Einschränkungen der Zellmenge ist ein Modell in jedem Fall homogen. In inhomogenen[10], multidimensionalen Modellen ist dementsprechend definiert, welche Zellen ursprünglich denkbar, aber nicht im Modell vorhanden sind. Ein Grund für inhomogene Modelle sind unbalancierte Hierarchien. Diese entstehen, wenn einzelne Dimensionsausprägungen unterschiedlich tief hinsichtlich einer weiteren Dimension differenziert bzw. verdichtet werden können (vgl. Oehler, 2006, S. 138). So kann es beispielsweise vorkommen, dass in einem Vertriebsmodell für das Jahr *2012* die *Produkte A*, *B* und *C* in einer Dimension Region über die Dimensionsausprägungen *Nord*, *Ost* und *Süd* in einer Ausprägung *Gesamt* verdichtet werden, ein *Produkt D* jedoch hinsichtlich des Vertriebs nicht in einzelne Regionen unterschieden werden kann und somit eine Hierarchiestufe fehlt. Abbildung 3-1 visualisiert anhand eines Würfels ein inhomogenes, dreidimensionales Modell.

[10] In der Literatur wird in diesem Zusammenhang auch von heterogenen Strukturen gesprochen (vgl. Rieger, 1994, S. 118f.).

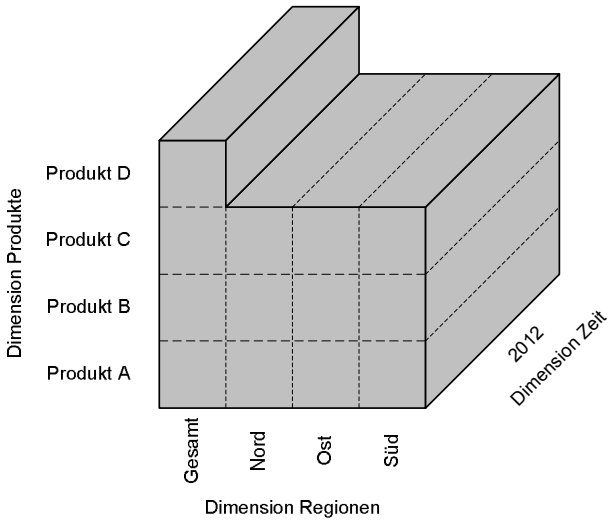


Abbildung 3-1: Inhomogener, dreidimensionaler Würfel

3.2.2 Formeln

Formeln werden in Modellen dazu benutzt Modellelemente, in der Regel Dimensionsausprägungen, miteinander in Beziehung zu setzen und dem Modell eine logische Komponente zu verleihen. Eine Möglichkeit zur Definition logischer Zusammenhänge sind Hierarchien, die eine Kennzahl anhand von Dimensionsausprägungen in unterschiedlichen Granularitäten darstellt. Die eigentliche Arithmetik eines Modells verbirgt sich hinter den Kennzahlendefinitionen, die auch als Formeln bezeichnet werden.

Es werden zwei Teile einer Formel unterschieden. Der erste Teil einer Formel bis zum Gleichheitszeichen wird *Bezugsteil* genannt und beschreibt, auf welche Zellen eines Modells die Formel angewandt wird. In einer Beispielformel $z = x + y$ wird der Bezugsteil durch die Variable z repräsentiert. Der Teil einer Formel, der dem Gleichheitszeichen folgt und die Logik definiert, wird *Logikteil* genannt. Im vorliegenden Beispiel wird er durch $x + y$ repräsentiert. Er beschreibt wie sich der Wert einer Zelle durch logische Verknüpfung weiterer Zellen zusammensetzt. In einem multidimensionalen Raum gibt es zwei Arten von Formeln, die sich in ihrer Wirkungsweise unterscheiden.

3.2.2.1 Zell-basierte Formeln

Zell-basierte Formeln sind dadurch gekennzeichnet, dass sie im Bezugsteil nur eine Modellzelle definieren. Im Logikteil werden weitere Modellzellen anhand von Referenzen auf Zell-IDs und logischen Operatoren (z.B. + oder *) derart miteinander in Beziehung gesetzt, dass sie einen logischen Zusammenanhang abbilden. Dies bedeutet, dass Formeln keine Bezüge zur Dimensions-Semantik in Form von inhaltlich verbundenen Beschreibungsob-

jekten aufweisen und damit keine Attribuierung stattfindet. Eine Zell-basierte Formel kann daher inhaltlich nur hinsichtlich der Logik, die sie repräsentiert, interpretiert werden.

Diese Art Formeln zu definieren wird in der Regel in Spreadsheet-Werkzeugen genutzt. Excel schließt z.B. bei der Definition einer Formel die Angabe eines Bezugsteils aus. Stattdessen wird lediglich der Logikteil einer Formel explizit für eine Zelle spezifiziert. Dies ermöglicht es dem Benutzer, sehr präzise alle logischen Zusammenhänge eines Modells zu definieren, führt jedoch im Umkehrschluss zu einem hohen Spezifikations- und Wartungsaufwand bei großen Modellen mit vielen Zellen, da jede Zelle einzeln spezifiziert werden muss. Die Formeln in Zellen, die durch eine identische Logik berechnet werden sollen, müssen redundant spezifiziert werden. Dies ist ein Grund für die in Kapitel 3.1 dargestellte Fehleranfälligkeit solcher Modelle. Ähnliche Effekte ergeben sich für What-If- oder Goal-Seeking Analysen (siehe Abschnitt 2.1.3) unter Anwendung von Modellen mit Zell-basierter Formeldefinition. Sollen beispielsweise die Auswirkungen einer geänderten Konstante in einer Umsatzformel über mehrere Perioden beobachtet werden, so müssen alle Modellzellen, welche mit der Umsatzformel belegt sind, manuell angepasst werden.

3.2.2.2 Vektor-basierte Formeln

Eine zweite Alternative stellen Vektor-basierte Formeln dar. Die Modelle der MDSS, für welche die konzeptuelle Modellierungsebene entworfen wird, nutzen diese Art der Formeldefinition. Eine Formel wird hierbei auf eine definierte Zellmenge angewandt. Je nach Definition kann diese Zellmenge eine oder mehrere Zellen umfassen. Die Definition einer Vektor-basierten Formel findet nicht auf Basis von Zell-IDs statt, sondern nutzt die Dimensionsausprägungen eines multidimensionalen Modells. Die Dimensionsausprägungen fungieren als Variablen einer Formel. Dadurch wird die Semantik einer Dimensionsausprägung sowie der Dimension, welcher die Ausprägung angehört, auf eine Formel übertragen. Im Gegensatz zu Zell-basierten Formeln repräsentieren Vektor-basierte Formeln daher eine inhaltliche Aussage, die über die Interpretation der logischen Zusammenhänge hinausgeht.

Der Bezugsteil einer Formel definiert die Zellmenge, auf welche eine Formel angewandt werden soll, durch Angabe von Dimensionsausprägungen (vgl. Abschnitt 3.2.1). Auf diese Weise wird der Modellierungsaufwand erheblich reduziert, da Mehrfachspezifikationen einer Formel entlang einer Dimension nicht nötig sind. Je Modelldimension kann im Bezugsteil höchstens eine Ausprägung angegeben werden. Durch Bilden der Schnittmenge aller Zellmengen, die durch die Dimensionsausprägungen repräsentiert werden, kann die

Zellmenge, auf die eine Formel angewandt wird, verringert werden. Ist im Bezugsteil einer Formel genau eine Dimensionsausprägung je Modelldimension vertreten, repräsentiert die Schnittmenge eine konkrete Modellzelle. Auf diese Weise kann eine Zell-basierte Formeldefinition imitiert werden.

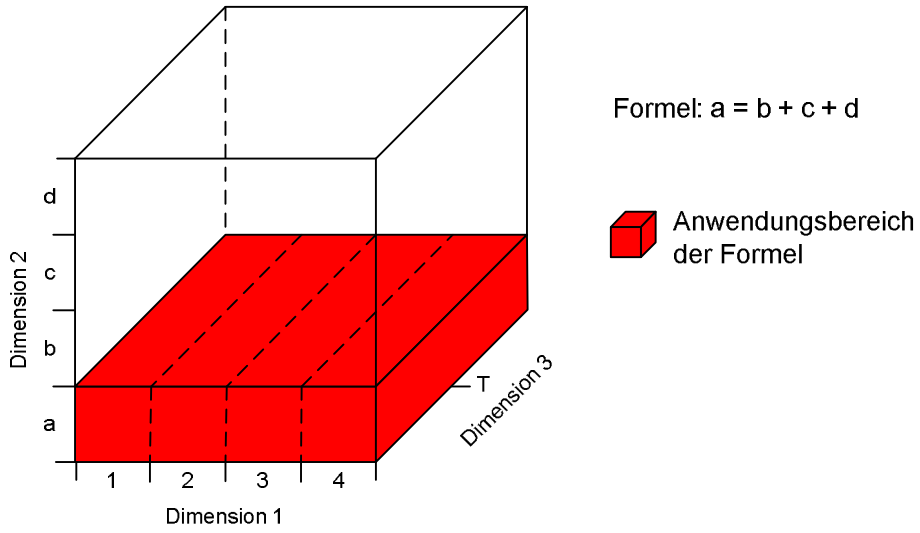


Abbildung 3-2: Eine Vektor-basierte Formel in einem multidimensionalen Modell

Abbildung 3-2 veranschaulicht die Wirkungsweise einer Vektor-basierten Formel innerhalb eines multidimensionalen Modells. Die Formel definiert mittels einer Aggregationslogik eine Summe in Dimension 2. Dadurch, dass der Bezugsteil der Formel nur aus Ausprägung a besteht, wird diese *vektoriell* auf alle Zellen des Modells angewandt, an denen a beteiligt ist. Bei einer Zell-basierten Formeldefinition müsste für alle Kombinationen der Ausprägungen 1, 2, 3 und 4 aus Dimension 1 mit der Ausprägung T aus Dimension 3 jeweils eine eigene Formel definiert werden.

Bei der Vektor-basierten Formeldefinition werden im Logikteil einer Formel – wie im Falle der Zell-basierten Definition – einzelne Modellzellen unter Nutzen logischer Operatoren miteinander in Beziehung gesetzt. Jeder Operand einer Formel kann wiederum aus maximal einer Ausprägung je Modelldimension bestehen. Im Beispielmodell in Abbildung 3-2 besteht jeder Operand aus genau einer Ausprägung von Dimension 2 (b, c und d). Die Formel a = b + c + d wird auf die vier markierten Modellzellen angewandt. Um festzustellen, welche Zelle z.B. im konkreten Fall der Zelle, die aus einer Kombination der Ausprägungen a, 1 und T besteht (siehe Abbildung 3-2: die Zelle „links unten“), durch den Operanden b der Formel repräsentiert wird, werden die nicht definierten Ausprägungen der übrigen Modelldimensionen entsprechend der Zelle ergänzt, auf welche die Formel angewandt wird. Das bedeutet in diesem Beispielfall, dass der Operand b die Zelle repräsen-

tiert, die aus den Dimensionsausprägungen b, 1 und T besteht (1 und T werden ergänzt, da keine Ausprägungen der Dimensionen 1 und 3 angegeben sind).

Der Bezugsteil einer Vektor-basierten Formel wird anhand von Dimensionsausprägungen des Modells definiert, die jeweils eine gewisse Zellmenge definieren, auf welche eine Formel angewandt werden soll. Es ist in diesem Zusammenhang möglich, dass sich die Bezugsteile mehrerer Formeln eines Modells überschneiden. Dies bedeutet, dass eine Zelle mit mehreren Formeln belegt sein kann bzw. sich mehrere Formeln in einer Zelle überschneiden können. Für die Anwendung eines Modells muss diese nicht eindeutige Spezifikation gelöst werden. Das heißt es muss sichergestellt sein, dass jede Zelle mit der Formel belegt ist, die im Sinne des Modellanwenders ist.

Die dargestellten Eigenschaften multidimensionaler Modelle sowie das Konzept der Vektor-basierten Formeldefinition sollen aufgrund des erheblich geringeren Modellierungsaufwands beim Erstellen einer konzeptuellen Modellierungsebene für MDSS berücksichtigt werden. Im folgenden Abschnitt 3.3 werden die generellen, charakteristischen Anforderungen einer konzeptuellen Modellierungsebene erläutert.

3.3 Anforderungen einer konzeptuellen Modellierungsebene

3.3.1 Freie Modellierung

Bei der Modellierung auf konzeptueller Ebene soll der Benutzer möglichst frei und unabhängig von proprietären Werkzeugkonzepten Modelle definieren können, damit das Abbilden realer Zusammenhänge stets im Fokus steht. Der Modellierer soll in keinem Fall in der Art und Weise beeinflusst werden, wie er einen Zusammenhang modelliert. Dies umfasst insbesondere das Zulassen alternativer Darstellungen einer Situation. Der Modellierer soll lediglich vom Modellierungszweck und den realen Gegebenheiten in Form von weiteren Modellformeln abhängig sein.

Dieser Aspekt stellt eine zentrale Eigenschaft konzeptueller Modelle dar (siehe Kapitel 2.2.2). Gabriel und Gluchowski führen in diesem Zusammenhang an, dass für das Verwenden von Modellen durch den Endnutzer ein intuitiver Zugang zur Modellierungsmethode notwendig ist (vgl. Gabriel und Gluchowski, 1998, S. 501).

3.3.2 Generische Dimensionalität

Die Forderung nach einer freien Modellierung drückt sich in der Forderung nach einer generischen Dimensionalität aus. Für den spezifischeren Teilbereich der OLAP-Systeme

existiert diese Forderung bereits (Regel 6) (vgl. Codd et al., 1993, S. 13ff.). Jede Dimension muss der Anforderung generischer Dimensionalität entsprechend dieselben strukturellen und funktionalen Möglichkeiten bereitstellen. Das Einführen bestimmter Dimensionstypen ist folglich zu vermeiden. Eng verknüpft mit dieser Regel ist die Forderung nach einer uneingeschränkten interdimensionalen Arithmetik (Regel 9) (vgl. Codd et al., 1993, S. 13ff.). Sie beschreibt, dass jegliche Kombination und Relation zwischen Dimensionen bzw. deren Ausprägungen zur Definition von logischen Zusammenhängen anhand von Formeln abbildbar sein muss.

3.3.3 Nichtprozeduralität

Die Forderung einer intuitiv zugänglichen Modellierungsmethode, die in engem Zusammenhang mit der klassischen DSS-Forderung zur Verringerung der Distanz zwischen Modellierer und Modellanwender steht (vgl. Power, 2002, S. 6; Turban et al., 2011a, S. 9), erfordert die Betrachtung der Reihenfolge der Formeln. In einem prozeduralen Modellierungsprozess wird dem Anwender die Aufgabe auferlegt für eine korrekte Berechnungsreihenfolge der Formeln zu sorgen. Nichtprozedurale Modellierungsprozesse ermöglichen es dem Modellierer hingegen, die logisch korrekte Berechnungsreihenfolge der Formeln in einem Modell außer Acht zu lassen und stattdessen sach-logisch zusammenhängende Einheiten aufeinander folgend zu spezifizieren (vgl. Rieger, 1994, S. 134). Dies erhöht die Modelltransparenz und sorgt für ein intuitives Verständnis des Modellierers (vgl. Rieger, 1994, S. 134f.).

Die Anforderung Nichtprozeduralität führt in einem multidimensionalen Modell mit Vektor-basierten Formeln zu Konstellationen, in denen die korrekte Berechnungsreihenfolge der Formeln schwer zu bestimmen ist. „Zum Beispiel müssen zur Umlage von Gemeinkosten auf Kostenstellen zuerst der Gesamtkostenwert der abgebenden Kostenstelle berechnet, die Summe der Verbrauchswerte aller empfangenden Kostenstellen bezüglich der Bezugsgröße ermittelt, danach die individuellen Verbrauchswerte zu dieser Summe in Beziehung gesetzt und mit dem Gesamtkostenwert multipliziert werden“ (Rieger, 1994, S. 135).

3.3.4 Simultanität

Eine weitere Anforderung befasst sich mit der Behandlung simultaner Berechnungsvorschriften. Simultanität drückt aus, dass zwei oder mehrere Formeln nicht mittels einer Modell-weiten Sortierung in ein prozedural lösbares Gleichungssystem transformiert werden können (vgl. Zwicker et al., 2001, S. 298f.). Dies führt dazu, dass zwei endogene Modell-

variablen eine Schleife bilden, die nicht mittels prozeduraler Rechenoperationen gelöst werden kann. Vielmehr müssen mittels algorithmischer Verfahren die Werte für die abhängigen Größen berechnet werden, so dass sie im Einklang mit den restlichen Bedingungen des Modells stehen. Es spielt bei einer Schleife keine Rolle, ob diese unmittelbar über zwei oder indirekt über mehrere Berechnungsvorschriften geschlossen wird. Die Möglichkeit simultane Beziehungen in einem Modell zu berücksichtigen, ist für die Modellierung auf konzeptueller Ebene eine weitere Anforderung. Es soll die Möglichkeit bestehen, simultane Beziehungen während der Modellierung automatisiert aufzudecken, damit der Modellersteller ggf. schon während des Modellierungsprozesses reagieren kann. Spiegelt die Spezifikation der Schleife nicht seine Modellierungsintention wider, kann er direkt entsprechende Anpassungen vornehmen.

3.3.5 Teilmengen-differenzierte Kennzahlenspezifikation

Abschnitt 3.2.1 erläuterte die Besonderheiten multidimensionaler Räume und stellte fest, dass Modelle nicht in jedem Fall homogene Gebilde sind, deren Dimensionen stets dieselbe Granularität aufweisen. Die Möglichkeit Teile bzw. Zellen von Modellen explizit ausschließen zu können ist zwar wünschenswert (vgl. Rieger, 1994, S. 118f.), stellt hier jedoch keine zentrale Anforderung dar. In Bezug auf homogene Strukturen muss gleichwohl die Möglichkeit bestehen *Teilmengen-differenzierte Kennzahlenspezifikationen* zu definieren. Dies bedeutet, dass eine Formel nicht auf alle Modellzellen angewandt wird, die aus dem Vektor-basierten Verständnis resultieren. Durch eine Teilmengen-differenzierte Kennzahlenspezifikation besteht die Möglichkeit Zellbereiche zu definieren, die explizit aus dem Bezugsteil einer Formel ausgeschlossen werden. Für einen Modellierungsprozess unter Berücksichtigung multidimensionaler Strukturen bedeutet dies eine erhöhte Flexibilität und eine Erhöhung von Modellierungskonstellationen, die anhand einer Modellierungssprache abgebildet werden können.

Zusammenfassend können eine freie Modellierung, generische Dimensionalität, Nichtprozeduralität, Simultanität und eine Teilmengen-differenzierte Kennzahlenspezifikation als Anforderungen an eine konzeptuelle Modellierungsebene für die betrachteten MDSS (siehe Kapitel 2.4) genannt werden. Abschnitt 3.4 untersucht nun bestehende konzeptuelle Modellierungsansätze und beurteilt deren Qualität in Bezug auf diese Anforderungen.

3.4 Bekannte konzeptuelle Modellierungsansätze

Die Modellierungsansätze stammen aus dem Bereich des Data Warehousing und befassen sich grundsätzlich mit der konzeptuellen, multidimensionalen Modellierung von Daten. Die Darstellung der Ansätze fokussiert lediglich Bestandteile, die im Rahmen dieser Arbeit relevant erscheinen. Allen Ansätzen ist gemein, dass sie auf einer abstrakten Meta-Ebene – oft als grafische Notation – die Möglichkeit einer konzeptuellen Modellierung schaffen, ohne den Fokus der Anwendbarkeit in Bezug auf eine konkrete Systemebene vermissen zu lassen. Die konkrete Auswahl der Modellierungsansätze erfolgt auf Basis von Forschungsarbeiten, die sich mit der konzeptuellen Modellierung aus Perspektive datengetriebener Systeme beschäftigen und eine möglichst große Adaptionswahrscheinlichkeit erhoffen lassen (Gabriel und Gluchowski, 1998; Hahne, 2010; Schelp, 2000).

3.4.1 Modellierungsaspekte in der Datenbewirtschaftung

Das Hauptaugenmerk der Datenbewirtschaftung liegt auf der Konsolidierung und zeitspezifischen Organisation großer Datenmengen, z.B. mittels Data Warehouses (vgl. March und Hevner, 2007, S. 1031). Datenbewirtschaftungsapplikationen beinhalten häufig eine Modellierungskomponente, die nicht als solche bezeichnet wird. Diese verbirgt sich hinter der Transformationskomponente von Datenbewirtschaftungs- bzw. ETL-Prozessen. ETL steht für Extraktion, Transformation und Laden und dient dem Befüllen eines DWH mit Daten, i.d.R. aus operativen Systemen (vgl. Kimball und Ross, 2002, S. 8f.). Extraktion beschreibt das Übertragen von Daten aus einer externen Datenquelle in einen Arbeitsbereich. In diesem Bereich findet die Transformation statt, welche sämtliche Umwandlungsoperationen repräsentiert, die zwischen der Extraktion und dem Laden der Daten in ein DWH stattfinden. Diese Operationen können grob in zwei Schritte unterteilt werden. Verknüpfungen sind notwendig, wenn einzelne Informationsobjekte in Quellsystemen getrennt vorliegen, diese im DWH jedoch innerhalb eines Objekts dargestellt werden sollen und mittels Bereinigungsmaßnahmen werden syntaktische und semantische Fehler innerhalb der Daten entfernt (vgl. Gluchowski et al., 2008, S. 137f.). Laden umschreibt die abschließende Speicherung der im Arbeitsbereich aufbereiteten Daten in ein Data Warehouse (vgl. Bauer und Günzel, 2004, S. 49ff.).

Die Modellelemente werden in einem Data Warehouse durch sogenannte Fakten- und Dimensionselemente repräsentiert. Faktenelemente repräsentieren Kennzahlen, die entlang von Dimensionselementen angeordnet werden. In einem ETL-Prozess können Faktenelemente miteinander verknüpft und auf diese Weise logische Zusammenhänge abgebildet

werden. Zumindest für Dokumentationszwecke werden Modellinformationen in Form von Meta-Daten auch heute schon dem Anwender in solchen Systemen zur Verfügung gestellt (vgl. Devlin, 1997, S. 52ff.; Schultewolter, 2010a, S. 4).

Anhand von ETL-Prozessen lässt sich grundsätzlich jeder logische Zusammenhang darstellen. Durch die charakteristische Prozeduralität fehlen jedoch viele Merkmale der effizienten und vor allem der freien Modellierung. Darüber hinaus bleiben simultane Formeln gänzlich unberücksichtigt, so dass insgesamt die Modellierungsaspekte in der Datenbewirtschaftung den Qualitätsanforderungen einer konzeptuellen Modellierungsebene für MDSS nicht entsprechen.

3.4.2 Semantisches Data-Warehouse Modell

Das semantische Data Warehouse Modell (SDWM) beschreibt eine Möglichkeit zur konzeptuellen Modellierung multidimensionaler Datenstrukturen. Die folgenden Ausführungen basieren auf der Arbeit von Böhnlein und Ulbrich-vom-Ende (vgl. Böhnlein und Ulbrich-vom Ende, 2001). Der Ansatz nutzt ein Konzept unterschiedlicher Sichten und unterscheidet grundsätzlich quantitative (*Kennzahlen*) und qualitative Aspekte (*Dimensionen*).

Eine Dimension stellt im SDWM einen Blickwinkel auf Kennzahlen dar. Die folgende Abbildung stellt das Meta-Modell einer Dimensionssicht dar:

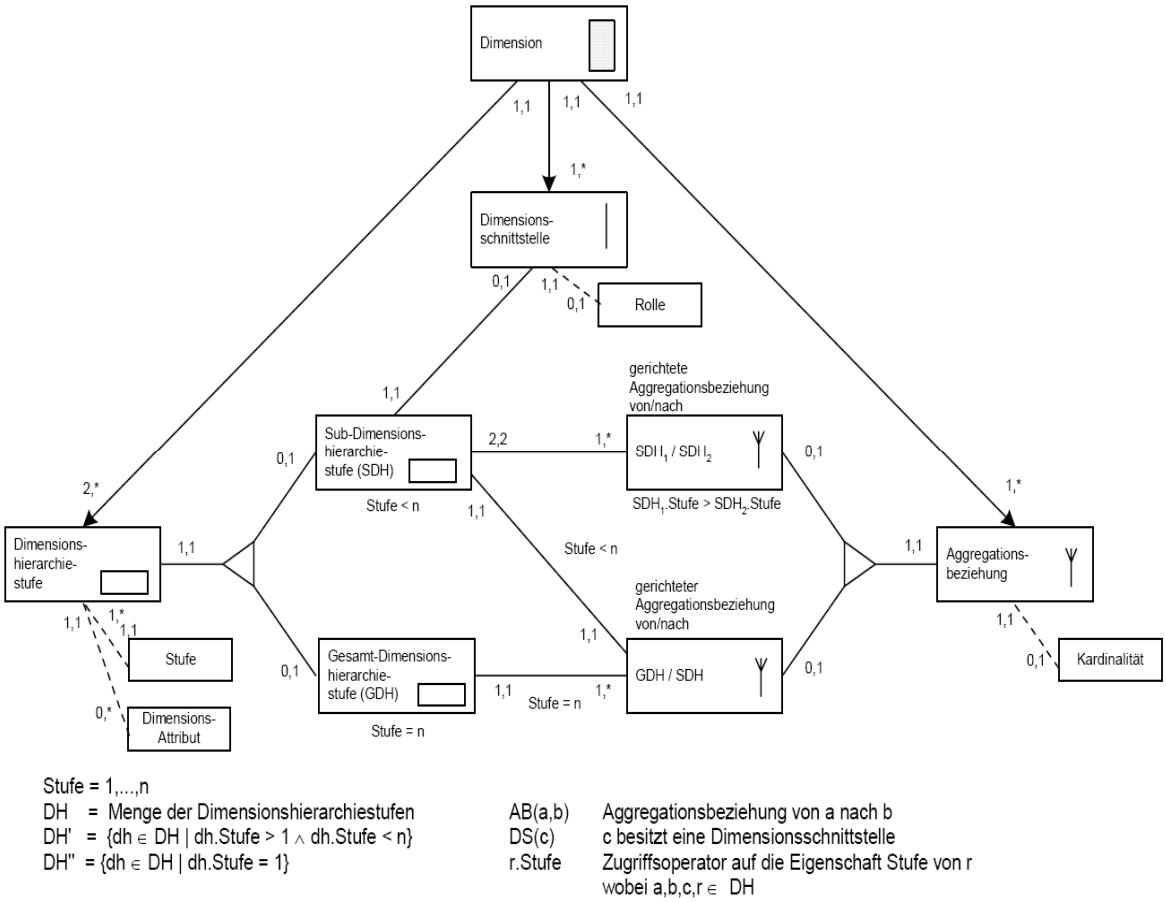


Abbildung 3-3: Meta-Modell einer Dimensionssicht

(Böhnlein und Ulbrich-vom Ende, 2001, S. 35)

Eine *Dimensionssicht* setzt sich aus zwei oder mehreren Dimensionsausprägungen zusammen. In Abbildung 3-3 werden diese durch *Dimensionshierarchiestufen* repräsentiert, die jeweils durch *Aggregationsbeziehungen* miteinander in Relation gesetzt werden. Dimensionsausprägungen sind immer einer Hierarchiestufe zugeordnet, so dass aufsteigend eine Verdichtung von Daten abgebildet wird. Eine *Dimensionsschnittstelle* verbindet Dimensionen mit einer *Basiskennzahl* und stellt auf diesem Weg die Wiederverwendbarkeit von Dimensionen sicher.

Einer Basiskennzahl liegt das Verständnis von absoluten Zahlen nach Staehle zugrunde (vgl. Staehle, 1969, S. 50). Die Dimensionsschnittstelle definiert die Additivität einer Basiskennzahl in Bezug auf eine Dimension durch ein Attribut Aggregierbarkeit, das die drei Ausprägungen *additiv*, *semiadditiv* und *nicht-additiv* annehmen kann (vgl. Kimball und Ross, 2002, S. 17). Über dieses Konstrukt wird die Verdichtung von Datenwerten in einer Hierarchie beschrieben. Additive Kennzahlen lassen sich ohne Einschränkung summieren, semiadditive Kennzahlen können lediglich entlang einiger Modelldimensionen addiert

werden und für nicht-additive Kennzahlen können keine Summen dargestellt werden. Abbildung 3-4 veranschaulicht das Meta-Modell einer Basiskennzahl:

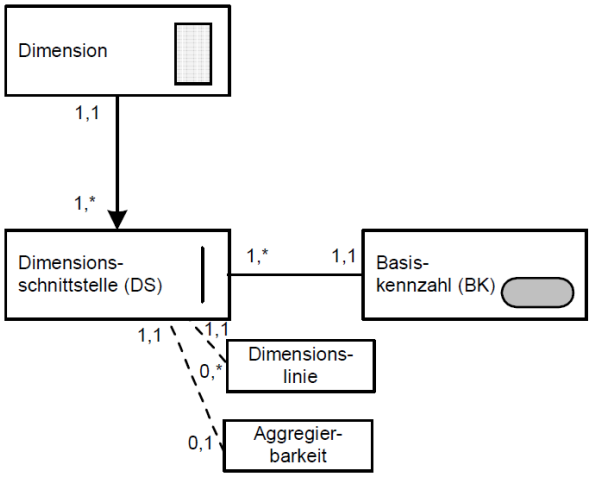


Abbildung 3-4: Meta-Modell für die Sicht auf eine Basiskennzahl

(Böhnlein und Ulbrich-vom Ende, 2001, S. 41)

Gemeinsam mit einer *Dimensionslinie*, welche die Granularität einer Basiskennzahl in Form der niedrigsten anzuwendenden Hierarchiestufe beschreibt, wird so das Verhalten einer Basiskennzahl in Bezug auf eine verdichtende Hierarchie festgelegt.

Kennzahlen, die über eine *Berechnungsvorschrift*[11] dargestellt werden, werden nicht in der Sicht auf eine Basiskennzahl definiert, sondern in einer Sicht auf ein *Kennzahlensystem*. Solche *abgeleiteten Kennzahlen* werden über eine *gerichtete Kennzahlenbeziehung* gebildet, die eine Basis- und eine abgeleitete oder aber zwei abgeleitete Kennzahlen miteinander in Beziehung setzen kann (vgl. Böhnlein und Ulbrich-vom Ende, 2001, S. 44). Für eine abgeleitete Kennzahl können drei unterschiedliche Verhältniszahlen unterschieden werden: Gliederungs-, Beziehungs- und Indexzahlen (vgl. Staehle, 1967, S. 64f.). Ausgehend von einer (abgeleiteten) Kennzahl und einer Kennzahlenbeziehung veranschaulicht Abbildung 3-5 die Elemente sowie das Zusammenspiel dieser innerhalb eines Kennzahlensystems.

[11] Berechnungsvorschriften beinhalten in diesem Zusammenhang auch einfache Verhältnisgrößen, die eine Relation zweier Basiskennzahlen ausdrücken (vgl. Böhnlein und Ulbrich-vom Ende, 2001, S. 43).

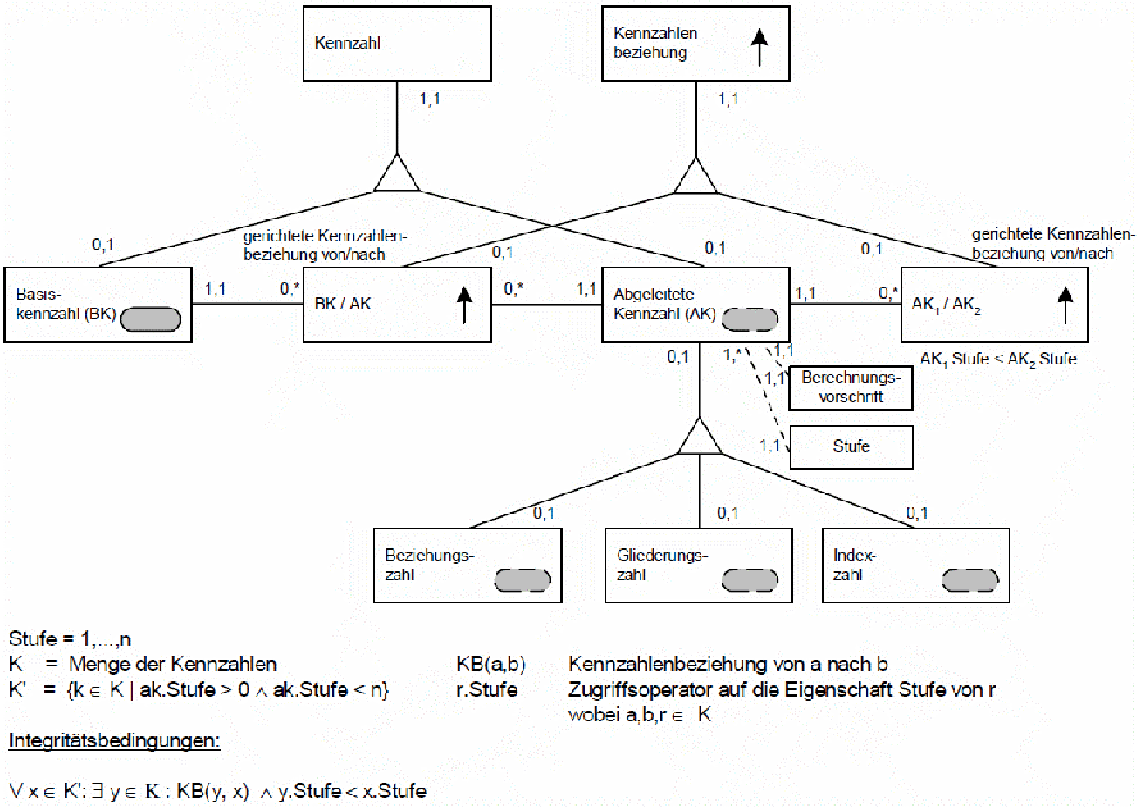


Abbildung 3-5: Meta-Modell für das Kennzahlensystem

(Böhnlein und Ulbrich-vom Ende, 2001, S. 44)

Über eine mehrstufige Verknüpfung von Basis- bzw. abgeleiteten Kennzahlen kann somit eine komplexe Logik schrittweise definiert werden (vgl. Böhnlein und Ulbrich-vom Ende, 2001, S. 33ff.; Schultewolter, 2009, S. 27ff.).

Damit die Nutzung von SDWM für die Umsetzung der im Rahmen dieser Arbeit zu erstellenden konzeptuellen Modellierungsebene für MDSS sinnvoll erscheinen könnte, müssten einige Anpassungen berücksichtigt werden. Logische Zusammenhänge werden immer ausgehend von einer Basiskennzahl gebildet, welche über eine Dimensionsschnittstelle mit einer Dimension verbunden ist. Es ist daher in jedem Fall notwendig eine komplexe Rechnung in mehrere Teilschritte zu zerlegen, da eine abgeleitete Kennzahl zwar über eine Berechnungsvorschrift, jedoch nicht über eine Dimensionsschnittstelle verfügt. Dieser Zusammenhang führt des Weiteren dazu, dass eine Kennzahl nicht für einen Ausschnitt des Modells definiert werden kann (Teilmengen-differenzierte Kennzahlenspezifikation) (vgl. Schultewolter, 2010a, S. 4). Es ist weder möglich, für eine Kennzahl einen korrespondierenden Geltungsbereich in Bezug auf Ausprägungen einer weiteren Dimension zu definieren, noch in einer Formel auf andere Dimensionen zu verweisen, da eine abgeleitete Kennzahl über keine Dimensionsschnittstelle verfügt. Des Weiteren impliziert die zu defi-

nierende Reihenfolge von Basis- und abgeleiteten Kennzahlen eine strenge Prozeduralität des Modellierungsprozesses. Simultane Referenzen sind daher im SDWM nur schwer zu berücksichtigen. Die Orientierung an Hierarchien erschwert die Entdeckung solcher Zirkelbezüge, da die Hierarchiestufen gesondert berücksichtigt werden müssten. Aus diesen Gründen und weil durch das Sichtenkonzept eine bestimmte Reihenfolge im Modellierungsprozess vorgeschrieben ist, welche den Benutzer in der Freiheit der Modellierung einschränkt, wird eine prinzipiell denkbare SDWM-Erweiterung hier nicht weiter verfolgt.

3.4.3 Multidimensionales Entity-Relationship Modell

Das multidimensionale Entity-Relationship Modell (ME/R Modell) erweitert das ursprüngliche Entity-Relationship Modell nach Chen[12] um die Möglichkeiten zur besonderen Berücksichtigung multidimensionaler Datenstrukturen (vgl. Sapia et al., 1999, S. 109). Neben den üblichen Größen der ER-Modellierung, wie z.B. Entitäten und Relationen, werden im ME/R Modell der spezielle Entitätstyp *dimension level* (Dimensionsausprägung) und die zwei speziellen Relationen *fact* und *classification* zur Verbindung von dimension levels eingeführt. Mittels einer Relation des Typs classification (vgl. Abbildung 3-6: *binary relationship set*) werden qualifizierende Daten modelliert. Mit dieser Relation lassen sich z.B. dimension levels zu einer Hierarchie kombinieren, wohingegen Relationen vom Typ fact Kennzahlen und dementsprechend quantitative Daten repräsentieren.

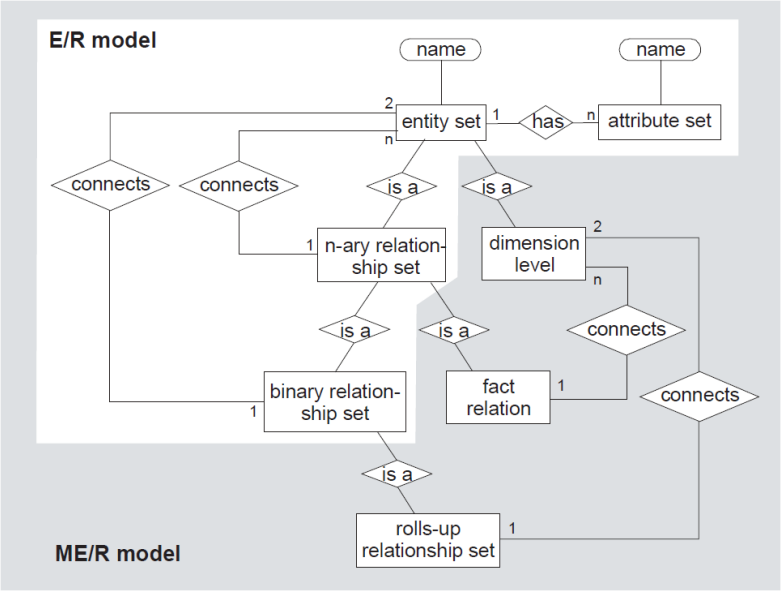


Abbildung 3-6: Metamodell des ME/R-Modells

(Sapia et al., 1999, S. 110)

[12]Nähere Informationen zum ER-Modell finden sich in der Erstpublikation von Chen (Chen, 1976).

Abbildung 3-6 veranschaulicht die Erweiterungen des ME/R- im Vergleich zum ursprünglichen E/R-Modell. Entitäten können sowohl Relationen als auch Dimensionsausprägungen darstellen (vgl. Sapia et al., 1999, S. 109ff.).

Beim Betrachten des Metamodells fällt auf, dass eine gesonderte Modellierung von Dimensionen in diesem Ansatz fehlt. Die Autoren begründen dies damit, dass mittels einer Relation des Typs classification implizit die Information vorliegt, welche Ausprägungen (dimension levels) zu einer Dimension gehören (vgl. Sapia et al., 1999, S. 111). Dies führt jedoch in Bezug auf eine konzeptuelle Modellierungsebene für MDSS dazu, dass sowohl ein ME/R-Modell an sich, als auch jede Berechnungsvorschrift innerhalb des Modells in jedem Fall homogen im Sinne der in Abschnitt 3.2.1 dargestellten Ausführungen ist. Das heißt, dass der Bezugsteil einer Formel nicht eingeschränkt werden kann. Es ist ebenfalls nicht möglich die Information zu hinterlegen, dass bezüglich bestimmter Dimensionsausprägungen des Modells eine Kennzahl differenziert berechnet werden soll (Teilmengendifferenzierte Kennzahlenspezifikation).

Eine weitere Einschränkung ergibt sich in Bezug auf das Formulieren logischer Zusammenhänge und Rechenvorschriften, welche in der Datenmodellierung i.d.R. eine untergeordnete Rolle spielen. Eine Erweiterung des Modells oder eine gemeinsame Betrachtung von ETL-Prozessen und ME/R-Modell könnte diese Schwäche unter Umständen beseitigen. Durch dieses Vorgehen ergibt sich jedoch ein prozeduraler Modellierungsprozess, da die Reihenfolge von ETL-Prozessen ein charakteristisches Merkmal dieses Ansatzes darstellt. Eine Möglichkeit zur Betrachtung simultaner Strukturen schon während der Modellierung ist durch diese Prozeduralität ebenfalls nicht vorhanden. Die Eignung des ME/R-Modells für die Implementierung einer konzeptuellen Modellierungsebene für MDSS ist daher als nicht zielführend zu bewerten.

3.4.4 Object Oriented Multidimensional Data Model

Das Object Oriented Multidimensional Data Model (OOMD) wurde für die Modellierung von OLAP-Anwendungen entwickelt. Die Ausführungen basieren auf der Arbeit von Nguyen et al. (vgl. Nguyen et al., 2000). Konstrukte, die für die Modellierung genutzt werden, sind *Dimensionen*, *Kennzahlen* und *Datenwürfel*.

Eine Dimension besteht aus genau einer *hierarchical domain* (hierarchischer Wirkungsraum). Dieser Wirkungsraum ist wiederum aus mehreren *dimension members* (Dimensionsausprägungen) aufgebaut, die jeweils einzelnen *levels* (Hierarchiestufen) zugeordnet werden und unterschiedlichen Granularitäten entsprechen. Ein *dimension schema* (Dimen-

sionsschema) entspricht diesbezüglich einer partiell geordneten Folge von Hierarchiestufen. Dieses Konstrukt wird dafür genutzt, unbalancierte Hierarchien zu modellieren. Eine Hierarchie besteht allgemein immer aus einem Dimensionsschema und einem Pfad. Abbildung 3-7 illustriert das Vorliegen einer unbalancierten Zeit-Hierarchie. Einzelne Tage sind Wochen und Monaten, die derselben Hierarchiestufe angehören, zugeordnet. Monate werden wiederum Quartalen und diese letztendlich Jahren untergeordnet, während Wochen direkt Jahren untergeordnet werden. Ein *dimension path* (Dimensionspfad) stellt eine linear geordnete Liste von Hierarchiestufen dar. Auf dem Pfad, der Wochen beinhaltet, findet sich dementsprechend eine Hierarchiestufe weniger, als im Pfad, der Monate berücksichtigt.

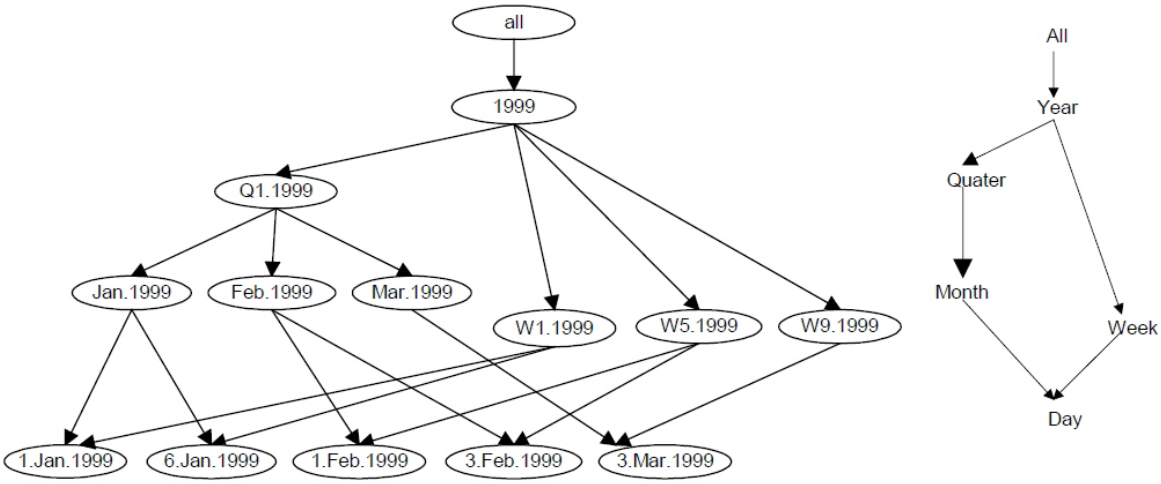


Abbildung 3-7: Unbalancierte und multiple hierarchische Strukturen in MDM

(in Anlehnung an: Nguyen et al., 2000, S. 70)

Kennzahlen bestehen im OOMD aus *facts* (Fakten) und *operations* (Rechenoperationen). Eine Rechenoperation kann aus typischen Aggregationsfunktionen (z.B. sum, count oder max) oder einer sog. Composite-Funktion bestehen. Diese wird benötigt, wenn Kennzahlen nicht automatisch für höhere Hierarchiestufen durch Aggregationsoperationen berechnet werden können. Ein typisches Beispiel hierfür sind Raten und Quoten, deren Werte auf höheren Aggregationsebenen häufig mittels gewichteter Durchschnitte berechnet werden.

Datenwürfel sind das dritte und letzte Konstrukt im OOMD. Ein Würfel besteht aus Dimensionsschemata und Kennzahlen (bzw. Kennzahlendaten), vereint also diese beiden Konstrukte. Die Schnittpunkte von Dimensionsausprägungen unterschiedlicher Dimensionen werden auch hier als Zellen bezeichnet. OOMD-Modelle sind immer homogen, da entsprechende Zellen nicht anhand eines Modellierungskonstrukts ausgeschlossen werden

können. Für einen Würfel lassen sich *GroupBy*, *Roll-Up* sowie *Drill-Down* Operatoren definieren. Ein GroupBy-Operator besteht aus einem *Gname* (Name), einem *GSchema* sowie einem Wirkungsbereich. Ein *GSchema* gibt an, welchen *levels* (Hierarchiestufen) die Kennzahlendaten im Würfel zuzuordnen sind. Dies geschieht auf Zellbasis, so dass jede Zelle einem Level in Bezug auf eine Kennzahl zugeordnet wird. Der Wirkungsbereich gibt die Dimensionsausprägungen an, die einen multidimensionalen Raum definieren, auf welchen der GroupBy-Operator angewandt werden soll. Roll-Up und Drill-Down Operatoren werden lediglich zur Navigation im Würfel genutzt und müssen nicht gesondert definiert werden.

Aus der Beschreibung der drei Hauptbestandteile wird ersichtlich, dass mittels OOMD definierte Modelle eine klare Trennung in Struktur (Dimensionen), Daten/Kennzahlen und Anwendung (Würfel) aufweisen. Diese Einteilung unterstützt einerseits den Benutzer beim Überblicken des Modellierungsprozesses, schränkt ihn jedoch andererseits auch in seiner Modellierungsfreiheit ein, da Kennzahlen im OOMD nicht als qualifizierende Dimension oder Dimensionsausprägung betrachtet werden können und somit die Anforderungen generischer Dimensionalität nicht erfüllt sind. OOMD-Modelle sind zwar strukturell immer homogen definiert, es besteht keine Möglichkeit bestimmte Zellen nicht zu betrachten bzw. sie gar nicht erst zu berücksichtigen. Es besteht jedoch die Möglichkeit, einzelnen Zellen keine Kennzahlenwerte bzw. Daten in einem Würfel zuzuweisen (GSchema) und somit eine Teilmengen-differenzierte Kennzahlenspezifikation zumindest in Ansätzen umzusetzen. Kritisch ist die Tatsache zu beurteilen, dass Formeln bzw. logische Zusammenhänge nur auf Basis von Hierarchien definiert werden können. Komplexe Rechenvorschriften auf unterster Ebene einer Hierarchie werden auf die vorhandenen Daten abgewälzt, so dass solche Logiken schon im Vorfeld des OOMD stattfinden müssten.

Abschließend ist zu erwähnen, dass auch in diesem Modellierungsansatz Zirkelschlüsse nicht gesondert berücksichtigt werden. Es besteht keine Möglichkeit sie im Modell aufzudecken, und da etwaige komplexe Rechenvorschriften nicht in OOMD definiert werden, sondern nur implizit in den Daten vorhanden sein können, ist ein automatisiertes Erkennen solcher Zirkelschlüsse nahezu ausgeschlossen. Für die Umsetzung einer konzeptuellen Modellierungsebene für MDSS eignet sich das OOMD dementsprechend nicht.

3.4.5 Dimensional Fact Model

Dimensional Fact Model (DFM) ist ein halbautomatisierter konzeptueller Modellierungsansatz für DWH-Systeme, der eine Grundlage für die Modellierung unternehmensweiter

DWH-Systeme darstellt. Die Beschreibung dieser Modellierungsmethode erfolgt auf Basis der Arbeiten von Golfarelli et al. (vgl. Golfarelli et al., 1998a, 1998b).

Ein DFM besteht aus sog. *Fakten-Schemata*, welche wiederum aus *Fakten*, *Dimensionen* und *Hierarchien* aufgebaut sind. Ein Faktum beschreibt eine Größe, die im DWH-System näher betrachtet, ausgewertet und interpretiert werden soll. Es besitzt typischerweise ein oder mehrere numerische Attribute, mittels derer eine quantitative Aussage hinsichtlich unterschiedlicher Perspektiven gemacht werden kann. Eine Dimension bestimmt die Granularität einer Perspektive auf die betrachteten Fakten, d.h. wie detailliert Fakten in Bezug auf eine Thematik unterschieden werden können. Eine Hierarchie gibt an, inwiefern und mittels welcher Logik Fakten entlang einer Dimension aggregiert werden können. Abbildung 3-8 zeigt beispielhaft ein dreidimensionales Fakt-Schema im DFM für das Faktum SALE mit den Attributen *quantity sold*, *returns*, *number of customers* und *inventory level* sowie den Dimensionen *product*, *store* und *week*. Die Äste, welche den Dimensionsknoten folgen, beschreiben Dimensionsausprägungen, die Hierarchien bilden. Dimensionsausprägungen werden durch einen weißen Punkt gekennzeichnet. Äste, die nicht durch einen weißen Punkt gekennzeichnet sind, stellen Attribute dar, welche die Dimension näher beschreiben und sind nicht als Dimensionsausprägung zu interpretieren. In Abbildung 3-8 ist dies z.B. das Attribut *size* der Dimension *product*.

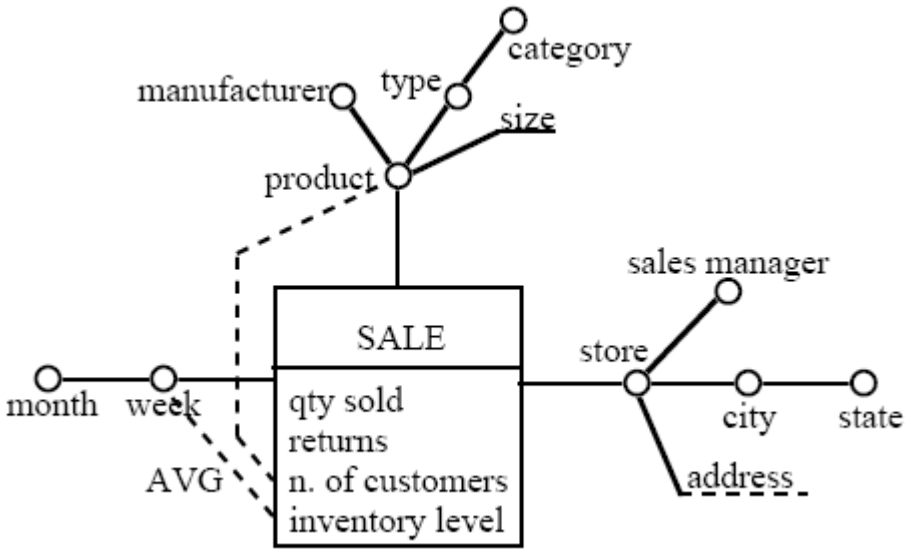


Abbildung 3-8: Ein dreidimensionales Fakt-Schema

(Golfarelli et al., 1998a, S. 336)

Wie schon in den zuvor beschriebenen multidimensionalen Modellierungsansätzen repräsentiert ein Faktum im DFM eine n-zu-n Beziehung zwischen Dimensionen. Diese Bezie-

hungen können wiederum als Zellen des multidimensionalen Raumes interpretiert werden. Eine Dimension kann im DFM als optional gekennzeichnet werden. Dies führt dazu, dass Zellen des Raumes nicht in jedem Fall betrachtet werden müssen und so die Möglichkeit besteht, heterogene Strukturen zu modellieren.

Ein weiterer zentraler Punkt ist die standardmäßige Annahme, dass sämtliche Attribute eines Fakt-Schemas uneingeschränkt *additiv* sind. Dies bedeutet im DFM, dass die numerischen Werte entlang aller Hierarchien summiert werden können, ohne dass inhaltliche bzw. logische Dissonanzen auftreten. Attribute, die mindestens hinsichtlich einer weiteren Dimension nicht summiert werden können, werden als *semiadditiv* und Attribute, die hinsichtlich keiner vorhandenen Dimension summiert werden können, als *nicht-additiv* bezeichnet. Es besteht für semi- oder nicht-additive Attribute jedoch häufig die Möglichkeit, die Werte auf höheren Hierarchieebenen mittels einer anderen Aggregationsfunktion, z.B. einem Durchschnitt, zu berechnen. Dies wird durch eine Verbindungslinie zwischen dem Attribut und der Dimensionsausprägung gekennzeichnet, neben welcher die Aggregationsfunktion steht, z.B. AVG (Durchschnitt) für die Ausprägung *inventory level* auf der Hierarchieebene *week* der Zeitdimension in Abbildung 3-8. Fehlt die Aggregationsfunktion, wird mittels der Verbindungslinie definiert, dass das Attribut nicht über die Hierarchie hinweg betrachtet wird und somit auch keine Berechnung der entsprechenden Werte für die unterschiedlichen Granularitäten erfolgt (vgl. Golfarelli et al., 1998a, S. 335ff., 1998b, S. 218ff.).

Wie auch schon die bereits dargestellten Modellierungsansätze bietet das DFM ebenfalls nur eine eingeschränkte Möglichkeit zur expliziten Formulierung logischer Zusammenhänge, da sich diese auf Hierarchien beschränken. Eine Erweiterung des DFM um diese Möglichkeit bzw. eine Anpassung des Konzepts erscheint nicht sinnvoll, da das grundlegende Konzept zur Modellierung von Hierarchien Beziehungen zwischen Dimensionsausprägungen unterschiedlicher Ebenen vorsieht, so dass letztendlich eine Baumstruktur entsteht. Logische Formeln mit Beteiligung von Ausprägungen derselben Hierarchiestufe oder unterschiedlicher Dimensionen werden grundsätzlich nicht adressiert und letztendlich würde nur die Modellierung struktureller Modellkomponenten übrig bleiben. Des Weiteren können im DFM simultane Beziehungen nur sehr schwierig ermittelt werden, da auf E/R-Modellen aufgesetzt wird. Die Beziehungen in einem E/R-Modell werden nur deskriptiv beschrieben und es ist insbesondere schwierig zu ermitteln, ob zwei Beziehungen zur selben Zeit eine Entität bzw. Modellvariable definieren. Das folgende Beispiel veranschaulicht dieses Problem: Ein Autor verfasst ein Buch, dieses wird von einem Verlag verlegt,

der wiederum den Autor bezahlt. Entitäten in diesem Beispiel sind *Autor*, *Buch* und *Verlag*, Beziehungen sind *verfasst*, *wird verlegt von* und *bezahlt*. Rein formal könnte dies eine simultane Beziehung darstellen. Da ein Buch jedoch in einem ersten Schritt verfasst und erst im Anschluss verlegt werden kann, ist diese Beziehung nicht simultan, sondern sequentiell. Das Fazit lautet daher, dass die Qualitätsanforderungen einer konzeptuellen Modellierung für MDSS im DFM ebenfalls unzureichend bedient werden.

3.4.6 Multidimensional Data Model

Beim Multidimensional Data Model (MD) handelt es sich um eine weitere konzeptuelle Modellierungsmethode, die für die Modellierung von OLAP-Systemen entworfen wurde und dem DFM ähnelt (vgl. Cabibbo und Torlone, 1998a, S. 319f., 1998b, S. 183f.). Neben einer grafischen Modellierungsnotation besteht es aus einem Vorgehensmodell, mittels dem, alternativ zu einer Modellierung von Grund auf (analog zum DFM), vorhandene E/R-Modelle als Basis für mehrdimensionale Modelle dienen (vgl. Schelp, 2000, S. 198). Die folgende Beschreibung des Ansatzes basiert auf den beiden Veröffentlichungen von Cabibbo und Torlone (Cabibbo und Torlone, 1998a, 1998b).

Die Hauptkonstrukte in MD sind Dimensionen (*dimensions*) und F-Tabellen (*f-tables*). Dimensionen können als syntaktische Kategorien im Sinne von Perspektiven, die unterschiedliche Blickwinkel auf Informationen bieten, verstanden werden. Jede Dimension besteht aus einer hierarchischen Anordnung von Dimensionsausprägungen (*levels*), wodurch sich unterschiedliche Granularitäten definieren lassen. Eine Dimensionsausprägung kann durch Beschreibungen (*descriptions*) genauer spezifiziert werden. Innerhalb der hierarchischen Anordnung wird durch *Roll-Up* Funktionen definiert, wie sich die einzelnen Ausprägungen unterschiedlicher Hierarchiestufen voneinander ableiten. F-Tabellen ordnen einer Kombination von Dimensionsausprägungen unterschiedlicher Dimensionen (Zellen) die Werte der betrachteten betriebswirtschaftlichen Kennzahlen (*measures*) zu.

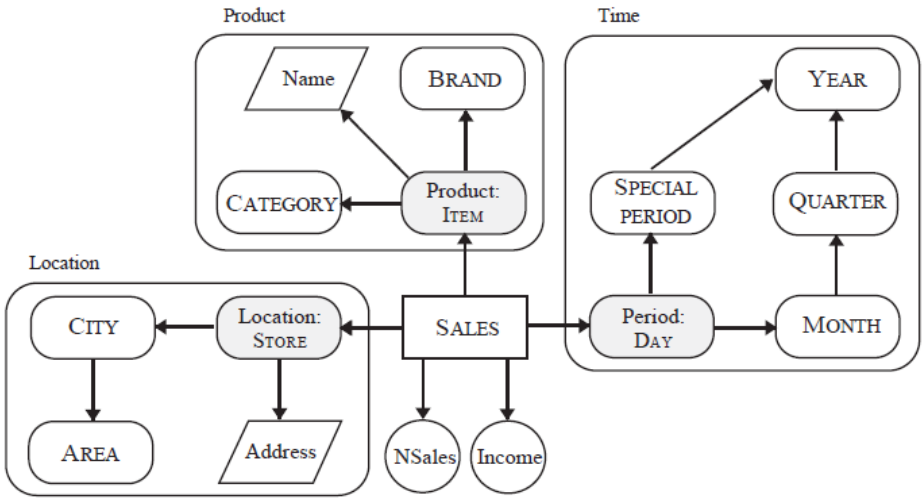


Abbildung 3-9: Graphische Repräsentation einer F-Tabelle in MD

(Cabibbo und Torlone, 1998c, S. 76)

Abbildung 3-9 zeigt ein mit dem MD definiertes Modell, das drei Dimensionen (Product, Time und Location) enthält. Eine Dimension wird durch das Zusammenführen von Dimensionsausprägungen in einem eingekreisten Bereich dargestellt. Die Ausprägungen sind innerhalb einer Dimension durch Pfeile miteinander verbunden, so dass über Roll-Up Funktionen ein Hierarchiepfad entsteht.

Im Zentrum der Abbildung steht ein F-Knoten *SALES*, der aus zwei Kennzahlen (*NSales* und *Income*) besteht. Die resultierende F-Tabelle kommt durch die Verbindungen des F-Knoten zu den einzelnen Dimensionen zustande und weist einer Kombination von je einer Ausprägung pro Dimension und einer Kennzahl den entsprechenden Wert zu. Ist ein Eintrag in einer Dimension durch ein Parallelogramm dargestellt (siehe *Name* in Abbildung 3-9), entspricht dieser Eintrag einer Beschreibung. Für die Berechnung der Werte der F-Tabelle kann der Benutzer generell sämtliche Grundrechenarten und Kennzahlen zu komplexen Formeln zusammenführen.

Die Berechnungsvorschriften können nur anhand der Kennzahlen eines F-Knoten definiert werden. Dies schließt dementsprechend eine Teilmengen-differenzierte Kennzahlenspezifikation aus. Des Weiteren führt die spezielle Behandlung von Kennzahlen in F-Tabellen dazu, dass Kennzahlen selbst nicht als qualifizierende Dimension bzw. Dimensionsausprägungen betrachtet werden. Der Benutzer muss bei der Modellierung dementsprechend entscheiden, ob ein Element durch eine Berechnungsvorschrift in Form einer Interaktion von Kennzahlen definiert werden soll oder ob es eine Dimensionsausprägung repräsentiert.

Durch das sequentielle Vorgehen bei der Modellierung im MD würde der Anwender durch diese Methode in der freien Modellierung eingeschränkt werden. Da die Definition von Berechnungsvorschriften wiederum auf bestimmte Modellelemente beschränkt ist und die Berücksichtigung von simultanen Bezügen durch die Möglichkeit auf einem existierenden E/R-Modell aufzusetzen stark begrenzt ist, ist eine Erweiterung oder Änderung der MD-Methode im Kontext dieser Arbeit nicht sinnvoll.

3.4.7 ADAPT

Das Akronym ADAPT steht *für Application Design for Analytical Processing Technologies* und stellt eine 1996 von Bulos entwickelte Methode zur grafischen Abbildung multidimensionaler Datenstrukturen dar, die insbesondere die analytischen Eigenschaften multidimensionaler Modelle in OLAP- und DWH-Systemen akzentuiert (vgl. Bulos, 1998; Schelp, 2000, S. 182).

Die zentralen Bestandteile des Ansatzes umfassen *Würfel*, *Dimensionen* und *Ableitungs-* bzw. *Berechnungsformeln* (ursprünglich von Bulos als *model* bezeichnet) (vgl. Bulos, 1998, S. 254; Totok, 2000, S. 133). Ein Würfel bzw. Hyperwürfel führt mehrere Dimensionen zusammen, nach denen Kennzahlen, die in einer eigenen Dimension modelliert werden, betrachtet werden sollen. Dimensionen bestehen wiederum aus Dimensionsausprägungen. Ursprünglich umfasste der Ansatz keine eigene Kennzahlendimension, sondern erforderte je Kennzahl einen eigenen Würfel (vgl. Bulos und Forsman, 1998, S. 3f.). Totok und Jaworski unterbreiten einen Vorschlag zur Erweiterung des Ansatzes um einen Dimensionstyp für gleichdimensionierte Kennzahlen, der auch hier zu Grunde gelegt wird (vgl. Totok und Jaworski, 1998, S. 20ff.; Totok, 2000, S. 133). Bezüglich der Dimensionen zur näheren Beschreibung von Kennzahlen werden in der ADAPT-Methode neben der speziellen Kennzahlendimension weitere fünf Typen unterschieden. *Aggregierende Dimensionen* entsprechen dem Standardfall und verdichten die Daten mittels Hierarchien. *Sequentielle Dimensionen* werden eingesetzt, wenn die Reihenfolge der Dimensionsausprägungen eine Rolle spielt, wie z.B. die Reihenfolge von Monaten in einer Dimension Zeit. *Eigenschaftsdimensionen* sind eng mit Aggregations- oder Sequenzdimensionen verknüpft und ermöglichen eine genauere Beschreibung der Ausprägungen mittels weiterer Attribute, die auf alle Dimensionsausprägungen Anwendung finden. Sollen unterschiedliche Varianten einer Kennzahl in einem Würfel unterschieden werden, kommen *Versionsdimensionen* – auch partitionierende Dimensionen genannt (vgl. Totok, 2000, S. 135) – zum Einsatz. Einen speziellen Typ, der in verwandten Ansätzen derartig

nicht zu finden ist, stellt die *Tupeldimension* dar. Sofern Elemente unterschiedlicher Dimensionen über eine bestimmte Kombinatorik miteinander verbunden werden sollen, stellt die Tupeldimension die resultierenden Elemente bereit. Ein typisches Beispiel hierfür ist eine Kombination von Dimensionen, die lediglich für ausgewählte Kennzahlen relevant ist. Sofern das Erstellen eines eigenen Würfels weitere Analysen, die auf diesen Daten aufbauen, einschränken würde, stellt eine Tupeldimension die benötigten Elemente bereit (vgl. Gabriel und Gluchowski, 1998, S. 499f.; Totok, 2000, S. 135).

Eine einzelne Dimension kann des Weiteren in ihrer Struktur unabhängig vom Typus durch mehrere Beschreibungsobjekte näher definiert werden (vgl. Gabriel und Gluchowski, 1998, S. 500). Das wichtigste Objekt ist die Hierarchie, die wie in den bereits dargestellten konzeptuellen Modellierungsmethoden (z.B. im DFM, siehe Abschnitt 3.4.5) anhand von Hierarchieebenen unterschiedliche Granularitäten entlang von Verdichtungs- und Konsolidierungspfaden abbildet (vgl. Totok und Jaworski, 1998, S. 23). Sofern der korrespondierende Wert einer Dimensionsausprägung nicht als Teil einer Hierarchie abgebildet werden soll, kann dies mittels Berechnungsformeln geschehen (vgl. Gabriel und Gluchowski, 1998, S. 500; Totok, 2000, S. 133). Dimensionsattribute beziehen sich wiederum auf sämtliche Ausprägungen einer Dimension und ermöglichen neben der näheren semantischen Beschreibung auch eine Sortierung (vgl. Gabriel und Gluchowski, 1998, S. 500).

Im Vergleich zu den bereits beschriebenen Ansätzen bietet ADAPT deutlich mehr Möglichkeiten für die flexible Abbildung multidimensionaler Strukturen. Hierarchien sind nicht die einzige Möglichkeit zur Abbildung logischer Berechnungsvorschriften, sondern es existiert mit der Berechnungsformel ein eigenes Element für abgeleitete Elemente, die nicht als Teil einer Hierarchie abgebildet werden sollen. Ein zentraler Punkt des Ansatzes ist die Unterscheidung von sechs Dimensionstypen, die jeweils eine eigene Charakteristik mit sich bringen und Einfluss auf die weiteren zur Verfügung stehenden Modellierungselemente in Bezug auf eine Dimension nehmen. Im Sinne der Anforderung einer generischen Dimensionalität ist diese Unterscheidung kritisch zu betrachten. Dem Benutzer wird im Modellierungsprozess eine genaue Kenntnis der unterschiedlichen Dimensionstypen auferlegt, so dass der Modellersteller in bestimmten Situationen auf seine intuitive Modellierungsalternative verzichten muss. Teilmengen-differenzierte Kennzahlenspezifikationen werden ebenfalls nicht unterstützt, da es nicht möglich ist, für eine Kennzahl unterschiedliche Berechnungsformeln in Abhängigkeit von Ausprägungen weiterer Dimensionen zu

hinterlegen. Da simultane Referenzen ebenfalls unberücksichtigt bleiben, wird eine Erweiterung der ADAPT-Methode ausgeschlossen.

3.4.8 Multidimensional Modeling Language

Multidimensional Modeling Language (MML) ist eine am OFFIS (Oldenburger Forschungs- und Entwicklungsinstitut für Informatik) der Universität Oldenburg im Rahmen des Projektes ODAWA (OFFIS Tools for Data Warehousing) entwickelte Sprache zur konzeptionellen Modellierung multidimensionaler Data-Warehouse-Schemata. Die Darstellung der Methodik basiert auf den Ausführungen von Herden und Harren (vgl. Herden und Harren, 2004, S. 89ff.) sowie dem Dissertationsprojekt von Herden (vgl. Herden, 2001, S. 73ff.). Grundsätzlich werden *Daten*, ein zugehöriger *multidimensionaler Kontext* sowie *Beschreibungselemente* für die Darstellung von Struktureigenschaften unterschieden. Die MML wurde mittels einer multidimensionalen Erweiterung der Unified Modeling Language (UML) realisiert und nutzt Konstrukte aus der objektorientierten Programmierung, wie z.B. Vererbung oder Generalisierung.

Die MML besteht aus verschiedenen Konstrukten, die in die fünf Bereiche *Daten-Elemente*, *Hilfsmetaklassen*, *Multidimensionaler Kontext*, *Allgemeine Verbindungen* und *Properties* unterteilt sind. Ursprung eines jeden MML-Konstrukts ist die abstrakte Metaklasse MMLElement, in der jeder Klasse eine eindeutige Bezeichnung zugeordnet wird. Abbildung 3-10 zeigt einen vereinfachten Überblick der MML-Metaklassen anhand der Vererbungshierarchie und teilt die Metaklassen den fünf Bereichen zu.

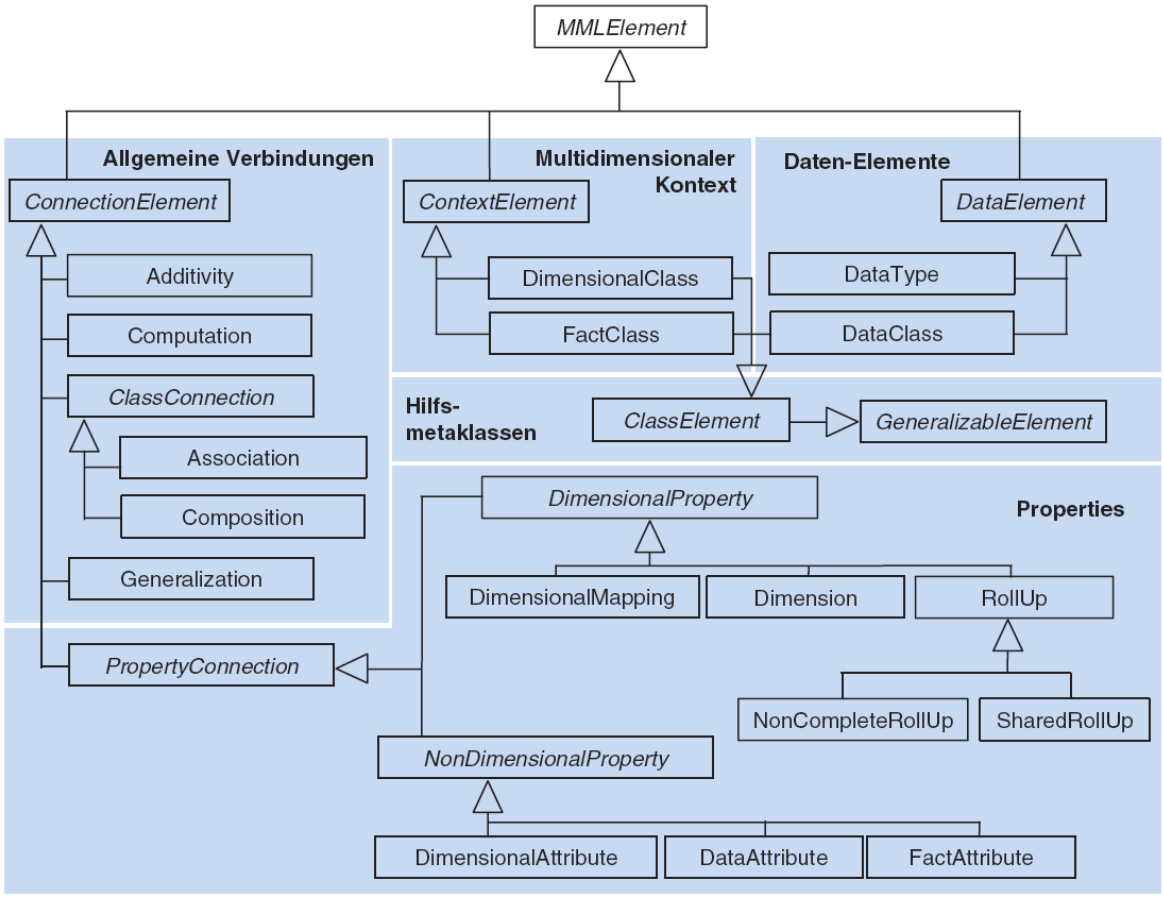


Abbildung 3-10: Das MML-Klassendiagramm

(Herden und Harren, 2004, S. 90)

Der Bereich *Daten-Elemente* gibt Auskunft über den Datentyp und beschreibt, ob es sich bei einem Datenelement um einen elementaren (Metaklasse *DataType*) oder einen komplexen Typ (Metaklasse *DataClass*) handelt. Komplexe Datenelemente sind in der Lage Vererbungsstrukturen aufzubauen (vgl. Herden und Harren, 2004, S. 90). Im Bereich *Hilfsmetaklassen* werden die Eigenschaften der Vererbung einer Klasse gesteuert. Der Bereich *Multidimensionaler Kontext* stellt die für die multidimensionale Datenmodellierung typische Differenzierung von Fakten und Dimensionsdaten bereit. Mögliche Verknüpfungen zwischen Schemaelementen werden in den Bereichen *Allgemeine Verbindungen* und *Properties* geregelt. Während erstgenannter typische objektorientierte Konstrukte, wie z.B. Assoziation und Komposition zur Verfügung stellt, dient der Bereich Properties der Darstellung multidimensionaler Sachverhalte, wie z.B. dem Aufbau von Hierarchien.

Anhand des Metamodells werden Schwachstellen einer Nutzung der MML für die Umsetzung einer konzeptuellen Modellierungsebene für MDSS deutlich. Im Bereich Allgemeine Verbindungen können Elemente über die Metaklassen *Association* oder *Composition* miteinander in Beziehung gesetzt werden. Bei einer Komposition ist sichergestellt, dass ein

(Teil-) Element, welches einem (Ganzes-)Objekt zugeordnet wird, nur existieren kann, sofern das aufnehmende Objekt selbst auch existiert. Bei einer Assoziation ist dies nicht sichergestellt. Eine Modellierung von Dimensionsausprägungen, die keiner Dimension zugeordnet sind, ist daher über eine Assoziation möglich.

Eine Komposition schließt aus, dass Dimensionen (*DimensionalClass*) und Fakten (*FactClass*) sowie zwei Dimensionen miteinander in Beziehung gesetzt werden dürfen. Diese Struktur hat zur Folge, dass Kennzahlen nicht als qualifizierende Dimension betrachtet werden können (vgl. Schultewolter, 2009, S. 29). Hinsichtlich Teilmengendifferenzierter Kennzahlenspezifikationen steht eine Klasse *NonCompleteRollUp* zur Verfügung, über die diese Anforderung zumindest für hierarchische Beziehungen umgesetzt werden kann.

Formeln bzw. Berechnungsvorschriften können mittels der Metaklasse *Computation* definiert werden. In einem Attribut *formula* können unabhängig von einer konkreten Formatierung logische Zusammenhänge bzw. Formeln auch mittels deskriptiver Beschreibungen hinterlegt werden. Für die Nutzung der MML im Kontext von MDSS ist dies ein bedeutender Nachteil, da somit Inkonsistenzen sowie logisch falsche oder unzulässige Formeldefinitionen im Sinne eines vorgegebenen Reglements nicht oder nur unter großem Aufwand geprüft werden können. Simultane Referenzen können aus diesem Grund in MML-Modellen ebenfalls nicht aufgedeckt werden, sodass letztendlich eine Erweiterung oder Anpassung der MML keine Alternative für die Umsetzung einer konzeptuellen Modellierungsebene für MDSS darstellt.

3.5 Zusammenfassende Betrachtung des State-of-the-Art

Nach der Betrachtung konzeptueller Modellierungsmethoden aus dem Themenbereich der multidimensionalen Datenmodellierung ist festzuhalten, dass sich die Methoden mit den hier adressierten MDSS (siehe Abschnitt 2.4) in beträchtlichem Maße überschneiden. Dies war aufgrund gemeinsamer bzw. ähnlicher Konstrukte (z.B. Dimensionen, Ausprägungen und Zellen) zu erwarten und ist ein Grund dafür, den Fokus bei der Betrachtung bestehender Ansätze auf diesen Bereich zu lenken.

Letztendlich ist das Charakteristikum datengetriebener Modelle jedoch ausschlaggebend dafür, dass eine Adaption oder eine Erweiterung eines Ansatzes für die Umsetzung einer konzeptuellen Modellierungsebene für MDSS nur unter sehr großem Aufwand sinnvoll bewerkstelligt werden könnte. In datengetriebenen Modellen liegt das Hauptaugenmerk auf

den Daten selbst, während in den Modellen der betrachteten MDSS (siehe Abschnitt 2.4) Daten zwar eine Rolle spielen, das Hauptaugenmerk jedoch auf den logischen Beziehungen zwischen diesen bzw. innerhalb dieser Daten liegt. Die charakteristischen Analysemethoden eines MDSS – *What-If-*, *Sensitivitäts-* und *Goal-Seeking-*Analyse – können jedoch nur dann Nutzen bringend eingesetzt werden, wenn die logischen Beziehungen im zugrunde liegenden Modell zur Verfügung stehen.

Für Ansätze, die auf bestehenden E/R-Modellen aufbauen, ergeben sich erhebliche Schwierigkeiten beim Aufdecken von Simultanität. Diese kann aus E/R-Modellen nur sehr schwer abgeleitet werden, da Beziehungen in einem E/R-Modell nur deskriptiv beschrieben werden. Ohne eine entsprechende Interpretation kann nicht festgestellt werden, ob eine bzw. mehrere Beziehungen simultane oder sequentielle Strukturen abbilden (vgl. Kapitel 3.4.5). Eine Unterstützung des Anwenders durch automatisiertes Aufdecken simultaner Formelbezüge ist daher in diesen Modellierungsansätzen unmöglich.

Hinsichtlich der formulierten Anforderungen für die Umsetzung einer konzeptuellen Modellierungsebene für MDSS (siehe Abschnitt 3.3) ist festzustellen, dass keine der untersuchten Modellierungsmethoden eine Berücksichtigung simultaner Beziehungen ermöglicht. Dies ist wiederum eine Folge des Fokus‘ auf die Daten, denn diese Anforderung ist eng mit den Berechnungsvorschriften bzw. Formeln in Modellen verknüpft. Teilmengen-differenzierte Kennzahlenspezifikationen, generische Dimensionalität und eine freie Modellierung hingegen werden in Teilen umgesetzt, jedoch bietet keine betrachtete Methode ein derart umfassendes Fundament, dass sich eine Adaption oder eine Erweiterung der Basiskonzepte eines Modellierungsansatzes anbieten würde, um die Anforderung einer konzeptuellen Modellierungsebene für MDSS zu erfüllen.

Durch die Beschreibung des Nutzens konzeptueller Modellierung (vgl. Abschnitte 2.2.2) wird deutlich, dass dieses methodische Vorgehen einen vielversprechenden Ansatz für MDSS darstellen kann. Heutige Werkzeuge zur Implementierung von MDSS verfügen in der Regel über eine proprietäre Modellierungssprache, die häufig nur für spezielle Situationen eine adäquate Modellierung ermöglicht. Für die hier adressierten MDSS ergibt ein konzeptuelles Vorgehen die Möglichkeit, proprietäre Methoden und deren Probleme zu umgehen und somit einen positiven Einfluss auf die Qualität von Modellen zu nehmen. Eine konzeptuelle Ebene beabsichtigt, Modelle für den Anwender verständlich darzustellen und sie letztendlich mit der Bewältigung von Komplexität durch Abstraktion und Verkürzung auf reale Probleme anwendbar zu machen. Aus diesen Gründen wird im Folgenden

ein konzeptueller Modellierungsansatz für MDSS entwickelt, der die formulierten Anforderungen freie Modellierung, generische Dimensionalität, Nichtprozeduralität, Simultanität, Teilmengen-differenzierte Kennzahlenspezifikation sowie das Problem der Fehleranfälligkeit von Spreadsheet-Modellen in MDSS berücksichtigt.

Teil II

Konzept eines multidimensionalen Modellierungsansatzes

4 Konzeption eines Modellierungsansatzes

Der erste Teil der Arbeit hat die Notwendigkeit sowie den Rahmen für eine effektive Unterstützung des Anwenders beim Erstellen von MDSS erläutert. Die adressierten Systemklassen im Bereich MDSS wurden durch spezifische, differenziert analysierte Charakteristika abgegrenzt. In Verbindung mit einer Analyse des relevanten State-of-the-Art bestehender konzeptueller Modellierungsansätze angrenzender Themenbereiche konnten Notwendigkeit und Anforderungen an eine konzeptuelle Modellierungsunterstützung spezifiziert werden.

Im zweiten Teil der Arbeit wird nun der Konzeptvorschlag erläutert. Konkret handelt es sich dabei um eine konzeptuelle Modellierungsebene für MDSS. Kapitel 4.1 stellt das Konzept aus einer ganzheitlichen Perspektive vor und ermöglicht so einen Überblick der einzelnen Konzeptteile und deren Nutzen. Kapitel 4.2 stellt die einzelnen Architekturkomponenten des Konzepts im Detail vor, erläutert deren Aufbau und stellt das jeweilig verfolgte Ziel vor. In Kapitel 4.3 wird eine Klassifizierung multidimensionaler Modellierungsfälle vorgeschlagen. Anhand der unterschiedlichen Charakteristika der Modellierungsfälle können Konsistenzinformationen eines Modells abgeleitet werden, die für die Unterstützung des Anwenders in den einzelnen Teilen des Konzepts genutzt werden. Kapitel 5 widmet sich einer detaillierten Beschreibung des konzeptuellen Modellierungsansatzes. Hier werden die Modellierungssprache, das resultierende konzeptuelle Modell, sowie die Möglichkeiten zur Unterstützung in den einzelnen Teilen des Konzepts detailliert erläutert.

4.1 Überblick

Das Leistungsvermögen und Potenzial modellgetriebener DSS wird vom Spektrum der Modellierungskonzepte definiert, die eine Modellierungskomponente dem Anwender für die Abbildung realer Situationen zur Verfügung stellt. Die Modellierungskomponente stellt die Schnittstelle zwischen MDSS und Anwender dar und setzt im Idealfall die klassische DSS-Forderung um, die Distanz zwischen Entwickler und Entscheidungsträger zu verringern (vgl. Power, 2002, S. 6; Sprague und Watson, 1996, S. 45; Turban et al., 2011a, S. 9). Analog zu datengetriebenen Systemen kann die Einführung einer konzeptuellen Ebene für

MDSS dem Entscheidungsträger das Verfolgen unterschiedlicher Modellzwecke[13] zur Unterstützung eines Entscheidungsprozesses erleichtern, indem dafür nötiges Werkzeug- bzw. Software-spezifisches Wissen zu Gunsten von Konzepten in den Hintergrund der Betrachtungen rückt. Konzeptuelle Modelle können grundsätzlich mit relativ geringem Aufwand in Modellinstanzen für unterschiedliche Anwendungssysteme je Modellzweck übersetzt werden, wobei deren Methodenspektrum bei der Übersetzung zu berücksichtigen ist.

Die Fehleranfälligkeit von Spreadsheet-Modellen (siehe Kapitel 3.1) hat ihren Ursprung in mehreren Teilen des Modellierungsprozesses. Die häufig schlechte oder unzureichende Integration multidimensionaler Zusammenhänge und die aus dem Vektor-basierten Formelverständnis resultierenden Formelüberschneidungen innerhalb von Modellzellen (siehe Kapitel 3.2.2.2) stellen zwei wichtige Aspekte dar, die im Prozess der Modellerstellung berücksichtigt werden müssen. Die Unterstützungsleistung der konzeptuellen Modellierungsebene bezieht ihren Nutzen daher maßgeblich aus genau dieser Berücksichtigung multidimensionaler Strukturen, Vektor-basierten Formeln und der Unabhängigkeit von konzeptuellem Modell und Anwendungssystemen mit unterschiedlichem, charakteristischem Funktionsspektrum.

Das Konzept adressiert inhaltlich drei Kernaufgaben: Die Modellierung multidimensionaler Modelle, die Herausforderung von Formelüberschneidungen und die Übersetzung der resultierenden konzeptuellen Modelle in Modellinstanzen der gewünschten Anwendungssysteme. Diese Einteilung schlägt sich im inhaltlichen Aufbau des Konzepts nieder. Der komplexe Anforderungskatalog (siehe Kapitel 3.3) resultiert zum Großteil aus den multidimensionalen Zusammenhängen und den damit verbundenen Problemen. Multidimensionalität sowie eine freie, transparente und verständliche Modellierung üben einen starken Einfluss auf den Prozess der Erstellung sowie den Aufbau eines Modells aus. Eine simultane Betrachtung multidimensionaler Strukturen und Vektor-basierter Formeln innerhalb einer Modellierungsphase würde dazu führen, dass der Anwender bei der Modellierung in Form von Rücksprüngen immer wieder bereits definierte Formeln anpassen müsste, weil später modellierte Formeln diese beeinflussen können, z.B. indem sich die Bezugsteile zweier Formeln überschneiden. Aus diesem Grund besteht das Konzept aus drei sequentiell aufeinander aufbauenden Phasen (Konzeptkomponenten). Abbildung 4-1 stellt die drei Phasen – Konfiguration, Transformation und Kompilierung – dar und visualisiert so den inhaltlichen Aufbau des Konzepts:

---

[13] Unterschiedliche Modellzwecke können in diesem Zusammenhang z.B. das Prognostizieren von Variablenwerten zukünftiger Perioden, Ursachenanalyse oder die Bewertung von Alternativen sein.

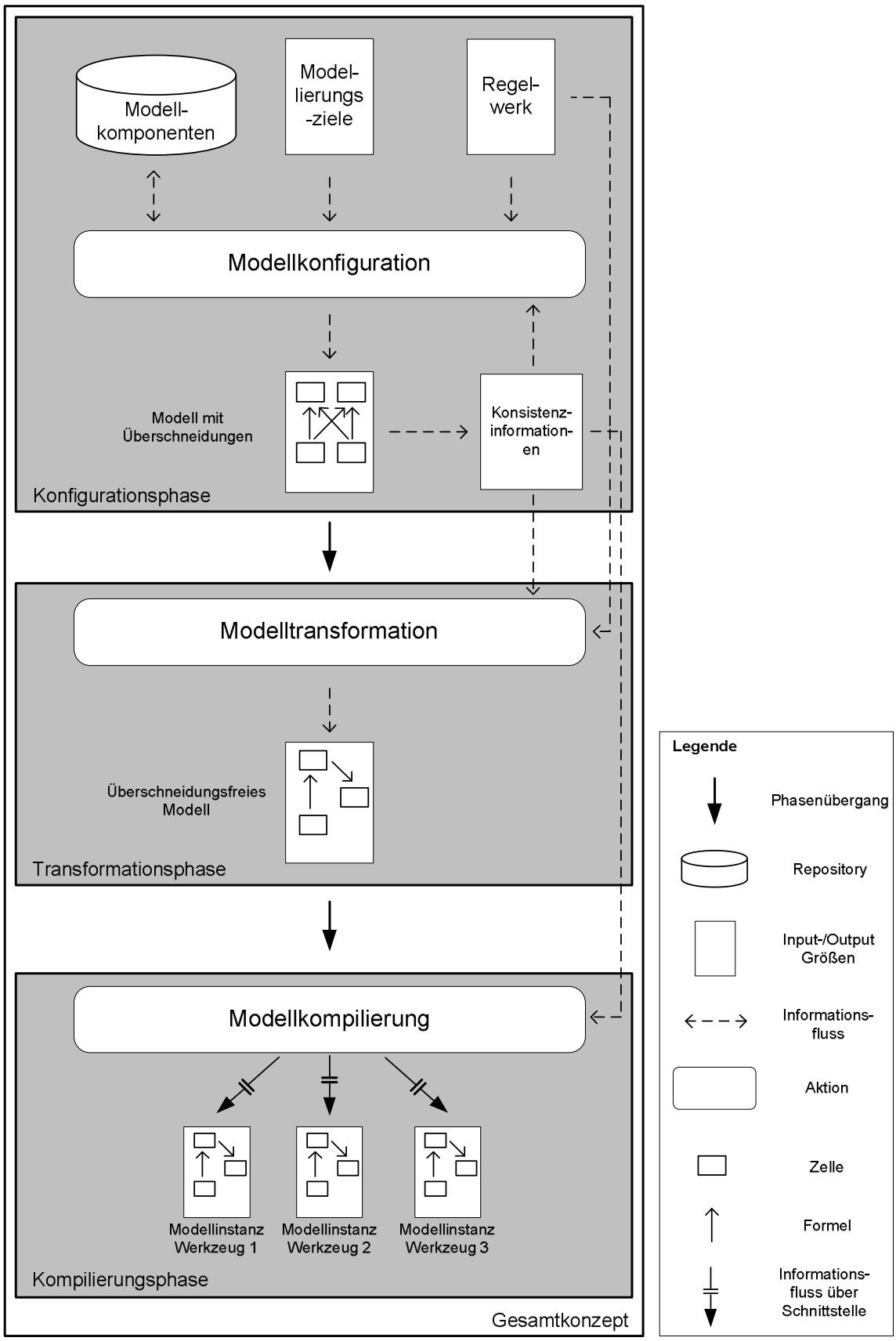


Abbildung 4-1: Überblick des Konzepts

In der Konfigurationsphase werden sämtliche Modellelemente (Dimensionen, Dimensionsausprägungen, Formeln, Gruppen) unter Berücksichtigung eines Regelwerks definiert. Es ist in diesem Schritt möglich, auf zuvor definierte Modellkomponenten innerhalb eines Repository zurückzugreifen. Des Weiteren wurde in der Subsumption des Modellverständnisses, das dieser Arbeit zugrunde liegt (siehe Kapitel 2.2.1.5), bereits festgestellt, dass jede Modellbildung für bestimmte Zwecke, respektive Ziele eines Modellbenutzers stattfindet. Während der Konfigurationsphase werden Überschneidungen von Formeln innerhalb von Modellzellen bewusst zugelassen und Konsistenzinformationen aus den definierten Zusammenhängen gewonnen.

In der darauffolgenden Transformationsphase[14] wird das in der Konfigurationsphase definierte Modell derart manipuliert, dass alle Überschneidungen von Formeln in sämtlichen Modellzellen beseitigt werden, sofern derartige Strukturen vorliegen. Diese zwei Teile des Konzepts – Konfigurations- und Transformationsphase – bilden den Modellierungsprozess der konzeptuellen Modellierungsebene für MDSS ab.

Die dritte Phase des Konzepts – die Kompilierungsphase – stellt abschließend die Integration der konzeptuellen Modelle in bestehende Systemarchitekturen sicher und befasst sich mit der automatisierten Übersetzung der konzeptuellen Modelle in proprietäre Modellinstanzen. Der Modellanwender kann auf diese Weise unterschiedliche Modellzwecke bei der Anwendung des Modells verfolgen. Die Kompilierungsphase ist nicht Teil des Modellierungsprozesses, weil lediglich innerhalb der Konfigurations- und der Transformationsphase (inhaltliche) Änderungen am Modell vorgenommen werden.

Die Trennung des Modellierungsprozesses in Konfigurations- und Transformationsphase bringt den Vorteil mit sich, dass dem Anwender im Sinne einer freien, transparenten und auch effizienten Modellierung die Aufgabe der Identifikation von Überschneidungen mehrerer Formeln in Modellzellen abgenommen werden kann. Diese Aufgabe übernimmt eine Komponente, welche die Transformationsphase des Konzepts realisiert. Sie identifiziert automatisiert die Teile eines konzeptuellen Modells, die einer eindeutigeren Definition bedürfen und greift dabei auf Konsistenzinformationen zurück. Mit Hilfe einer Anwenderinteraktion wird im Anschluss eine Konfliktlösung erarbeitet. Am Ende der Transformationsphase entsteht somit eine überschneidungsfreie Modellinstanz. Abbildung 4-2 gibt einen Überblick der Modellierungsschritte und fügt sie zu einem Modellierungsprozess

[14] Die Bezeichnung Transformation wurde gewählt, da in diesem Teil des Konzepts lediglich Änderungen des Bezugsteils einer Formel vorgenommen werden und Änderungen oder Definitionen von Modellelementen an dieser Stelle nicht möglich sind.

zusammen. Auf die Darstellung der Konsistenzinformationen wurde aus Gründen der Übersicht in dieser Abbildung verzichtet.

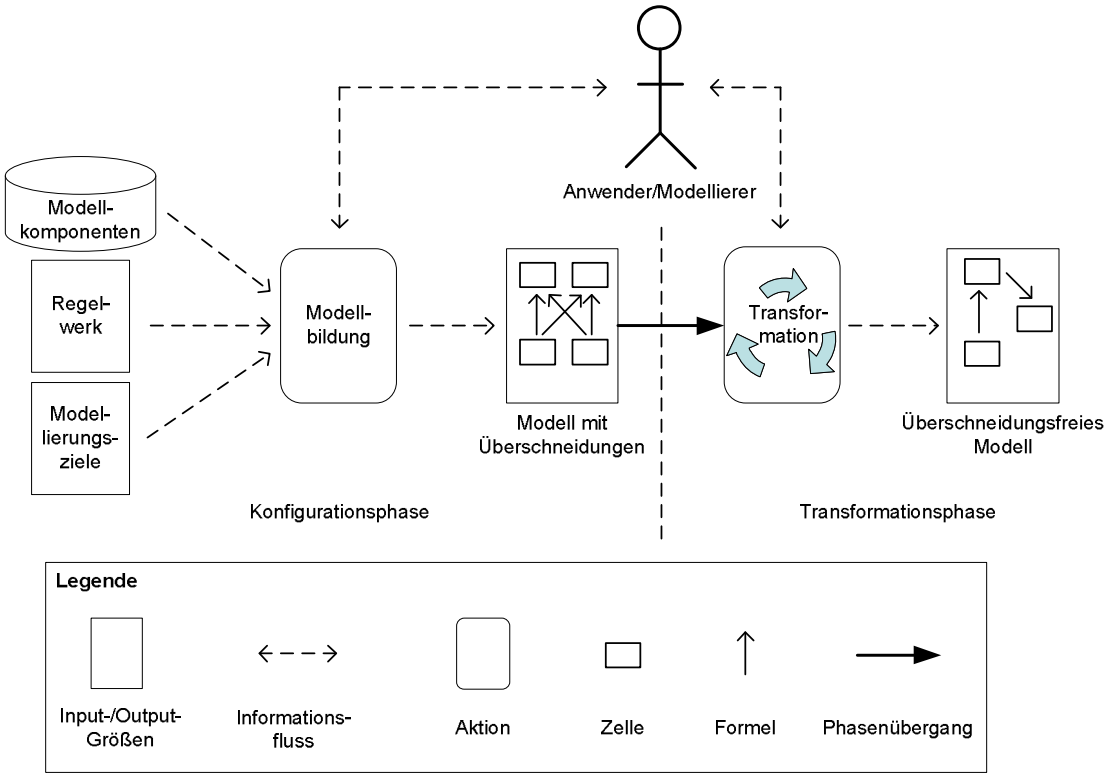


Abbildung 4-2: Überblick des Modellierungsprozesses

Die Aufspaltung des Modellierungsprozesses in zwei Phasen bietet des Weiteren die Möglichkeit einer effizienteren Unterstützung des Anwenders. Die Modellierungsumgebung kann Informationen bereitstellen, die sich auf das Ziel der jeweiligen aktuellen Phase beziehen und somit den Modellierer eine leichtere Interpretation der Zusammenhänge ermöglichen. Ohne getrennte Betrachtung in Form der beiden Phasen Konfiguration und Transformation wäre die Bereitstellung dieser Informationen zwar generell auch möglich, jedoch könnten die Informationen nicht so wirkungsvoll verwertet werden.

Um eine Bereitstellung dieser situativ relevanten Informationen zu ermöglichen, ist eine informationstechnisch realisierte Modellierungskomponente notwendig. Eine konzeptuelle Modellierungsebene für MDSS kann dementsprechend nicht ausschließlich anhand einer grafischen Notation umgesetzt werden (wie z.B. die ursprüngliche Entwicklung der ADAPT-Notation).

Das Konzept ermöglicht darüber hinaus die *Konsistenz* der Modelle hinsichtlich der Kriterien *Vollständigkeit* und *Eindeutigkeit* zu überprüfen, ohne auf eine proprietäre Modellinstanz angewiesen zu sein. Diese Konsistenzinformationen können bereits während der Konfigurations- und Transformationsphase des Modellierungsprozesses zur effektiven

Unterstützung des Anwenders ermittelt und bereitgestellt werden. Unter Konsistenz wird im Umfeld der Modellierung häufig die informationelle und formale Widerspruchsfreiheit hinsichtlich eines Meta-Modells verstanden, welches diesbezüglich als Regelwerk anzusehen ist (vgl. Balzer, 2009, S. 32; Becker et al., 1995, S. 437f.; Glinz, 2000, S. 51). Dies wird hier unter dem Begriff *Eindeutigkeit* subsummiert. Eindeutigkeit bezieht sich auf den inhaltlichen Aspekt der Formelüberschneidungen sowie die widerspruchsfrei definierte formale Struktur eines Modells. Ein weiterer wichtiger Aspekt ist die *Vollständigkeit*. Dies zielt im vorgefundenen multidimensionalen Kontext insbesondere auf ein unbeabsichtigtes Fehlen von Formeldefinitionen ab, deren Identifikation mit steigendem Modellumfang schwieriger wird (vgl. Fischer et al., 2008, S. 255). Die Unterstützungskomponente kann Auffälligkeiten hinsichtlich Vollständigkeit und Eindeutigkeit dem Anwender auf adäquate Art und Weise als Konsistenzinformation bereitstellen. Sind in einem Modell beispielsweise aus dem Bezugsteil einer Aggregationsformel bestimmte Modellzellen ausgeschlossen und existiert für diese Zellen keine alternative Formeldefinition, wird der Modellierer darauf aufmerksam gemacht.

Die einzelnen Konzeptkomponenten (siehe Kapitel 4.2) nutzen die Konsistenzinformationen eines konzeptuellen Modells[15]. Um diese einfacher und effektiver ableiten zu können, wird in Kapitel 4.3 eine Klassifizierung möglicher Modellierungsfälle in den adressierten MDSS vorgenommen.

Der folgende Abschnitt 4.2 stellt nun die Architekturkomponenten zur Umsetzung der einzelnen Phasen des Gesamtkonzepts vor und geht dabei vorrangig auf Anforderungen und Aufgaben der einzelnen Elemente ein, um die Unterstützungsleistung der einzelnen Teile des Konzepts zu betonen.

4.2 Architekturkomponenten

4.2.1 Modellierungseditor

Der Modellierungseditor realisiert die Konfigurationsphase des Konzepts und stellt die komplexeste Komponente dar. Der Anwender nutzt den Modellierungseditor, um ein Modell zu erstellen bzw. auf Basis bestehender Modellkomponenten eines Repository zu konfigurieren. Das Repository bietet die Möglichkeit, Elemente eines Modells unabhängig vom Modell zu speichern und stellt auf diese Weise deren Wiederverwendbarkeit sicher:

[15] Die Konsistenzinformation sind eng mit der Modellierungssprache sowie der Repräsentation eines konzeptuellen Modells verbunden. Sie werden im Detail im folgenden Kapitel erläutert (siehe Kapitel 5.3).

Es besteht die Möglichkeit modellierte Elemente in das Repository zu übernehmen und zu einem späteren Zeitpunkt wieder auf diese zuzugreifen, um sie für die Konfiguration eines anderen Modells zu nutzen. Darüber hinaus können Elemente, die aus einem Repository stammen, im Modellierungseditor angepasst bzw. verändert werden.

Ein wichtiger Bestandteil des Modellierungseditors sind die Regeln der Modellierung. Es muss beispielsweise sichergestellt werden, dass innerhalb eines Modells eine Dimensionsausprägung genau einer Dimension zugeordnet ist, oder dass eine Formel keine Dimensionsausprägungen enthält, die in der Zwischenzeit gelöscht wurden bzw. nicht im Modell vorhanden sind. Diese Regeln sind eng mit den Konsistenzinformationen verbunden und werden durch den Modellierungseditor implementiert. Sofern beispielsweise in einer Formel nicht existente Dimensionsausprägungen referenziert werden, ist ein Modell inkonsistent. Der Modellierungseditor erkennt dies und macht den Anwender während der Modellierung bzw. Konfiguration darauf aufmerksam, sodass dieser adäquate Änderungen zur Wiederherstellung der Modellkonsistenz ergreifen kann.

Für die Implementierung der Regeln der Modellierung, des Respository sowie einer Modellinstanz bietet sich ein gemeinsames Format an, da diese drei Komponenten eng miteinander verknüpft sind. Die eXtensible Markup Language (XML) ist eine flexibel gestaltbare Auszeichnungssprache (vgl. Vogt, 2006, S. 289). XML Schema ist eine vergleichsweise einfach umsetzbare Möglichkeit zur Spezifikation zulässiger Inhalte und der Struktur eines XML-Dokuments (vgl. Evjen et al., 2007, S. 169ff.). XML Schema ist die meist verwendete Methode zur Überprüfung der Validität eines XML-Dokuments (vgl. Evjen et al., 2007, S. 136). Im Kontext dieser Arbeit bietet sich XML Schema für die Umsetzung der Regeln der Modellierung an. Bezogen auf diese allgemeine XML Schema-Definition muss jedes mittels des Modellierungseditors definierte Modell valide sein. Modelle und das Repository werden jeweils als XML-Dokument umgesetzt. Abschnitt 5.2 erläutert das XML Schema der mittels des Editors definierten Modelle sowie eine beispielhafte Modellinstanz in Form eines XML-Dokuments.

4.2.2 Transformationskomponente

Die Transformationskomponente des Konzepts realisiert die gleichnamige zweite Phase des Modellierungsprozesses. Sie hat dementsprechend als einzige Aufgabe die Herstellung einer überschneidungsfreien Modelldefinition, auf Basis der im Modellierungseditor erstellten Zusammenhänge. Diese weisen in der Regel Inkonsistenzen hinsichtlich der Eindeutigkeit modellierter Zusammenhänge auf: Einzelne Zellen sind mit mehreren Formeln

belegt, da in der Konfigurationsphase Formelüberschneidungen explizit ausgeblendet werden.

Inhaltliche Änderungen an Modellelementen sowie das Erstellen oder Löschen von Modellelementen können ausschließlich in der Konfigurationsphase des Modellierungsprozesses vorgenommen werden. Auf diese Weise kann sichergestellt werden, dass zu einem transformierten Modell stets ein Ausgangsmodell existiert, das abgesehen von möglichen Formelüberschneidungen die gleiche Modelllogik repräsentiert. Des Weiteren müssen Konsistenzinformationen nach jeder Änderung von Modellelementen neu berechnet werden. Würde man lediglich Teile eines Modells transformieren und danach eine Dimensionsausprägung löschen, könnten Teile des Modells Fehler bzw. Inkonsistenzen aufweisen, z.B. indem die gelöschte Dimensionsausprägung in Formeln des bereits transformierten Modellteils referenziert wird. Um Transparenz und Flexibilität der Modellierung zu wahren, ist die Transformationskomponente eng mit dem Modellierungseditor verzahnt und erlaubt dem Anwender jederzeit aus der Transformations- zurück in die Konfigurationsphase zu springen. Diese Rückschritte resultieren aufgrund der möglichen Inkonsistenzen immer in einem Neustart der Transformationsphase.

Der eigentliche Prozess zur Auflösung der Überschneidungen und der daraus folgenden Beseitigung inkonsistenter, nicht eindeutiger Zelldefinitionen erfolgt im Dialog mit dem Anwender. Mittels einer Dialogkomponente werden ihm sämtliche Modellzellen, die mit mehr als einer Formel belegt sind, sowie die entsprechenden Formeln präsentiert. Eine dieser Formeln kann nun für die Zelldefinition ausgewählt werden. In Ausnahmefällen kann es auch Sinn machen, keine Formel zu wählen und die Zelle somit undefiniert zu lassen. In der Transformationsphase ist die Reihenfolge der Zellen bedeutungslos, in denen die Überschneidungen beseitigt werden. Während der Formelauswahl kann der Anwender die Auswirkungen der unterschiedlichen Formeln auf den Rest des Modells analysieren und eine Auswahl treffen.

Ist für jede Modellzelle eine Formel gewählt und somit eine eindeutige Modelldefinition im Sinne des Modellierers gefunden, wird das Modell abgespeichert. Für die Modelldefinition wird dabei auf eine Teilmengen-differenzierte Kennzahlenspezifikation zurückgegriffen: Sämtliche Modellzellen, die durch dieselbe Formel definiert werden, werden innerhalb sogenannter Gruppen zusammengefasst. Diese Gruppen werden in der Formeldefinition referenziert, um Redundanzen zu vermeiden. Auf diese Weise findet eine Zuordnung zwischen Zellmengen und Formel statt. Durch die Nutzung desselben Formats, wie in der

Konfigurationsphase, können auch nach der Transformationsphase Modellelemente, insbesondere Formelspezifikationen bearbeitet werden. Dies entspricht einem Rücksprung in die Konfigurationsphase des Modellierungsprozesses.

4.2.3 Kompilierungskomponente

Die Kompilierung[16] ist der Teil des Konzepts, der die Anwendbarkeit und Integration in bestehende Systemarchitekturen ermöglicht. Auf diese Weise können mit einem konzeptuellen Modell mehrere Modellzwecke verfolgt werden, da das Modell in unterschiedlichen Werkzeugen und für verschiedene Analysemethoden genutzt werden kann. Bei diesen Werkzeugen muss es sich nicht zwingend um Spreadsheet-basierte MDSS handeln, vielmehr ist es denkbar, dass die im Modellierungsprozess erstellten Modelle auch verwandten Systemklassen, wie z.B. OLAP-Systemen, zugänglich gemacht werden.

Die Überführung der konzeptuellen Modelle in vorhandene Anwendungssysteme erfordert eine Übersetzung in die vorhandenen, zumeist proprietären Modellierungssprachen. Dieser Übersetzungsprozess erfordert von der Schnittstelle eine Abbildung der konzeptuellen Modellelemente auf die angebotenen, proprietären Elemente. Durch den hohen Grad an Flexibilität und die möglichst geringen Einschränkungen der Modellierungsfreiheit kann eine Großzahl proprietärer Modellierungskonzepte in der Kompilierungsphase berücksichtigt werden. So ist es beispielsweise denkbar, dass die Unterscheidung von Dimensionstypen in einem Ziel-Anwendungssystem unterstützt wird. Im Fall einer Zeitdimension müsste z.B. vom Anwender während der Kompilierung in einem Dialog spezifiziert werden, welche der Dimensionen im konzeptuellen Modell diese Rolle in der Zielapplikation übernehmen soll.

Es kann in diesem Zusammenhang nicht sichergestellt werden, dass konzeptuelle Modelle eins zu eins in jedes Zielsystem übersetzt werden können. Dies liegt vor allem am eingeschränkten Spektrum der proprietären Konzepte der Zielsysteme, die grundlegende Konzepte, wie z.B. Simultanität, häufig nicht unterstützen. In diesen Fällen muss der Anwender während der Kompilierung Anpassungen am Modell vornehmen. Andere Einschränkungen, wie z.B. eine fehlende Unterstützung Teilmengen-differenzierter Kennzahlenspezifikationen, können bei der Übersetzung in eine Zielsystem-konforme Definition

[16] Die Bezeichnung Kompilierung wurde aufgrund der Nähe dieses Konzeptteils zu Compilern gewählt. Ein Compiler ist eine Systemsoftware, mittels derer Anweisungen (ein Programm), die in einer Programmiersprache verfasst sind, in eine Maschinensprache übersetzt (kompiliert) werden. Während der Übersetzung werden die Anweisungen auf Einhaltung der Syntax-Regeln der Programmiersprache geprüft (vgl. Heinrich et al., 2004, S. 151). Analog dazu werden die konzeptuellen Modelle in unterschiedliche proprietäre Formate verschiedener DSS-Anwendungen übersetzt.

ignoriert werden, sodass ein Teil der Modellinformationen im Zielsystem nicht zur Verfügung steht.

4.3 Klassifizierung multidimensionaler Modellierung

Die Konsistenz eines Modells betrifft Aussagen über die Eindeutigkeit und Vollständigkeit. Eindeutigkeit wird durch eine formale Widerspruchsfreiheit erreicht und zielt in den konzeptuellen Modellen auf Formelüberschneidungen. Vollständigkeit adressiert in den Modellen das unbeabsichtigte Fehlen von Formelspezifikationen. Dieser Abschnitt unterbreitet einen Klassifizierungsvorschlag für multidimensionale Formel-Modellierungsfälle. Anhand der Fälle lassen sich Konsistenzinformationen hinsichtlich Vollständigkeit und Eindeutigkeit von Modellen automatisiert und effektiv bestimmen. Des Weiteren kann der Anwender anhand der Fälle automatisiert auf notwendige Anpassungen von Formeln für die Kompilierung hingewiesen und es können Informationen für die Herstellung einer korrekten Berechnungsreihenfolge aller Modellzellen gewonnen werden. Diese ist für die jeweilige Modell-Berechnungs-Komponente eines Werkzeugs wichtig, in das ein konzeptuelles Modell kompiliert wird.

Sie berücksichtigt die Kriterien *Grad*, *Homogenität* und *Simultanität*, die sich aus den Anforderungen von Multidimensionalität und Vektor-basierten Formeln ergeben (vgl. Abschnitt 3.3). Nach einer theoretischen Einordnung und Darstellung der Basisannahmen, werden diese in Abschnitt 4.3.2 vorgestellt.

4.3.1 Basisannahmen

Die Klassifizierung[17] bezieht sich explizit auf die Formeln zur Abbildung von Relationen in einem diskreten, multidimensionalen Raum. In Verbindung mit den Konzepten MOO (minimale ontologische Überschneidung) nach Weber und Zhang (Weber und Zhang, 1996) sowie MOC (maximale ontologische Vollständigkeit) nach Green (Green, 1997) ist für die Klassifizierungskriterien zu fordern, dass alle realen Situationen mittels dieser Kriterien abgebildet werden können und sie sich möglichst nicht ontologisch überschneiden. Je weniger sich die Kriterien ontologisch überschneiden, desto besser sind die einzelnen Fälle voneinander abgrenzbar. Dies führt wiederum zu hochwertigen Konsistenzinformationen, die dem Anwender während der Modellierung sowie der Transformation zur Verfügung gestellt werden können. Ähnlich verhält es sich mit der maximalen ontologischen

[17] Die Ausführungen des Ansatzes zur Klassifizierung multidimensionaler Modellierung basieren auf drei bereits veröffentlichen Forschungsarbeiten (Schultewolter, 2009, 2010a, 2010b).

Vollständigkeit. Auch diese trägt dazu bei, dass einzelne Modellierungsfälle besser voneinander abgegrenzt werden können.

Sofern anhand der Fälle sämtliche Formeln innerhalb eines Modells einwandfrei voneinander unterschieden werden können, kann der Anwender bei der Modellierung und anschließenden Transformation sämtlicher realer Situationen adäquat unterstützt werden. Die Klassifizierungskriterien fördern damit über die Konsistenzinformationen eine effiziente, freie und transparente Modellierung.

Die Klassifikation berücksichtigt die Aspekte einer Formel, welche die Vollständigkeit und Eindeutigkeit eines Modells determinieren (vgl. Kapitel 4.1). Ein Fall der Klassifizierung ist immer im Kontext des gesamten Modells zu betrachten, da dieser Auswirkungen auf die Klassifizierung selbst hat. Das bedeutet, dass ein und dieselbe Formel in Abhängigkeit von den restlichen Modellformeln unterschiedlich klassifiziert werden kann, z.B. indem sich simultane Bezüge durch das Hinzufügen weiterer Formeln ergeben.

4.3.2 Klassifizierungskriterien

Die vorgeschlagenen Kriterien gehen in ihrer Grundstruktur auf Forschungsarbeiten von Rieger (vgl. Rieger, 1990, S. 253ff., 1994, S. 136f.) zurück. Es ist jedoch eine Anpassung der drei ursprünglich vorgeschlagenen Kriterien – Anzahl und Art der betroffenen Modelldimensionen sowie das betroffene Intervall – notwendig, da eine Unterscheidung von Dimensionstypen nicht verfolgt werden soll (siehe Kapitel 3.3.2) und damit die Art der Modelldimension nicht unterschieden werden kann. Die vorgeschlagenen Kriterien lauten *Grad*, *Homogenität* und *Simultanität* und werden in den folgenden Abschnitten 4.3.2.1, 4.3.2.2 und 4.3.2.3 detailliert dargestellt. Grad repräsentiert eine Eigenschaft multidimensionaler Modelle (vgl. Kapitel 3.2.1), Homogenität und Simultanität stellen direkte Anforderungen an eine konzeptuelle Modellierungsebene dar (vgl. Kapitel 3.3). Jedes Kriterium ist binärer Natur, d.h. kann zwei unterschiedliche Ausprägungen annehmen und bezieht sich auf die Art und Weise wie eine Formel Dimensionsausprägungen miteinander in Beziehung setzt.

Abbildung 4-3 stellt die Kriterien und die kombinatorisch resultierenden acht Fälle multidimensionaler Modellierung schematisch dar:

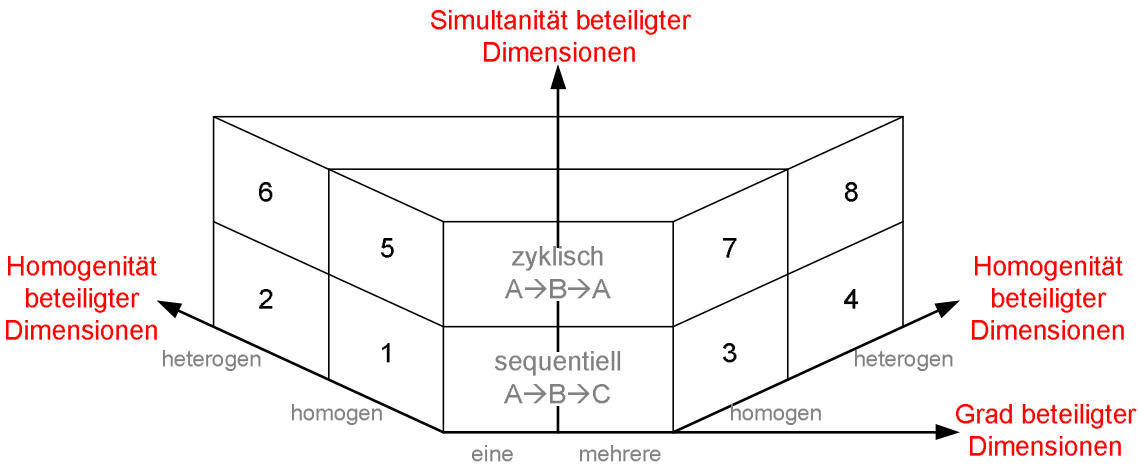


Abbildung 4-3: Klassifizierungskriterien multidimensionaler Modellierungsfälle

(Schultewolter, 2009, S. 20)

4.3.2.1 Grad beteiligter Dimensionen

Der *Grad beteiligter Dimensionen* (kurz Grad) beschreibt die Anzahl von Dimensionen, die an einer Formeldefinition explizit im Logikteil beteiligt sind. Mögliche Ausprägungen sind *eine* oder *mehrere* (Dimensionen). Der Grad einer Formel ermöglicht Aussagen darüber, ob eine Formel nur intra-dimensional wirkt, d.h. in ihrer Logik nur Dimensionsausprägungen einer Dimension miteinander in Beziehung setzt oder ob eine interdimensionale Verflechtung von Ausprägungen unterschiedlicher Dimensionen vorliegt (vgl. Schultewolter, 2009, S. 21, 2010a, S. 5, 2010b, S. 1117).

Aus Sicht des Gesamtkonzepts bietet sich hier die Möglichkeit, Informationen in Bezug auf die Modell-Berechnungs-Komponente zu gewinnen. Insbesondere das Herstellen der korrekten Berechnungsreihenfolge aller Modellzellen, die für die Kompilierung des konzeptuellen Modells in weitere Werkzeuge eine wichtige Rolle spielt, kann unterstützt werden.

Im Falle von intra-dimensionalen Formeln müssen für eine korrekte Berechnung der Zellwerte jene Modellzellen innerhalb einer Dimension richtig sortiert werden, die durch Formeln definiert werden. Als Kriterium für die Sortierung wird die Abhängigkeit der Formeln untereinander herangezogen. In Abbildung 4-4 sind alle Zellen, deren Wert durch *Formel 1* berechnet wird, mit einem Kreuz und alle Zellen, deren Wert durch *Formel 2* berechnet wird, mit einem Kreis gekennzeichnet. Für die Berechnung der Werte aller Zellen, an denen Ausprägung *a* der *Dimension 2* beteiligt ist, bedeutet dies, dass die Zelle *Größe 2:a* (Größe 2:a entspricht der Zelle mit den Ausprägungen *Größe 2* aus *Dimension 1* und *a* aus *Dimension 2*) vor der Zelle *Größe 1:a* berechnet werden muss. Dies ist notwen-

dig, da für die Berechnung des korrekten Wertes der Zelle *Größe 1:a* der Wert der Zelle *Größe 2:a* bereits vorliegen muss. Für die Modellzellen, an denen die Dimensionsausprägungen *b* und *c* aus Dimension 2 beteiligt sind, gilt diese Sortierung analog.

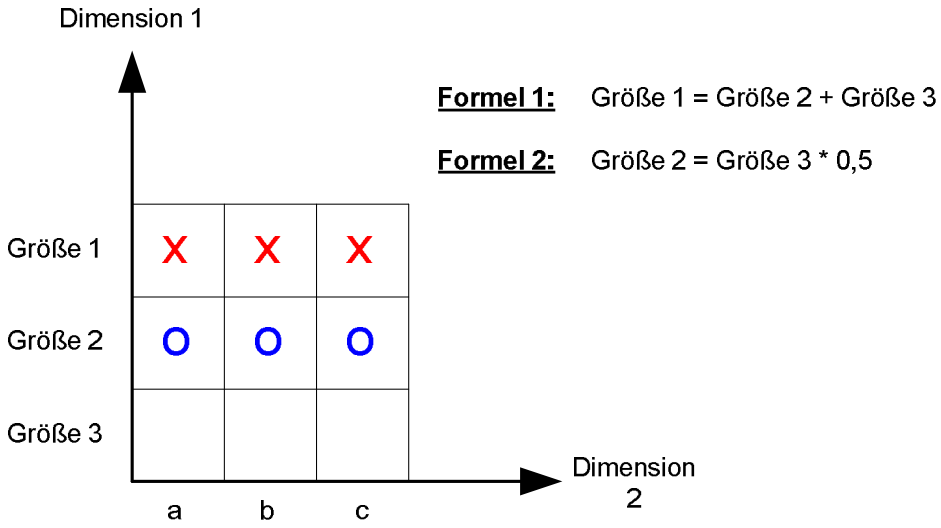


Abbildung 4-4: Beispielmodell für die Wirkung des Kriteriums Grad

Durch eine Modifikation von Formel 1 zu *Größe 1 = Größe 2 + Größe 3:b* können interdimensionale Formeln produziert werden. In diesem Fall muss für die Berechnung der korrekten Werte der Modellzellen die Referenz auf die Dimensionsausprägung *b* aus *Dimension 2* (*Größe 3:b*) berücksichtigt werden. Für die Berechnung der Werte aller Zellen, an denen Dimensionsausprägung *a* aus *Dimension 2* beteiligt ist, muss der Wert der Zelle *Größe 3:b* vorliegen. Die Zelle *Größe 1:a* kann nur berechnet werden, wenn der Wert der Zelle *Größe 3:b* vorliegt, da diese Zelle in Formel 1 explizit referenziert wird. Diese Sortierung gilt analog für die Modellzellen, an denen die Dimensionsausprägung *c* aus Dimension 2 beteiligt ist.

4.3.2.2 Homogenität beteiligter Dimensionen

Das Kriterium *Homogenität* bezieht sich auf den Anwendungsbereich (Bezugsteil) einer Formel. Aufgrund der Verwendung vektorieller Formeln im konzeptuellen Modell wirken sie grundsätzlich für alle Dimensionsausprägungen (des Bezugsteils). Der Anwender hat jedoch die Möglichkeit im Sinne einer Teilmengen-differenzierten Kennzahlenspezifikation den Anwendungsbereich von Formeln auf bestimmte Modellteile einzugrenzen (siehe Kapitel 3.3.5). Die Ausprägungen, die das Kriterium annehmen kann, sind *homogen* und *heterogen*. Homogene Formeln beziehen sich dabei immer auf eine uneingeschränkte Zellmenge innerhalb einer Dimension, heterogene Formeln sind dagegen auf einen Zellbereich mit einer echten Teilmenge mindestens einer Dimension beschränkt (vgl. Schulte-

wolter, 2009, S. 21, 2010a, S. 5f., 2010b, S. 1118). Auf diese Weise wird eine effiziente Modellierung ermöglicht, da keine redundanten Spezifikationen vorgenommen werden müssen. Des Weiteren wird eine exakte Definition derjenigen Zellen eines Modells ermöglicht, auf die eine Formel angewandt werden soll.

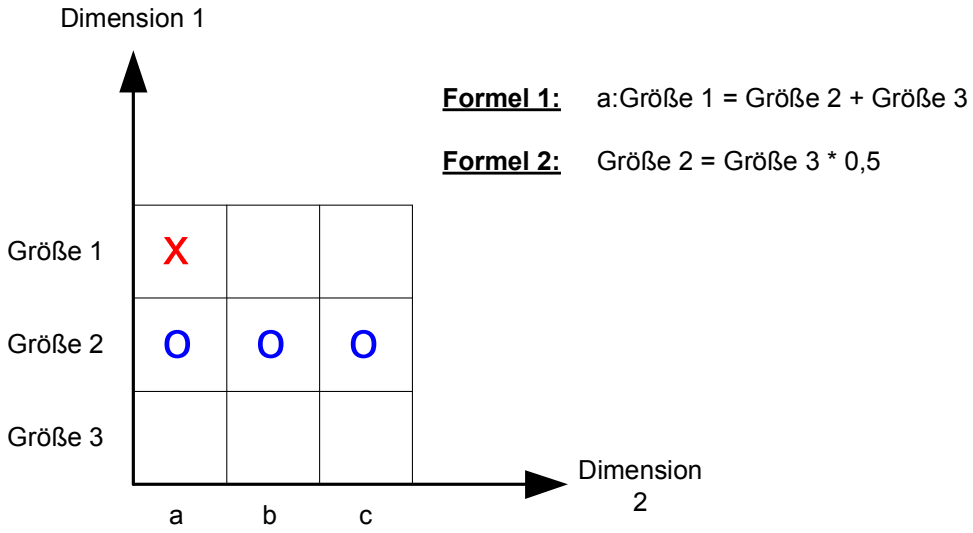


Abbildung 4-5: Beispielmodell für die Wirkung des Kriteriums Homogenität

Formel 1 in Abbildung 4-5 ist heterogener Natur, da die Zellmenge des Bezugsteils eingeschränkt ist. In diesem Fall ist dies die Zelle *Größe 1:a* (hier mit einem X gekennzeichnet). *Formel 2* hingegen ist homogener Natur und wird dementsprechend auf alle Zellen des Modells angewandt, die durch Kombination mit Ausprägung *Größe 2* aus *Dimension 1* entstehen (hier mit einem O gekennzeichnet). Für eine korrekte Berechnung aller Zellwerte muss eine Sortierung von Zellen nur für diejenigen Zellen vorgenommen werden, an denen Dimensionsausprägung *a* aus *Dimension 2* beteiligt ist. Im vorliegenden Beispiel muss dementsprechend der Wert der Zelle *Größe 3:a* vor dem Wert der Zelle *Größe 2:a* vorliegen. Der Wert der Zelle *Größe 1:a* kann danach berechnet werden. Für diejenigen Zellmengen, an denen Dimensionsausprägung *b* oder *c* aus *Dimension 2* beteiligt ist, muss eine analoge Sortierung stattfinden (im vorliegenden Beispiel sind die Zellen *Größe 1:b* und *Größe 1:c* nicht durch eine Formel belegt).

4.3.2.3 Simultanität beteiligter Dimensionen

Simultanität als drittes Kriterium macht anhand der Werte *zyklisch* und *sequentiell* eine Aussage darüber, ob eine Formel Bestandteil einer simultanen Berechnungsstruktur ist. Es unterscheidet sich daher von den Kriterien Grad und Homogenität, da Simultanität nur vorliegen kann, wenn mindestens zwei Formeln in einem Modell vorhanden sind. Simulta-

nität bezeichnet den Zustand, in welchem mehrere Formeln zur selben Zeit (diskret) voneinander abhängig sind (vgl. Schultewolter, 2009, S. 21, 2010a, S. 6, 2010b, S. 1118).

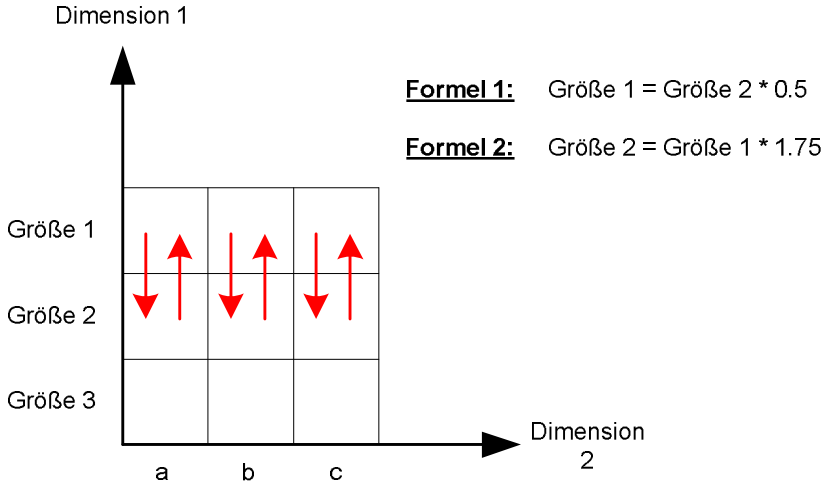


Abbildung 4-6: Beispielmodell für die Wirkung des Kriteriums Simultantität

Abbildung 4-6 stellt ein Modell dar, in welchem die Ausprägungen *Größe 1* und *Größe 2* direkt und zur selben Zeit voneinander abhängig sind. Die Pfeile beschreiben die Abhängigkeiten und Interaktionen der beiden Formeln untereinander.

In einem sequentiellen Berechnungsprozess können diese Berechnungsstrukturen nicht aufgelöst werden. Zur Bestimmung der Zellwerte müssen alle am Kreisschluss beteiligten Formeln in einem gesonderten Rechenschritt aufgelöst werden. Eine bekannte Methode hierfür stellt das Gauß'sche Eliminationsverfahren[18] dar.

Simultane Berechnungsstrukturen können über mehrere Formeln hinweg vorliegen. Hierbei ist eine Ausprägung a in Formel 1 von einer Ausprägung b in Formel 2 abhängig, Ausprägung b in Formel 2 ist wiederum von Ausprägung c in Formel 3 und Ausprägung c in Formel 3 letztendlich wieder von Ausprägung a in Formel 1. Die Menge dieser Zwischenschritte ist unbegrenzt und erschwert das Entdecken simultaner Strukturen.

4.3.3 Acht Fälle multidimensionaler Modellierung

Im folgenden Abschnitt 4.3.3 werden nun die acht Fälle multidimensionaler Modellierung beschrieben, die sich aus der Kombination der drei binären Kriterien ergeben. Anhand von Beispielmodellen werden die Strukturen und Charakteristika erläutert. Die vier sequentiellen und die vier simultanen Fälle werden aus Gründen der Übersicht gesondert voneinander betrachtet.

---

[18] Nähere Informationen zum Gauß'schen Eliminationsverfahren finden sich z.B. in (Lenze, 2006, S. 129ff.) und (Filler, 2011, S. 22ff.).

4.3.3.1 Sequentielle Fälle multidimensionaler Modellierung

Die sequentiellen Fälle multidimensionaler Modellierung zeichnen sich dadurch aus, dass immer eine korrekte Reihenfolge für die Berechnung der beteiligten Modellzellen ermittelt werden kann. Abbildung 4-7 skizziert jeweils ein zweidimensionales Beispielmodell für die unterschiedlichen Fälle.

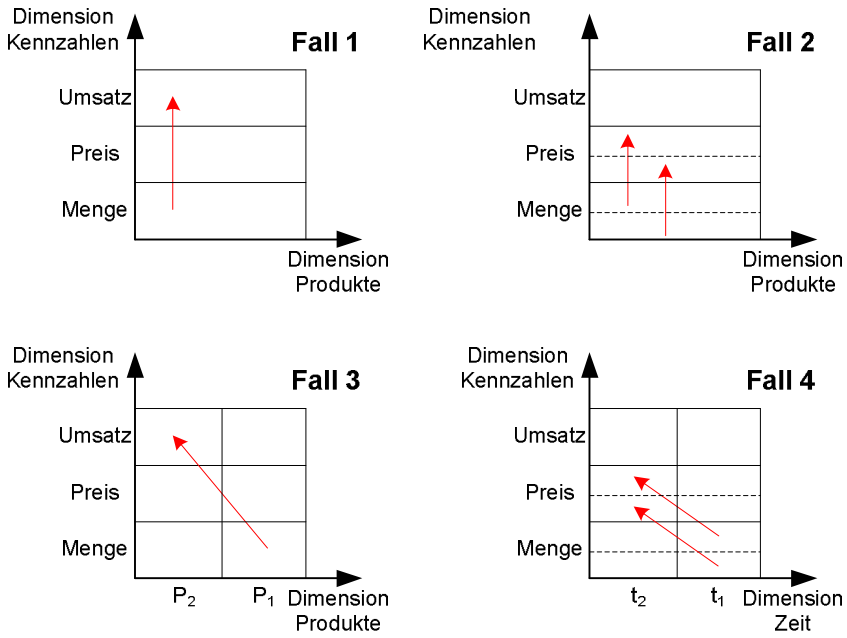


Abbildung 4-7: Sequentielle Fälle multidimensionaler Modellierung

Eine Formel von *Fall 1* ist homogener Natur und kombiniert ausschließlich Ausprägungen einer Dimension (Grad = eine). Eine korrekte Berechnungsreihenfolge der betroffenen Zellen muss hergestellt werden, sofern mindestens zwei Zellen innerhalb einer Dimension durch eine Formel spezifiziert werden. Des Weiteren muss durch eine Formel ein Zellwert referenziert werden, der anhand einer weiteren Formel berechnet wird. In diesem Fall ist als Sortierungskriterium die Abhängigkeitsstruktur der Formeln untereinander ausschlaggebend.

Fall 2 ist charakterisiert durch heterogene Dimensionsbezüge und berücksichtigt wie Fall 1 ebenfalls nur Ausprägungen einer Dimension. Auch hier kann die korrekte Berechnungsreihenfolge der Zellen innerhalb einer Dimension durch Sortierung anhand der Abhängigkeitsstruktur der betroffenen Formeln hergestellt werden. Im Unterschied zu Fall 1 ist zusätzlich eine Teilmengen-differenzierte Kennzahlenspezifikation spezifiziert. Dementsprechend wird die Formel nicht auf alle Zellen innerhalb einer Dimension angewandt. Denkbar ist auch, dass lediglich ein Parameter der Formel verändert wird. Dies wird jedoch wie eine weitere Formel behandelt, da ein Parameterwert „0“ in einer Multiplikation

die Wirkung einer Formel aufheben kann. Abbildung 4-7 zeigt für Fall 2 ein Modell, welches implizit einen Mengenrabatt berücksichtigt, so dass mindestens zwei (Teil-)Formeln für die Berechnung des Preises existieren. Abhängig von einer bestimmten Menge wird ein Rabatt auf den Preis berücksichtigt.

Liegen homogene Strukturen und Ausprägungen unterschiedlicher Dimensionen in einer Formel vor, ist die Konstellation von *Fall 3* gegeben. Für die korrekte Berechnungsreihenfolge können nicht ausschließlich die Zellen innerhalb *einer* Dimension herangezogen werden, sondern es müssen Zellwerte einer oder mehrerer weiterer Modelldimensionen berücksichtigt werden. Die Zellen werden entsprechend der Abhängigkeitsstruktur unter den beteiligten Formeln sortiert. In Abbildung 4-7 wird Fall 3 durch ein Modell repräsentiert, in welchem ein Produkt P_1 als Teil- bzw. Vorprodukt eines Produkts P_2 dient. Der Wert der Zelle *Menge:*P_1 muss in jedem Fall vor dem der Zelle *Umsatz:*P_2 im Berechnungsprozess ermittelt werden.

Fall 4 kennzeichnet Formeln, die neben heterogenen Strukturen Ausprägungen unterschiedlicher Dimensionen einbeziehen. Die Berechnungsreihenfolge von Zellen kann auch in diesem Fall ebenfalls nicht durch Sortierung der Zellen innerhalb einer Dimension hergestellt werden, sondern es müssen Zellwerte einer oder mehrerer weiterer Modelldimensionen berücksichtigt werden. Des Weiteren wird durch die Formel eine Teilmengendifferenzierte Kennzahlenspezifikation umgesetzt, sodass die Formel nicht auf alle Zellen innerhalb einer Dimension angewandt wird. Das Beispielmodell in Abbildung 4-7 berücksichtigt einen Mengenrabatt, welcher in Periode t_2 auf einen *Preis* in Abhängigkeit der *Menge* einer vorangegangenen Periode t_1 gewährt wird. Der Wert der Zelle *Menge:*t_1 muss in diesem Fall vor dem Wert der Zelle *Preis:*t_2 vorliegen.

4.3.3.2 Simultane Fälle multidimensionaler Modellierung

Für die simultanen Fälle der Klassifizierung gilt allgemein das gleiche, wie für das jeweilige Pendant der sequentiellen Fälle (z.B. Fall 1 und Fall 5). Die Formeln an sich weisen keine andere Struktur oder besonderen Merkmale auf. Über die Ausführung des vorangehenden Abschnitts hinaus liegen jedoch gegenseitige Abhängigkeiten zwischen Dimensionsausprägungen anhand einer oder mehrerer Formeln vor.

Bei der Herstellung der korrekten Berechnungsreihenfolge werden zuerst sämtliche Zellen berechnet, die unabhängig von allen Zellen sind, über welche der kreisförmige, simultane Bezug zustande kommt. Die Werte dieser „simultanen Zellen“ werden jeweils in einer Nebenrechnung unter Einsatz eines algorithmischen Verfahrens bestimmt. Die Herausforde-

rung simultaner Bezüge besteht demnach in der Bestimmung der Punkte im Gesamtberechnungsprozess, an welchen die Nebenrechnungen eingeschoben werden.

Abbildung 4-8 gibt wiederum einen Überblick der vier simultanen Fälle und skizziert ein Beispielmodell sowie die Wirkung der charakterisierten Formel.

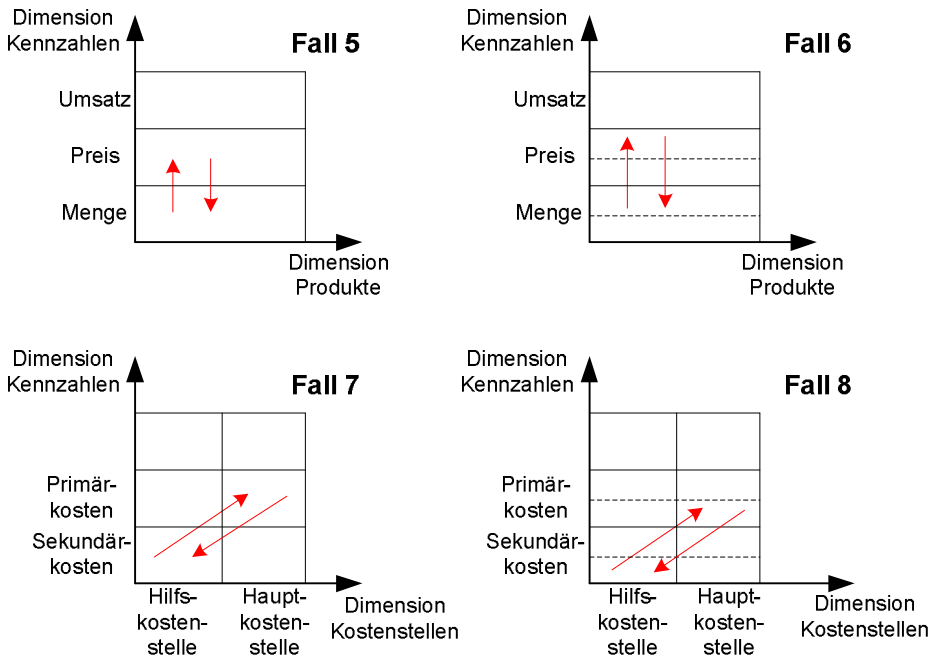


Abbildung 4-8: Simultane Fälle multidimensionaler Modellierung

Aufgrund der Gemeinsamkeiten mit den sequentiellen Fällen wird im Folgenden die Struktur der Fälle stark verkürzt angegeben und lediglich das Beispielmodell erläutert.

Fall 5 (homogen, eine Dimension) bildet eine Angebots- und Nachfragefunktion (-formel) ab. Die (Angebots-)*Menge* bedingt den *Preis* und der *Preis* wird gleichzeitig von der (Nachfrage-)*Menge* beeinflusst. Die Werte der Zellen *Preis* und *Menge* müssen in diesem Beispiel in einer Nebenrechnung ermittelt werden, da sie einen simultanen Bezug abbilden. Im Anschluss an die Nebenrechnung könnte der Wert der Zelle *Umsatz* (=Menge * Preis) bestimmt werden.

Im Unterschied zur Angebots- und Nachfragefunktion aus Fall 5 wird in *Fall 6* (heterogen, eine Dimension) die Angebotsfunktion für ein bestimmtes Preis- und die Nachfragefunktion für ein bestimmtes Mengenintervall als vollkommen unelastisch angenommen. Entsprechend einer Teilmengen-differenzierten Kennzahlenspezifikation liegt für die Zellmenge der Dimension *Kennzahlen*, die das elastische Intervall repräsentiert, ein simultaner Bezug zwischen den Ausprägungen *Preis* und *Menge* vor, der in einer Nebenrechnung aufgelöst werden muss.

Fall 7 (homogen, mehrere Dimensionen) wird durch das Beispiel der innerbetrieblichen Leistungsverrechnung dargestellt. Leistungen der *Hilfskostenstelle* werden von der *Hauptkostenstelle* in Anspruch genommen, gleichzeitig werden auf der *Hilfskostenstelle* jedoch auch Leistungen der *Hauptkostenstelle* verrechnet. In die Nebenrechnung zur Auflösung des simultanen Bezugs müssen in diesem Fall Zellen unterschiedlicher Dimensionen berücksichtigt werden.

Erweitert man die innerbetriebliche Leistungsverrechnung um Intervalle, gelangt man zu *Fall 8* (heterogen, mehrere Dimensionen). Sofern aus Kapazitätsgründen die *Hilfskostenstelle* der *Hauptkostenstelle* eine gewisse Leistung nur in begrenztem Maße zur Verfügung stellen kann, die *Hauptkostenstelle* auf diese Leistung jedoch angewiesen ist, muss sie diese zu marktüblichen *Preisen* (die sich von der internen Verrechnung unterscheiden) „von außen“ beziehen. Es ergibt sich dadurch wiederum nur für diejenigen Zellen der Dimension *Kennzahlen* ein simultaner Bezug, welche die interne Verrechnung repräsentieren. Dieser muss in einer Nebenrechnung aufgelöst werden.

5 Konzeptuelle Modellierungssprache

Nachdem im letzten Kapitel ein Gesamtüberblick über das Konzept gegeben wurde, wird nun detailliert auf die entworfene Sprache mit ihren Modellierungselementen, deren Verwendung im Modellierungsprozess sowie mögliche Unterstützungsfunktionen auf Basis von Konsistenzinformationen eingegangen.

5.1 Wahl der Methodik

Dem Framework zur Erforschung konzeptueller Modellierung von Wand und Weber folgend (siehe Abschnitt 2.2.2), bietet eine konzeptuelle Modellierungssprache (conceptual modeling grammar) Konstrukte (Modellelemente) und Regeln zur Kombination dieser Konstrukte an, mit deren Hilfe reale Zusammenhänge abgebildet werden können (vgl. Wand und Weber, 2002, S. 364). Für die Wahl der Methodik zur Realisierung der konzeptuellen Modellierungssprache bedeutet dies, dass ein Anwender in der Lage sein muss, anhand der Methodik Modellelemente und Regeln zur Kombination dieser Modellelemente zu spezifizieren. Bei der Auswahl werden die beiden Konzepte MOO (minimale ontologische Überschneidung) nach Weber und Zhang (Weber und Zhang, 1996) und MOC (maximale ontologische Vollständigkeit) nach Green (vgl. Green und Rosemann, 1999, S. 236; Green, 1997, S. 7) als Anforderungen an eine Modellierungssprache berücksichtigt, weil anhand der Modellelemente erstens sämtliche reale Situationen und zweitens diese Situationen möglichst eindeutig abbildbar sein sollen (siehe Kapitel 2.2.2).

Die eXtensible Markup Language (XML)[19] ist eine flexibel gestaltbare Auszeichnungssprache (vgl. Vogt, 2006, S. 289) und erfüllt im Zusammenspiel mit XML Schema, für die Formulierung von Regeln (vgl. Evjen et al., 2007, S. 169ff.), diese Anforderungen. Es ergibt sich durch einen einfachen Datenaustausch anhand von XML-Dokumenten die Möglichkeit, die einzelnen Teile des Konzepts (Modellierungseditor, Transformations- und Kompilierungskomponente), die eng miteinander verknüpft sind, mittels unterschiedlicher Werkzeuge oder Entwicklungsumgebungen umzusetzen. Jede Komponente muss lediglich eine XML-Schnittstelle bereitstellen, mittels derer Modelle interpretiert werden können. Im Falle des Modellierungseditors sowie der Transformationskomponente müssen die mo-

[19] Der jeweils aktuelle XML Standard des W3C (world wide web consortium) ist hier zu finden: http://www.w3.org/TR/xml/

dellierten und transformierten Modelle darüber hinaus im definierten XML-Format gespeichert werden können.

XML ermöglicht mittels *Tags*, *Elementen* und *Attributen* die Strukturierung von Daten (vgl. Evjen et al., 2007, S. 5f.). Elemente spiegeln Daten wider, die mit Meta-Daten in Form von Tags angereichert ist. Diese Tags verleihen den Daten eine Semantik (vgl. Evjen et al., 2007, S. 7). Attribute ermöglichen die nähere Beschreibung von Tags und sind ebenfalls als Meta-Daten zu interpretieren. Sie ermöglichen es die Semantik eines Tags zu beeinflussen, ohne die Struktur der Daten zu ändern (z.B. weitere Kind-Elemente zur näheren Beschreibung). XML-Dokumente weisen eine frei definierbare Baumstruktur von Elementen auf und sind nicht an spezielle Anwendungsumgebungen, Betriebssysteme oder weitere Software gebunden. Sie können durch die strikte Trennung von Daten- und Darstellungsspezifikation auf unterschiedlichen Medien verfügbar gemacht werden. Des Weiteren können sie mit relativ geringem Aufwand erweitert werden (vgl. Evjen et al., 2007, S. 4ff.; Harold und Means, 2004, S. 3ff.). Abbildung 5-1 zeigt ein Beispiel eines typischen XML-Dokuments:

```
<?xml version="1.0"?>
<product barcode="2394287410">
  <manufacturer>Verbatim</manufacturer>
  <name>DataLife MF 2HD</name>
  <quantity>10</quantity>
  <size>3.5"</size>
  <color>black</color>
  <description>floppy disks</description>
</product>
```

Abbildung 5-1: Ein typisches XML-Dokument

(Harold und Means, 2004, S. 7)

Das beispielhafte XML-Dokument zeigt eine Produktspezifikation, die aus den Kind-Elementen *manufacturer*, *name*, *quantity*, *size*, *color* und *description* sowie einem Attribut *barcode* besteht.

XML bietet als beschreibende Sprache nur wenige integrierte Ansätze zur Abbildung logischer Zusammenhänge im Sinne von Verzweigungen und Verknüpfungen. Eine dieser wenigen Ansätze ermöglicht es wenigstens, Elemente in XML mit einer dokumentweit eindeutigen ID zu versehen. Der Anwender wird dadurch bei der Vermeidung redundanter Definitionen und damit bei der Erzeugung valider XML-Dokumente unterstützt. Jedes Element muss durch dieses Konzept nur einmal in einem XML-Dokument definiert wer-

den und kann in komplexen Baumstrukturen mittels eines Attributs *idref* referenziert werden.

Die Regeln der konzeptuellen Modellierung (siehe Kapitel 4.1) werden mittels einer XML Schema-Definition abgebildet. Das XML Schema legt die Struktur eines konzeptuellen Modells auf XML-Ebene fest, indem z.B. Regeln über Kardinalitäten in Elter-Kind-Beziehungen von XML-Elementen und deren Datentypen festgelegt werden. Ein konzeptuelles Modell in Form eines XML-Dokuments kann nach dem Modellierungsprozess auf Validität hinsichtlich der XML Schema-Definition geprüft werden. Bezogen auf diese XML Schema-Definition muss jedes mittels des Modellierungseditors definierte Modell valide sein. Ursprünglich war vom W3C für die Validierung von XML-Dokumenten die sogenannte *Document Type Definition* (DTD) vorgesehen. 2001 nahm das W3C XML Schema in den XML-Standard auf, da auf diesem Weg deutlich detailliertere und genauere Spezifikationen von XML-Dokumenten, insbesondere die Angabe unterschiedlicher Datentypen, möglich sind. Heute ist XML Schema die meist verwendete Methode zur Überprüfung der Validität von XML-Dokumenten (vgl. Evjen et al., 2007, S. 136). Des Weiteren erfolgt ihre Definition auf einer XML-basierten Schreibweise und nicht in einem eigenen Format, wie es bei einer DTD der Fall ist (vgl. Evjen et al., 2007, S. 169; Vogt, 2006, S. 292).

Im folgenden Kapitel 5.2 werden die Definitionen der Modellelemente auf Ebene des entwickelten XML Schemas dargestellt. Im Anschluss an die Erläuterung der einzelnen Modellelemente wird jeweils ein beispielhaftes XML-Element vorgestellt, das hinsichtlich der XML Schema-Definition valide ist.

5.2 Modellierungssprache

Die Modellierungssprache besteht aus fünf Hauptkomponenten, mittels derer ein Modell definiert wird. *Modelle*, *Dimensionen*, *Dimensionsausprägungen*, *Gruppen* und *Formeln* interagieren miteinander und lehnen sich an die Modellierungskomponenten an, die das Spreadsheet-basierte Werkzeug *Quantrix Modeler*[20] zur Verfügung stellt. Diese Wahl begründet sich durch die größte Nähe zur hier entwickelten konzeptuellen Modellierungsebene für MDSS. Einzelne (gleichnamige) Modellelemente unterscheiden sich jedoch von denen im Werkzeug Quantrix Modeler. So besitzen sämtliche Modellelemente Attribute,

[20] Quantrix ist ein Unternehmen der ID Business Solutions LTD. Die Aussagen beziehen sich auf die Modellierungskomponente des Quantrix Modeler in der Version 4.1.47. Nähere Informationen finden sich unter http://www.quantrix.com.

welche die Auswertbarkeit und Interpretierbarkeit der Elemente gegenüber Quantrix erweitern. Über dies hinaus gibt es auch funktionale Unterschiede der einzelnen Elemente. Es ist im Gegensatz zur Quantrix-Notation beispielsweise möglich, Dimensionsausprägungen unterschiedlicher Dimensionen in einer Gruppe zusammenzufassen, einzelne Dimensionsausprägungen als Teil mehrerer Gruppen zu modellieren und das Verhalten von Dimensionsausprägungen zu beeinflussen, wenn diese auf eine summierende Formel treffen.

5.2.1 XML-Modellkomponente Model

Das grundlegende Element eines jeden konzeptuellen Modells auf XML-Basis ist *model*. Ein model-Element umfasst ein Bezeichner-Element (*name*) sowie mehrere Dimensionselemente (*dimension*), die in ihrer Anzahl unbegrenzt sind. Hinzu können die Elemente *from-repository*, das die Bezeichnung eines Repository enthält, sofern bei der Modellierung Elemente aus einem Repository genutzt wurden, sowie eine unbeschränkte Anzahl von Gruppen-Elementen (*group*) kommen. Diese sind jedoch nicht zwingend vorhanden. Abbildung 5-2 zeigt die Definition des model-Elements im XML Schema. Grundlegend ist zu erwähnen, dass mittels der Indikatoren *minOccurs* und *maxOccurs* Kardinalitäten, genauer die Mindest- sowie die Höchstanzahl von Kind-Elementen vorgegeben wird, die das jeweilige Elter-Element beinhalten kann. Sofern minOccurs und maxOccurs nicht angegeben werden, wird als Standardwert beider Ausprägung „1“ angenommen. (vgl. Evjen et al., 2007, S. 192ff.). Wird auf die Angabe von maxOccurs und minOccurs verzichtet, muss ein Element genau einmal als Elter-Element oder als Kind-Element (sofern es als Teil eines anderen Elements definiert wird) vorhanden sein.

```
<xs:element name="model">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="from-repository" type="xs:string" minOccurs="0"/>
      <xs:element name="name" type="xs:string"/>
      <xs:element maxOccurs="unbounded" ref="dimension"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="group"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>
```

Abbildung 5-2: XML Schema Definition des Elements Model

Die Elemente von Gruppen und Dimensionen werden an dieser Stelle lediglich referenziert und mit Kardinalitäten versehen (siehe Abbildung 5-2: Attribut *ref*). Die Definitionen der Elemente Dimension und Gruppe folgen in den nächsten Abschnitten 5.2.2 und 5.2.4. Ab-

schließend erhält ein model-Element ein Attribut *ID*, über das ein Element in einem XML-Dokument eindeutig referenziert werden kann.

```
<model id="Model_1">
  <name>Modell</name>
 - <dimension id="dimension_0">
  (...)
  </dimension>
  <group id="group_1" contentType="dim-item">
  (...)
  </group>
</model>
```

Abbildung 5-3: Beispiel für ein XML-Element Model

Abbildung 5-3 zeigt ein Beispiel eines XML-Elements model, das hinsichtlich der beschriebenen XML Schema-Definition valide ist. Die Kind-Elemente des Elements Model werden erst in der Folge definiert und daher sind nur einleitende und schließende Tags einer Dimension und einer Gruppe dargestellt.

5.2.2 XML-Modellkomponente Dimension

Eine Dimension (*dimension*) ist strukturell sehr ähnlich aufgebaut wie ein model-Element. Abbildung 5-4 gibt einen Überblick der Definition. Kind-Elemente sind ein Bezeichner (*name*) sowie mindestens eine Dimensionsausprägung (siehe Abbildung 5-4: Attribut *ref="dim-item"*) und bei Bedarf eine beliebige Anzahl an Formeln (siehe Abbildung 5-4: Attribut *ref="formula"*).

```
<xs:element name="dimension">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element maxOccurs="unbounded" ref="dim-item"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="formula"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>
</xs:element>
```

Abbildung 5-4: XML Schema Definition des Elements Dimension

Dimensionen werden genutzt, um inhaltlich zusammenhängende Beschreibungsobjekte zusammenzufassen (siehe Abschnitt 3.2.1). Sie stellen eine Art Gerüst dar, welches die grundlegende inhaltliche Thematik eines Modells definiert. Die Anzahl von Dimensionen

ist grundsätzlich unbeschränkt, jedoch wird als Richtwert eine Beschränkung auf sechs bis acht, in Ausnahmefällen auch zehn Dimensionen angenommen (vgl. Gabriel et al., 2009, S. 78). Bei Modellen mit mehr Dimensionen kann deren Übersichtlichkeit nicht gewährleistet werden (vgl. Gluchowski et al., 2008, S. 147) und es ist davon auszugehen, dass eine unsaubere, nicht realitätskonforme Modellierung vorliegt.

```
<dimension id="dimension_1">
  <name>Zeit</name>
  <dim-item id="Zeit_100" cid="Zeit2011" (...)>
  (...)
  </dim-item>
  (...)
  <formula id="formula_100" (...)>
  (...)
  </formula>
</dimension>
```

Abbildung 5-5: XML-Element einer Dimension

Abbildung 5-5 visualisiert ein Beispiel einer Dimension *Zeit* im XML-Format eines Modells. Auf die genaue Darstellung der Kind-Elemente, welche Dimensionsausprägungen und Formeln darstellen, wird wiederum verzichtet, da diese in den nächsten Abschnitten 5.2.3 und 5.2.5 erläutert werden.

5.2.3 XML-Modellkomponente Dimensionsausprägung

Dimensionsausprägungen (*dim-item*) repräsentieren die unterste Ebene eines Modells und können dementsprechend keine strukturierten Kind-Elemente beinhalten, die selbst Elter-Element weiterer Kind-Elemente sind. Abbildung 5-6 zeigt die XML Schema-Definition einer Dimensionsausprägung. Neben einem Kind-Element für den Bezeichner (*name*) verfügen Dimensionsausprägungen über drei Attribute. Das Attribut *dimension* enthält aus Effizienzgründen den Bezeichner der Dimension, welcher die Dimensionsausprägung angehört. Bei der Weiterverarbeitung dieser Informationen kann auf diese Weise die Zahl der Abfragen reduziert werden. Die beiden Attribute *summierbar* und *summiert* ermöglichen Aussagen über das Formelverhalten von Modellzellen und können jeweils die Werte „*ja*“ oder „*nein*“ annehmen. Zellen entstehen durch die Kombination von genau einer Dimensionsausprägung je Modelldimension (siehe Abschnitt 3.2.1). Der Anwender kann die Dimensionsausprägung mit dem Attribut „*summiert=ja*“ kennzeichnen, wenn diese Dimensionsausprägung als Variable im Bezugsteil einer Summenformel genutzt wird. Dimensionsausprägungen mit dem Attribut „*summierbar=nein*“ drücken aus, dass Modellzel-

len, an denen eine solche Dimensionsausprägung beteiligt ist, nicht mit einer Summenformel belegt werden können. Entsteht eine Modellzelle durch die Kombination einer Dimensionsausprägung mit dem Attribut „*summiert=ja*“ und einer Dimensionsausprägung mit dem Attribut „*summierbar=nein*“, kann eine Summenformel nicht ohne inhaltliche Widersprüche zu verursachen auf eine Zelle angewandt werden. Dieses Problem ergibt sich z.B. bei einer Modellzelle, die durch Kombination der Dimensionsausprägungen *SummeÜberZeit*, *ProduktA*, *Kunde1* und *Rabattsatz* entsteht und auf die eine Formel *SummeÜberZeit = 2011 + 2012* angewandt werden soll. Die Dimensionsausprägung *Rabattsatz* würde mit dem Attribut „*summierbar=nein*“ und die Dimensionsausprägung „*SummeÜberZeit*“ mit dem Attribut „*summiert=ja*“ versehen werden. Es ist inhaltlich in dieser Konstellation nicht sinnvoll, Rabattsätze zu addieren, sodass die Summenformel auf diese Zelle nicht angewandt wird. Stattdessen müsste für diese Modellzelle beispielsweise eine Formel einen gewichteten Durchschnitt der Rabattsätze auf Basis von Verkaufsmengen spezifizieren. Sofern dies nicht der Fall ist, kann der Anwender in der Transformationsphase auf diesen Umstand aufmerksam gemacht werden und entsprechende Anpassungen bzw. Definitionen im konzeptuellen Modell vornehmen. Eine weitere Möglichkeit wäre es, die Zelle aus dem Modell im Ziel-Anwendungssystem zu löschen, indem man sie entsprechend kennzeichnet oder blockiert.

```
<xs:element name="dim-item">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
    <xs:attribute name="cid" type="xs:string" use="required"/>
    <xs:attribute name="summierbar" type="xs:string" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="ja"/>
          <xs:enumeration value="nein"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="summiert" type="xs:string" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="ja"/>
          <xs:enumeration value="nein"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="dimension" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

Abbildung 5-6: XML Schema Definition des Elements Dimensionsausprägung

Dimensionsausprägungen beziehen sich inhaltlich auf eine von der jeweiligen Dimension vorgegebene Thematik und stellen eine Unterteilung einer Dimension dar. Die Anzahl von Dimensionsausprägungen je Dimension ist grundsätzlich unbegrenzt, jedoch sollte im Sinne des Konzeptes MOO nach Zhang und Weber (minimale ontologische Überschneidung; vgl. Abschnitt 2.2.2) jede Dimensionsausprägung eine Modellgröße repräsentieren, die sich von den übrigen Dimensionsausprägungen eines Modells möglichst genau abgrenzen lässt. Abbildung 5-7 stellt ein Beispiel einer Dimensionsausprägung *2011* dar, die Teil einer Dimension *Zeit* ist:

```
<dim-item id="Zeit_100" cid="Zeit2011" summierbar="ja" summiert="nein" dimension="Zeit">
  <name>2011</name>
</dim-item>
```

Abbildung 5-7: XML-Element einer Dimensionsausprägung

Im Unterschied zu einer Dimension verfügt eine Dimensionsausprägung über ein zusammengesetztes Attribut *ID*, in welchem der Bezeichner der Dimension als Präfix enthalten

ist. Dem Präfix folgt ein Unterstrich, dem drei Ziffern als fortlaufende Nummer folgen. Das Attribut *cid* spiegelt eine Zusammensetzung des Dimensions- und des Ausprägungsbezeichners wider, und das Attribut *dimension* enthält den Bezeichner der Dimension, zu welcher die Ausprägung gehört. Diese teilweise redundanten Informationen werden aus Gründen der Performance-Steigerung vorgenommen und ermöglichen eine einfachere Weiterverarbeitung der Modellinformationen. Bezüglich des Speicherplatzes und der Übersichtlichkeit sind sie als unkritisch zu erachten.

5.2.4 XML-Modellkomponente Gruppe

Das XML Element Gruppe (*group*) kann Dimensionen oder Dimensionsausprägungen zusammenfassen und wird dafür genutzt, eine definierte Menge dieser Elemente direkt ansprechen zu können. Abbildung 5-8 zeigt die XML Schema-Definition dieses Elements. Zusätzlich zu einem Bezeichner (*name*) wird über eine *choice*-Umgebung sichergestellt, dass entweder Dimensionen (siehe Abbildung 5-8, Zeile 11: *name="dim-ref"*) oder Ausprägungen (siehe Abbildung 5-8, Zeile 6: *name="dim-item-ref"*) gruppiert werden, wobei die Anzahl der jeweiligen Art über das Attribut *maxOccurs="unbounded"* nicht begrenzt wird. Die Elemente *dim-ref*, im Falle von Dimensionen, und *dim-item-ref*, im Falle von Dimensionsausprägungen, verfügen über ein Attribut *ref*, das eine Referenz auf eine Dimension (siehe Abbildung 5-8, Zeile 6: type="xs:IDREF") bzw. eine Dimensionsausprägung (siehe Abbildung 5-8, Zeile 11: type="xs:IDREF") darstellt. Es enthält dementsprechend die ID des referenzierten Modellelements. Zudem wird im Attribut *contentType* festgehalten, ob es sich um Dimensionen oder Dimensionsausprägungen innerhalb dieser Gruppe handelt. Im Element *simpleType* wird durch zwei *enumeration*-Elemente zudem festgehalten, dass die erlaubten Werte des Attributs contentType *dimension* und *dim-item* lauten.

```
<xs:element name="group">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:choice>
Zeile 6..... <xs:element maxOccurs="unbounded" name="dim-item-ref">
          <xs:complexType>
            <xs:attribute name="ref" type="xs:IDREF" use="required"/>
          </xs:complexType>
        </xs:element>
Zeile 11..... <xs:element maxOccurs="unbounded" name="dim-ref">
          <xs:complexType>
            <xs:attribute name="ref" type="xs:IDREF" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
    <xs:attribute name="contentType" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="dimension"/>
          <xs:enumeration value="dim-item"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Abbildung 5-8: XML Schema Definition des Elements Gruppe

Von besonderer Bedeutung sind Gruppen von Dimensionsausprägungen. Diese können z.B. die Definition von Formeln für Teilmengen-differenzierte Kennzahlenspezifikationen reduzieren. Wie dies genau funktioniert, wird im folgenden Abschnitt 5.2.5 thematisiert. Um das Konzept möglichst flexibel zu gestalten und spätere Erweiterungen zu erleichtern, ist es ebenfalls möglich komplette Dimensionen zu gruppieren.

Eine gemischte Gruppe mit Dimensions- und Dimensionsausprägungselementen ist nicht vorgesehen, Dimensionsausprägungen können aber aus unterschiedlichen Dimensionen stammen. Abbildung 5-9 zeigt eine beispielhafte XML-Spezifikation einer Gruppe von Dimensionsausprägungen:

```
<group id="group_1" contentType="dim-item">
  <name>Group_100</name>
  <dim-item-ref ref="Kennzahlen_000"/>
  <dim-item-ref ref="Zeit_100"/>
</group>
```

Abbildung 5-9: XML-Element einer Gruppe

Die Gruppe mit der Bezeichnung *Group_100* setzt sich aus zwei Referenzen auf die Dimensionsausprägungen mit den IDs *Kennzahlen_000* und *Zeit_100* zusammen. Das Attribut *contentType="dim-item"* gibt an, dass diese Gruppe Dimensionsausprägungen zusammenfasst.

5.2.5 XML-Modellkomponente Formel

Die bislang beschriebenen Elemente sind für die Abbildung der Struktur eines konzeptuellen Modells bestimmt und geben den Rahmen und die Möglichkeiten vor, die für die Modellierung logischer Strukturen und Zusammenhänge anhand von Formeln zur Verfügung stehen. Die folgenden Ausführungen beziehen sich auf diesen logischen Teil der konzeptuellen Modelle.

Bei der Modellierung logischer Strukturen werden die zuvor definierten Elemente eines Modells miteinander in Beziehung gesetzt. Innerhalb einer Formel können Dimensionsausprägungen als Variablen angesehen werden, die im mehrdimensionalen Raum einzelne Zellen arithmetisch miteinander verknüpfen (siehe Abschnitt 3.2.2).

Die XML Schema-Definition des Formelelements ist sehr komplex, da der Anwender in der Möglichkeit der Zusammenstellung einer Formel nicht eingeschränkt werden soll, gleichzeitig aber alle Möglichkeiten in der XML Schema-Definition erfasst werden müssen. Aufgrund der Komplexität wird die XML Schema-Definition des Formelelements daher in vier Teilen dargestellt.

```
<xs:element name="formula">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:choice>
        <xs:sequence>
          <xs:choice>
            <xs:element name="formula-string"/>
            <xs:element name="formula-string-original"/>
          </xs:choice>
```

Abbildung 5-10: XML Schema Definition des Elements Formel – Teil 1

Abbildung 5-10 zeigt den ersten Teil möglicher Kind-Elemente eines Formelelements. Jede Formel besitzt entweder ein Element *formula-string* oder ein Element *formula-string-original*, in welchem die vollständige Formeldefinition als String enthalten ist, z.B. *Zeit_Summe = 2010 + 2011 + 2012*. Eine choice-Umgebung stellt sicher, dass nur eines der beiden Elemente vertreten sein darf. Die Unterscheidung der beiden Elemente macht deutlich, ob eine Transformation bereits stattgefunden hat (formula-string-original) und demensprechend eine Anpassung des Bezugsteils der Formel vorgenommen wurde oder nicht (formula-string).

Teil 2 der XML Schema-Definition des Formelelements legt die Möglichkeiten zur Spezifizierung des Bezugsteils einer Formel (siehe Abschnitt 3.2.2) fest. Für eine möglichst exakte Modellierungsmöglichkeit von Verknüpfungen in Formeln (siehe Abschnitt 3.3.5) können mehrere Dimensionsausprägungen miteinander verbunden werden. In einem XML-Dokument werden hierfür die beiden Elemente *scopeItem* sowie *mainItem* bereitgestellt. Ein scopeItem dient für die Spezifikation des Bezugsteils einer Formel, d.h. auf welche Zellen eine Formel angewandt werden soll. Ein mainItem hingegen dient der Definition eines bestimmten Zellbezugs im Logikteil einer Formel. Dimensionsausprägungen innerhalb eines scopeItem oder eines mainItem werden aus Übersichtsgründen immer durch einen Doppelpunkt getrennt, der in einem XML-Element *formula-item* definiert ist. Die XML Schema-Definition dieser beiden XML-Elemente sind in Abbildung 5-11 (scopeItem) und Abbildung 5-14 (mainItem) enthalten.

Der Bezugsteil einer Formel

Hinsichtlich der Modellierung des Bezugsteils einer Formel sind zwei Arten zu unterscheiden: Eine Alternative ist es, direkt nach dem formula-string Element ein scopeItem zu definieren. Dieses enthält eine oder mehrere Dimensionsausprägungen unterschiedlicher Modelldimensionen und beschreibt dadurch eine Zellmenge. In diesem Fall muss mindestens eine Dimensionsausprägung im scopeItem vertreten sein, die aus der aktuellen Dimension stammt (welcher die Formel angehört).

Eine weitere Alternative ist ein *IN-Statement*. Ein IN-Statement besteht aus zwei scopeItem Elementen und dem Schlüssel-Element *IN* („IN:“). Das erste scopeItem Element enthält eine Liste von Dimensionsausprägungen und/oder eine Gruppe. Es beschreibt eine Zellmenge *A* des Modells. Das zweite scopeItem, welches aus genau einer Dimensionsausprägung aus der aktuellen Dimension sowie höchstens einer Dimensionsausprägung je weiterer Modelldimension besteht, beschreibt eine weitere Zellmenge *B* des Modells. Die

beiden scopeItems und damit die beiden Zellmengen sind durch ein Komma getrennt („IN:Zellmenge *A*, Zellmenge *B* = …“). Inhaltlich wird mit einem IN-Statement eine Teilmenge eines Modells definiert (Zellmenge *A*, erstes scopeItem). Die Formel wird auf eine Zellmenge *B* (zweites scopeItem) angewandt, die Teil der Zellmenge *A* ist. Es ergibt sich die Möglichkeit eine Gruppe zur Definition des Bezugsteils zu nutzen (Zellmenge *A*). Auf diese Weise können mehrere Dimensionsausprägungen aus einer Dimension für die Definition des Bezugsteils genutzt werden (z.B. *2011*, *2012* und *2013* aus einer Zeitdimension). Dadurch kann der Anwender den Bezugsteil einer Formel im Sinne einer freien Modellierung flexibel und präzise gestalten.

```
<xs:element name="formula-element" maxOccurs="2" minOccurs="0">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="IN"/>
      <xs:enumeration value=":"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Schlüssel-Element "IN" zur Einleitung eines IN-Statements

```
<xs:sequence maxOccurs="2">
  <xs:element name="scopeItem">
    <xs:complexType mixed="true">
      <xs:sequence maxOccurs="unbounded">
        <xs:choice>
          <xs:element name="group-ref">
            <xs:complexType>
              <xs:attribute name="ref" type="xs:IDREF" use="required"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="dim-item-ref">
            <xs:complexType>
              <xs:attribute name="ref" type="xs:IDREF" use="required"/>
            </xs:complexType>
          </xs:element>
        </xs:choice>
        <xs:element name="formula-element" maxOccurs="unbounded" minOccurs="0">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value=":"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

scopeItem, bestehend aus Referenzen auf Dimensionsausprägungen oder Gruppen, jeweils getrennt durch einen ":"

Abbildung 5-11: XML Schema Definition des Elements Formel – Teil 2

Abbildung 5-11 zeigt den zweiten Teil der XML Schema-Definition, welcher die Möglichkeiten zur Modellierung des Bezugsteils einer Formel festlegt. Teil 2 schließt sich unmittelbar an Teil 1 der XML Schema-Definition des Elements Formel (siehe Abbildung 5-10) an. Die Angabe des Attributs *minOccurs=0* im ersten formula-element (oberer Kasten) ermöglicht es, ein IN-Statement („IN:“) einzuleiten oder auf diese Angabe zu verzichten. Das zweite scopeItem Element zur Definition eines IN-Statements, wird durch das Attribut *maxOccurs=2* des sequence-Elements (zwischen den beiden Kästen) ermöglicht. Mittels des choice-Elements wird die Möglichkeit unterbreitet Gruppen-, oder Dimensionsausprägungselemente als Kind-Element eines scopeItem zu definieren. Dies geschieht jeweils mittels einer Referenz auf die ID des entsprechenden Elements (siehe Abbildung 5-11: xs:IDREF).

Der Logikteil einer Formel

Im Anschluss an den Bezugsteil folgt die Repräsentation des Logikteils einer Formel. Die Variablen einer Formel werden dabei mittels des Elements mainItem auf XML-Ebene definiert. Die Operatoren, welche die Interaktion zwischen den Variablen einer Formel beschreiben, werden durch das XML-Element formula-element definiert, welches den jeweiligen Operator als String oder ggf. eine Zahl beinhaltet. An dieser Stelle wäre es ebenfalls denkbar, einzelne Operatoren und andere Formelbestandteile durch jeweils eigene XML-Elemente zu repräsentieren. Da ein spezifischer Zugriff der Operatoren im weiteren Verlauf des Konzepts jedoch nicht vorgesehen ist und keinen Mehrwert für die Verarbeitung bedeutet, werden eigene Operator-Elemente im Sinne des schlankeren Designs verworfen. Zusätzlich zu den Basis-Operatoren Plus (+), Minus (-), Mal (*), Geteilt (/), einer Potenz (^) sowie einer öffnenden und einer schließenden Klammer, werden dem Benutzer die zwei speziellen Funktionsoperatoren *IF-THEN-ELSE* und *PREV/NEXT* zur Verfügung gestellt. Abbildung 5-12 stellt ein Beispiel für eine *IF-THEN-ELSE* Klausel dar:

```
<formula-element>=</formula-element>
<formula-element>IF</formula-element>
<formula-element>{</formula-element>
<mainItem>
  <dim-item-ref ref="Kennzahlen_002"/>
</mainItem>
<formula-element>></formula-element>
<formula-element>10</formula-element>
<formula-element>;</formula-element>
<formula-element>100</formula-element>
<formula-element>;</formula-element>
<formula-element>120</formula-element>
<formula-element>}</formula-element>
```

Abbildung 5-12: Beispiel einer IF-THEN-ELSE Klausel in einer Formel

Durch ein mainItem Element wird eine Variable definiert, deren Wert mit einem Referenzwert unter Angabe eines Operators (möglich sind <, >, <=, >= und =) verglichen wird. Angenommen, dass die referenzierte Dimensionsausprägung mit der ID *"Kennzahlen_002"* als Menge bezeichnet wird, lautet die IF-THEN-ELSE Klausel im vorliegenden Fall *IF{Menge>10;100;120}*. Dem Referenzwert folgt der *THEN*-Wert, der bei einem positiven Prüfungsergebnis angewandt wird und im vorliegenden Fall durch das formula-element mit dem Inhalt *100* repräsentiert wird. Abschließend folgt der *ELSE*-Wert, der bei negativem Prüfungsergebnis angewendet wird. Alle Werte und Operatoren werden im XML-Dokument durch das Element formula-element repräsentiert. Referenz-, THEN- und ELSE-Wert werden zudem durch ein Semikolon voneinander getrennt.

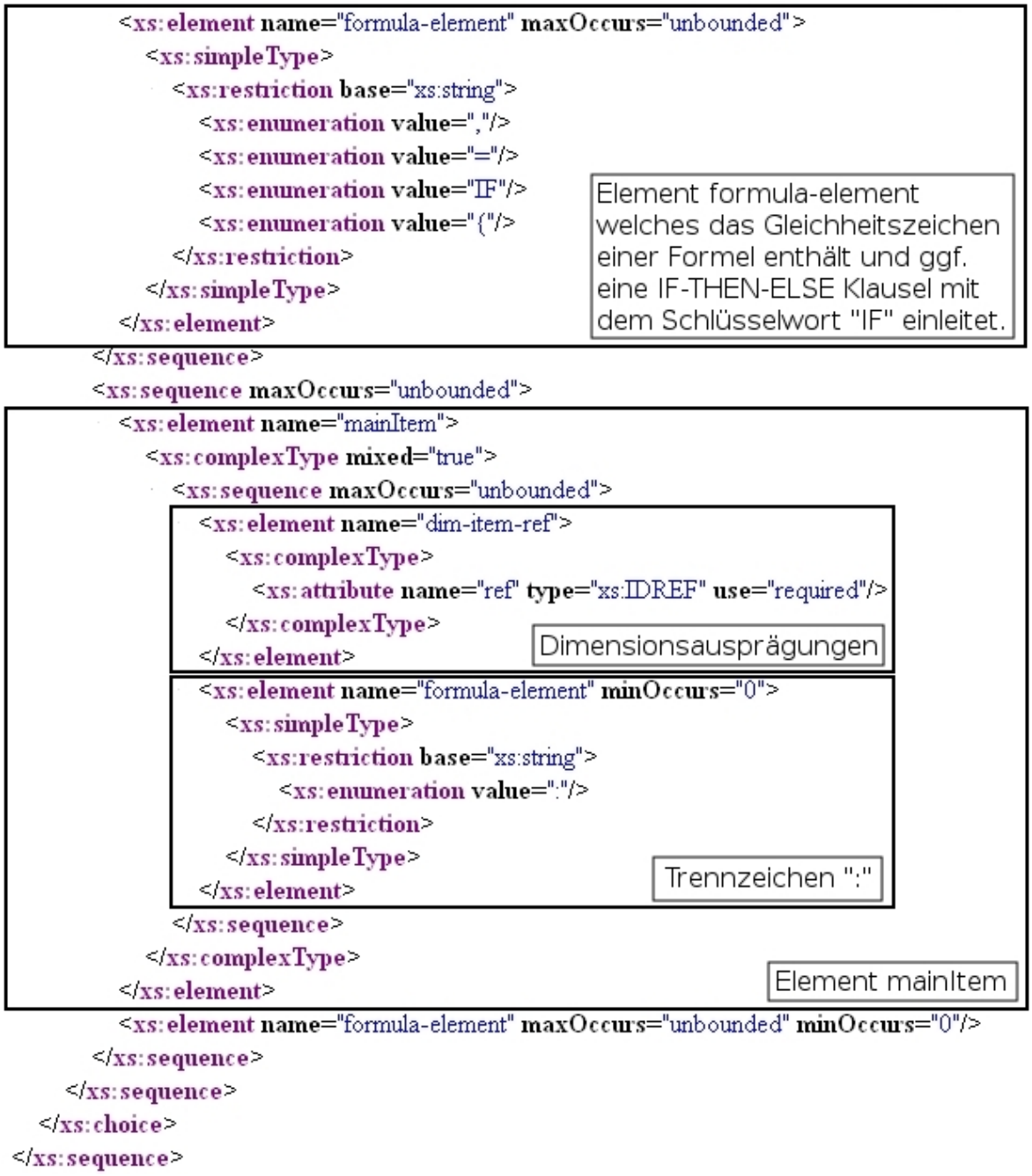
Der zweite spezielle Funktionsoperator beeinflusst die Definition einer Variablen in einer Formel. Mittels der Schlüsselwörter *PREV* (engl. für previous: vorausgehend) und *NEXT* (engl.: nächstfolgend) kann definiert werden, dass eine Variable sich auf eine vorhergehende (PREV) oder nachfolgende (NEXT) Dimensionsausprägung einer anderen Modelldimension bezieht. Die Reihenfolge der Dimensionsausprägungen in einer Dimension spielt in diesem Fall eine wichtige Rolle. In der Regel wird dieser Operator mit Bezug auf eine Zeit-Dimension angewandt. Abbildung 5-13 zeigt ein Beispiel eines PREV-Operators in einem XML-Dokument:

```
<mainItem>
  <dim-item-ref ref="Kennzahlen_002"/>
  <formula-element dimension="Zeit">PREV</formula-element>
</mainItem>
```

Abbildung 5-13: Beispiel eines PREV-Operators

Angenommen ein Beispielmodell verfügt über eine Zeitdimension mit den Ausprägungen *2011* und *2012* sowie einer Dimension Kennzahlen mit den Ausprägungen *Umsatz*, *Menge* und *Preis*. Die Ausprägung *Menge* wird in Abbildung 5-13 durch das *dim-item-ref* Element mit dem Attribut *ref=Kennzahlen_002* repräsentiert. Wird in einer Formel *Umsatz = Menge * Preis* die Variable *Menge* mit dem Schlüsselwort *PREV* bezüglich der (korrespondierenden) Dimension *Zeit* (siehe Abbildung 5-13: Attribut *dimension* des Elements formula-element) gekennzeichnet, wird für die Berechnung der Zelle *Umsatz:2012* nicht der Wert der Zelle *Menge:2012* herangezogen, sondern der Wert der *Zelle Menge:2011*. Für ein konsistentes Modell folgt daraus, dass der Bezugsteil der Formel in diesem Fall auf die Zelle *Umsatz:2012* eingegrenzt und die Zelle *Umsatz:2011* durch eine weitere Formel definiert werden muss. Das Schlüsselwort NEXT kehrt diese Wirkung um und sorgt für einen Bezug der gekennzeichneten Variable auf eine folgende Ausprägung der korrespondierenden Dimension, statt einer vorhergehenden, wie beim PREV-Operator.

Abbildung 5-14 zeigt den dritten Teil der XML Schema-Definition einer Formel und schließt sich unmittelbar an Teil 2 (siehe Abbildung 5-11) an. In diesem Teil werden die möglichen Inhalte des Logikteils einer Formel definiert. Das erste formula-element Element (erster Kasten) enthält ein Gleichheitszeichen und stellt ggf. die Einleitung einer IF-THEN-ELSE Klausel dar. Der zweite Teil besteht aus der Deklaration der Variablen einer Formel, die jeweils durch das XML-Element mainItem repräsentiert werden. Dieses besteht aus höchstens einer Referenz auf eine Dimensionsausprägung je vorhandener Modelldimension. Die einzelnen Dimensionsausprägungen in einem mainItem-Element sind jeweils durch einen „:“ getrennt, der in einem formula-element Element definiert ist.




```
        <xs:element name="formula-element" maxOccurs="unbounded">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value=","/>
              <xs:enumeration value="="/>
              <xs:enumeration value="IF"/>
              <xs:enumeration value="("/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="mainItem">
          <xs:complexType mixed="true">
            <xs:sequence maxOccurs="unbounded">
              <xs:element name="dim-item-ref">
                <xs:complexType>
                  <xs:attribute name="ref" type="xs:IDREF" use="required"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="formula-element" minOccurs="0">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value=":"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="formula-element" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:sequence>
  </xs:choice>
</xs:sequence>
```

Abbildung 5-14: XML Schema Definition des Elements Formel – Teil 3

Es ist im Logikteil einer Formel nicht vorgesehen, Gruppen zu referenzieren, weil eine Variable einer Formel eine inhaltliche Aussage enthält, die sich in einer konkreten Referenz einer Zelle niederschlägt. Wären an dieser Stelle mehrere Ausprägungen aus einer Dimension möglich, wäre ein eindeutiger Zellbezug nicht in jedem Fall gegeben.

Das Element formula-Element unter der Definition des Elements mainItem in Abbildung 5-14 übernimmt mehrere Funktionen. Es definiert den Prüfungsteil einer IF-THEN-ELSE Klausel und sämtliche Operatoren zur Verknüpfung der mainItems (Variablen) des Logikteils einer Formel. Da das Element auch für die Abbildung des Prüfungsteils einer IF-

THEN-ELSE Klausel herangezogen wird, müssen unter anderem auch Zahlen dargestellt werden können (z.B. *IF{Menge > 100(…)}*). Aus diesem Grund können die zulässigen Werte des Elements formula-element nicht beschränkt werden. Des Weiteren wird anhand des Elements formula-item der optionale *Skip*-Teil einer Formel eingeleitet.

Der Skip-Teil einer Formel

Skip (engl.: auslassen, überspringen) fungiert als Schlüsselwort mit Verbindung zum Bezugsteil einer Formel. An dieser Stelle können (noch einmal) einzelne Zellen oder Zellmengen innerhalb von mainItem Elementen definiert werden, die im Bezugsteil der Formel eigentlich enthalten sind, jedoch explizit aus dem Bezugsteil ausgeschlossen werden sollen. Für die Umsetzung einer Teilmengen-differenzierten Kennzahlenspezifikation bietet dieses Vorgehen den Vorteil, dass ein Bezugsteil einer Formel in der Regel relativ generell definiert und die Spezifikation (typischerweise weniger) auszuschließender Zellen getrennt erfolgen kann. Die Definition des Skip-Teils einer Formel sollte daher kurz gehalten werden, da größere Einschränkungen (besser) bereits im Bezugsteil einer Formel berücksichtigt werden können. Abbildung 5-15 zeigt beispielhaft einen Skip-Teil einer Formel, der alle Modellzellen aus dem Bezugsteil einer Formel ausschließt, die aus einer Kombination der Dimensionsausprägungen *Kennzahlen_000* und *Zeit_100* bestehen.

```
<mainItem>
  <dim-item-ref ref="Kennzahlen_000"/>
  <formula-element>:</formula-element>
  <dim-item-ref ref="Zeit_100"/>
</mainItem>
</formula>
```

Abbildung 5-15: Beispiel eines Skip-Teils einer Formel

Der vierte und letzte Teil der XML Schema-Definition des Elements Formel schließt sich unmittelbar an den dritten Teil (siehe Abbildung 5-14) an. Er stellt die Attribute *id*, *cid* und *case* einer Formel dar. Das Attribut case kann die Werte 1 bis 8 annehmen. Es repräsentiert den Modellierungsfall einer Formel im Kontext des Modells, entsprechend der Klassifizierung multidimensionaler Modellierung (siehe Kapitel 4.3). Abbildung 5-16 illustriert diese Zusammenhänge:

```
<xs:attribute name="id" type="xs:ID" use="required"/>
<xs:attribute name="cid" type="xs:string" use="required"/>
<xs:attribute name="case" minOccurs="0">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="1"/>
      <xs:enumeration value="2"/>
      <xs:enumeration value="3"/>
      <xs:enumeration value="4"/>
      <xs:enumeration value="5"/>
      <xs:enumeration value="6"/>
      <xs:enumeration value="7"/>
      <xs:enumeration value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
```

Abbildung 5-16: XML Schema Definition des Elements Formel – Teil 4

Die komplexe Definition des Formelelements auf Ebene des XML Schemas ist mit Teil 4 abgeschlossen. Abbildung 5-17 unterbreitet ein Beispiel einer Formel *Zeit_Summe = 2011 + 2012 Skip Rabattsatz* im XML-Format. *Zeit_Summe* entspricht dem referenzierten Element *Zeit_102*, *2011* entspricht *Zeit_100*, *2012* entspricht *Zeit_101* und *Rabattsatz* entspricht dem Element *Kennzahlen_005*.

```
<formula id="formula_100" cid="_100" case="1">
  <formula-string> ZeitSumme = 2011 + 2012</formula-string>
  <scopeItem>
    <dim-item-ref ref="Zeit_102"/>
  </scopeItem>
  <formula-element>=</formula-element>
  <mainItem>
    <dim-item-ref ref="Zeit_100"/>
  </mainItem>
  <formula-element>+</formula-element>
  <mainItem>
    <dim-item-ref ref="Zeit_101"/>
  </mainItem>
  <formula-element>Skip</formula-element>
  <formula-element>:</formula-element>
  <mainItem>
    <dim-item-ref ref="Kennzahlen_005"/>
  </mainItem>
</formula>
```

Abbildung 5-17: XML-Element einer Formel

5.3 Unterstützung des Modellierungsprozesses

Nach der Erläuterung der Regeln der Modellierung in Form der XML Schema-Definition kann in diesem Abschnitt die Darstellung von Unterstützungsfunktionen während des Modellierungsprozesses erfolgen. Diese weisen teilweise einen ausgeprägten Bezug auf die Elemente der konzeptuellen Modellierungssprache auf.

5.3.1 Unterstützungsfunktionen im Modellierungseditor

Im Modellierungseditor stehen vor allem die Regeln der Modellierung im Fokus, die aus der XML Schema-Definition hervorgehen. Alle im Modellierungseditor entwickelten konzeptuellen Modelle, auch solche die noch nicht transformiert wurden und dementsprechend Überschneidungen mehrerer Formeln in Zellen enthalten können, sollen gegenüber dem XML Schema valide sein. Validität beschreibt in diesem Zusammenhang die Konformität der XML-Strukturen eines Modells entsprechend der im XML Schema definierten Regeln. Das bedeutet, dass ein Element nur die im XML Schema definierten Attribute besitzen darf, die Schachtelung von Elter- und Kind-Elementen den Vorgaben entsprechen muss und die gültige Anzahl der Elemente auf den unterschiedlichen Ebenen der Baumstruktur eingehalten wird. Der Modellierungseditor schließt nicht erlaubte Kombinationen von Elementen daher von vornherein aus. Des Weiteren dürfen Änderungen an Modellelemen-

ten nicht zu Inkonsistenzen hinsichtlich Eindeutigkeit oder Vollständigkeit des Modells führen. Demnach müssen z.B. Referenzen auf Dimensionsausprägungen in Gruppen entfernt werden, sobald diese Ausprägungen gelöscht werden. In diesem Zusammenhang müssen Formeln, welche die entsprechende Ausprägung referenzieren, ebenfalls angepasst oder ggf. gelöscht werden.

Hinsichtlich der Klassifizierung multidimensionaler Modellierungsfälle (siehe Abschnitt 4.3) bleibt festzustellen, dass in der Konfigurationsphase des Modellierungsprozesses die Klassifizierung nicht oder lediglich sehr eingeschränkt sinnvoll genutzt werden kann. Dies hängt damit zusammen, dass die Formeln und deren Fallklassifizierungen sich untereinander beeinflussen (siehe Kapitel 4.3.1). Wenn weitere Formeln in ein Modell eingefügt werden, können sich die Fälle der vorhandenen Formeln ändern, was Auswirkungen auf die bereitgestellten Informationen hat. Die Auswertung der Fälle multidimensionaler Modellierung (siehe Kapitel 4.3.3) erlaubt es in dieser Phase zumindest, den Anwender hinsichtlich der Vollständigkeit des Modells zu unterstützen: Liegen heterogene Strukturen vor (Fälle 2, 4, 6 und 8) kann der Editor automatisiert überprüfen, ob sämtliche Intervalle, die durch Einschränkungen im Bezugsteil einer Formel entstehen, durch eine Formel definiert sind. Ist dies nicht der Fall, kann der Anwender auf die möglicherweise unvollständige Definition hingewiesen werden.

Eine ähnliche Unterstützungsleistung kann der Modellierungseditor dem Anwender durch Auswertung der beiden Attribute *summierbar* und *summiert* einer Dimensionsausprägung bereitstellen. Für diejenigen Modellzellen, die durch Kombination einer Dimensionsausprägungen mit der Eigenschaft „summiert=ja“ und einer Dimensionsausprägung mit der Eigenschaft „summierbar=nein“ entstehen, kann der Anwender auf die Notwendigkeit einer gesonderten Formel aufmerksam gemacht werden. Wird diese als nicht nützlich erachtet, kann der Editor anhand der Definition eines Skip-Teils die Möglichkeit anbieten, entsprechende Modellzellen auszuschließen.

Schließlich kann der Modellierungseditor beim Vorliegen von PREV- bzw. NEXT-Bezügen in einer Formel die entsprechenden Zellbereiche, die von der Formel ausgeschlossen werden müssen, berechnen und sie in den Skip-Teil der Formel einfügen. Außerdem kann die Möglichkeit der Definition einer weiteren Formel für die erste (im Falle von PREV) oder letzte (im Falle von NEXT) Ausprägung der korrespondierenden Dimension angeboten werden.

5.3.2 Unterstützungsfunktionen für die Transformationskomponente

Die Transformationskomponente unterstützt den Anwender bei der Erstellung überschneidungsfreier Modellstrukturen. In diesem Zusammenhang steht dementsprechend die Eindeutigkeit der Modellierung im Fokus. Die Transformationskomponente kann dem Anwender über eine Dialogfunktion sämtliche Modellzellen präsentieren, die mit mehr als einer Formel belegt sind. Dies erfolgt idealerweise begleitet durch eine Visualisierung der Modellstrukturen, damit der Anwender zu jeder Zeit einen Überblick hat, an welcher Stelle des Modells sich die betrachtete Zelle befindet. Die Dialogkomponente bietet des Weiteren die Möglichkeit, die Auswirkungen der Auswahl der identifizierten, alternativen Formel zu beurteilen. In diesem Zusammenhang ist es sehr wichtig, dass die Transformationskomponente über eine eigene Modell-Berechnungs-Komponente verfügt, welche auf Basis von Testdaten oder Anwender-Eingaben die korrekten Werte für die betrachtete und sämtliche abhängige Zellen berechnen kann. Des Weiteren kann der Anwender bei der Transformation darauf aufmerksam gemacht werden, wenn eine bestimmte Auswahl einer Formel zu simultanen Bezügen führt. Für die betrachtete Zelle kann er unter Umständen eine andere Formel wählen, die dies verhindert, so dass kein Algorithmus für die Berechnung der Zellen eines simultanen Bezugs formuliert werden muss.

Da in der Transformationshase von größeren Modellen unter Umständen sehr viele Zellen nicht eindeutig definiert sind, kann die Transformationskomponente dem Benutzer auf Basis der Auswertung der Attribute *summierbar* und *summiert* sowie der weiteren, nicht eindeutig definierten Zellen eine Zellmenge vorschlagen, auf welche die vom Benutzer ausgewählte Formel angewandt werden kann. Des Weiteren kann der Anwender den vorgeschlagenen Zellbereich, auf welchen eine ausgewählte Formel angewandt wird, ändern oder neu definieren. Die Transformationskomponente erkennt in diesem Zusammenhang dadurch eventuell induzierte Probleme und macht den Anwender darauf aufmerksam, z.B. wenn definierte Teilbereiche Zellen beinhalten, die aus einer Dimensionsausprägung mit dem Attribut „summierbar=nein“ und einer Dimensionsausprägung mit dem Attribut „summiert=ja“ bestehen. Dies ist notwendig, weil solche Bereiche während der ersten Phase des Modellierungsprozesses i.d.R. durch eine gesonderte Formel definiert werden.

5.3.3 Unterstützungsfunktionen für die Kompilierungskomponente

Die Kompilierungskomponente übersetzt die überschneidungsfrei definierten Modelle in vorhandene Ziel-Anwendungssysteme und integriert sie dadurch in eine vorhandene Systemlandschaft.

Die Kompilierungskomponente hält für jede Übersetzung des Modells in eine Anwendungssoftware eine spezifische Schnittstelle bereit. Diese beinhaltet sämtliche Konzepte der Ziel-Anwendung und bildet die konzeptuell modellierten Strukturen weitest möglich auf diese ab. Der Anwender muss in den Übersetzungsprozess eingebunden werden, sobald Informationen notwendig sind, die im konzeptuellen Modell nicht vorliegen. Dies ist beispielsweise der Fall, wenn die Software keine Teilmengen-differenzierte Kennzahlenspezifikation unterstützt. In diesem Fall können die Formeln vom Anwender in ein unterstütztes Format überführt werden. Überschüssige Informationen, wie der Skip-Teil einer Formel oder IN-Statements würden automatisch entfernt werden.

Viele Anwendungen, die für die Umsetzung von MDSS genutzt werden, unterstützen keine simultanen Bezüge. In diesem Fall wertet die Kompilierungskomponente die Fall-Attribute der Modellformeln aus und gibt dem Anwender die Möglichkeit, die Formeln unter Verwendung der konzeptuellen Modellierungssyntax anzupassen. Der Anwender kann dabei auf Formeldefinitionen, die nicht konform mit den Konzepten des Ziel-Anwendungssystems sind, aufmerksam gemacht werden.

Die Informationen der Fälle multidimensionaler Modellierung können außerdem für die Ermittlung der korrekten Berechnungsreihenfolge der Modellzellen genutzt und hierfür der Modell-Berechnungs-Komponente des Ziel-Anwendungssystems zur Verfügung gestellt werden. In diesem Zusammenhang kann beispielsweise die Information abgeleitet werden, dass Zellen mit Formeln, in denen Dimensionsausprägungen mehrerer Dimensionen genutzt werden, nach solchen Zellen berechnet werden, die mit Formeln belegt sind, in denen lediglich Dimensionsausprägungen aus einer Dimension enthalten sind. Des Weiteren können die Informationen der Modellierungsfälle auch für die Herstellung der korrekten Reihenfolge der definierten Formeln genutzt werden, sofern das Ziel-Anwendungssystem keine Funktion zum Feststellen einer korrekten Formelreihenfolge bietet.

Teil III

Prototypische Implementierung und Evaluation des konzeptuellen Modellierungsansatzes

6 Prototypische Implementierung

Teil II der vorliegenden Arbeit stellte ein Konzept zur freien, transparenten konzeptuellen Modellierung multidimensionaler Strukturen für MDSS vor. Der im Folgenden beschriebene Prototyp (Mod^{2}DSS)[21] setzt das Konzept um und schafft dadurch eine Möglichkeit, Machbarkeit und Vorteile zu beurteilen. Angefangen mit der Darstellung der Entwicklungsumgebung, in welcher der zweistufige Modellierungsprozess implementiert wurde (Kapitel 6.1), wird der Modellierungseditor (Kapitel 6.2) eingehendend erläutert. Die Implementierung dieses Teils stützt sich auf die Entwicklungsarbeit, die im Rahmen einer Bachelorarbeit[22] am Fachgebiet Management Support und Wirtschaftsinformatik der Universität Osnabrück geleistet wurde. Darauf folgend werden die XML-Schnittstelle (Kapitel 6.3) sowie die Transformations- und Kompilierungskomponente (Kapitel 6.4) betrachtet. Abschließend gibt Kapitel 6.5 einen Überblick der Limitationen des Prototyps.

6.1 Entwicklungsumgebung

Die Entwicklung des Prototyps wurde mittels unterschiedlicher Programmiersprachen und Plattformen umgesetzt. Sofern von Entwicklungsumgebung gesprochen wird, ist im Folgenden das Zusammenspiel diverser Komponenten gemeint. Konkret sind dies die Programmiersprache *Java*, die Programmiersprache *Visual Basic for Applications* (VBA) in Verbindung mit *Microsoft Excel* und *XML*[23], deren Eignung in der Folge kurz beschrieben wird.

Java ist eine von *Sun Microsystems* entwickelte Programmiersprache, welche sich durch *Objektorientierung*, *Plattformunabhängigkeit* und *universelle Einsetzbarkeit* auszeichnet (vgl. Ullenboom, 2009, S. 57 ff.). Plattformunabhängigkeit und Objektorientierung sind zwei wesentliche Punkte, die bei der Wahl des Werkzeugs zur Implementierung des Modellierungseditors den Ausschlag für Java gegeben haben. Java-Applikationen können auf annähernd allen Betriebssystemen genutzt werden, da zumeist eine Java-Unterstützung in Form sogenannter Laufzeitumgebungen vorliegt (vgl. Flanagan, 2003, S. 4 f.). Des Weite-

[21] Mod^{2}DSS als Akronym für *Modeling to Decision Support Systems* bezeichnet den entwickelten Prototyp.

[22] Es handelt sich um eine Bachelorarbeit mit dem Titel *Konzeption und prototypische Implementierung eines Modellierungseditors für modellgetriebene DSS auf Basis von Java*, die von Herrn Nick Oldenburger angefertigt und am 17. Juni 2010 am Fachgebiet Management Support und Wirtschaftsinformatik des Fachbereichs Wirtschaftswissenschaften an der Universität Osnabrück vorgelegt wurde.

[23] Ausführungen zu XML sind bereits in Kapitel 5.1 erfolgt.

ren können sie in leicht eingeschränkter Form als Applets, die ein ähnliches Funktionsspektrum wie Java-Applikationen bieten, in Webseiten integriert werden und sind dadurch weltweit verfügbar (vgl. Krüger, 2009, S. 45). Objektorientierung ist in Verbindung mit dem Paradigma der Modellierung vor allem in Informationssystemen ein oft verwandtes Konzept, da es eine natürliche und realitätsnahe Abbildung beobachteter Zusammenhänge unterstützt (vgl. Riel, 1996, S. 2). Die Elemente der Sprache, die unter Berücksichtigung eines gewissen Abstraktionsgrades ein Abbild der Realität darstellen, können genutzt werden, um eine starke Orientierung an der behandelten Domäne zu realisieren (vgl. Kreutzer, 1990, S. 213; Repenning und Sumner, 1995, S. 17).

VBA ist eine Programmiersprache, die innerhalb von Applikationen genutzt werden kann und aus dem Hause *Microsoft* stammt. Sie ähnelt der „echten" Programmiersprache *Visual Basic* (VB) sehr stark, bietet jedoch spezielle Möglichkeiten in Bezug auf die Umgebungsapplikation (vgl. Buhl, 2005, S. 1 f.). Unter anderem steht dem Anwender VBA in Microsoft-Office Applikationen (z.B. Excel und Access) zur Verfügung. VBA wurde vorrangig gewählt, da eine Visualisierung von modellierten Inhalten in Zusammenhang mit *Microsoft Excel* sehr unkompliziert möglich ist. Es stehen dem Benutzer spezielle Befehle zur Verfügung, die VBA in Excel als Applikationsumgebung integrieren (vgl. Buhl, 2005, S. 2). In Bezug auf den zweistufigen Ansatz ist dies hilfreich, da hierdurch der Benutzer bei der Transformation des in der ersten Modellierungsphase definierten Modells visuell unterstützt werden kann.

6.2 Modellierungseditor

Der Modellierungseditor setzt die erste Phase des konzeptuellen Modellierungsprozesses um. Er bietet die Möglichkeit, die allgemeine Modellstruktur sowie die Modelllogik zu definieren und wurde mittels *Java* programmiert. Im Folgenden werden diejenigen Funktionen und Klassen eingehender erläutert, die für die Implementierung des Editors eine zentrale Rolle einnehmen.

6.2.1 Strukturierung des Programmcodes

Der Programmcode des Modellierungseditors gliedert sich in drei Themenbereiche auf, die jeweils bestimmten Aufgabengebieten zugeordnet sind. In Java wurden hierfür die packages `editor`, `model` und `xml` angelegt, welche die insgesamt 21 Klassen enthalten. Der Editor umfasst insgesamt 11295 Programmierzeilen (inklusive Kommentaren und Strukturierung).

6.2.1.1 Editor-package

Das package `editor` stellt neun Klassen bereit, die für den Aufbau graphischer Oberflächen und sämtliche Steuerungen der Benutzerinteraktionen verantwortlich sind. Jede Funktion des Modellierungseditors ist, soweit sinnvoll und möglich, anhand einer eigenen Klasse umgesetzt. Die grundlegende `main`-Methode, mittels derer jedes Java-Programm gestartet wird, ist Teil dieses package. Die Klasse `EditorFrame` ist der zentrale Bestandteil der grafischen Repräsentation von Mod^{2}DSS. Sie erstellt das Hauptfenster und initialisiert das Reiter-Menü an dessen oberen Rand. Sämtliche Aufrufe von Methoden, über welche die weiteren Funktionalitäten (Klassen) des Editors und deren grafische Umsetzungen aufgerufen werden können, befinden sich ebenfalls in dieser Klasse.

Modell und Repository des Konzepts werden im Prototyp lediglich insofern getrennt betrachtet, als dass ein Repository nicht für die anschließende Transformationsphase vorgesehen ist. Werden Elemente in einem Repository modelliert, stellt das Hauptmenü daher nicht die Möglichkeit bereit, eine Transformationsphase zu starten. Damit das Nutzen und Integrieren von Repository-Elementen innerhalb einer Modelldefinition ohne großen Aufwand gewährleistet werden kann, werden nahezu identische Klassen und Methoden für das Anlegen oder Bearbeiten von Elementen in einem Repository und in einem Modell genutzt. Diese unterscheiden sich lediglich in wenigen Einzelheiten, z.B. einem Element ID, voneinander, sind aber aufgrund dieser Unterschiede getrennt voneinander umgesetzt.

6.2.1.2 Model-package

`Model` fasst als package sieben Klassen zusammen, die Struktur und sämtliche Bestandteile von Modellen auf Java-Ebene erzeugen bzw. verwalten. Jede Modellkomponente auf erster Ebene der XML-Modellstruktur (siehe Kapitel 5.2: Model, Dimension, Dimensionsausprägung, Gruppe und Formel) ist als eigenständige Klasse umgesetzt.

Die Klasse `DSSModel` führt sämtliche Komponenten eines Modells zusammen und repräsentiert die definierten Inhalte als Modellinstanz. Für eine redundanzfreie Objektstruktur im Sinne einer sauberen Programmierung wurden die Datenstrukturen *LinkedList*[24] und *Hashmap*[25] genutzt. Eine *Hashmap* setzt einen sogenannten *assoziativen Speicher* um,

---

[24] „LinkedList ist eine doppelt verkettete Liste, also eine Liste von Einträgen mit einer Referenz auf den jeweiligen Nachfolger und Vorgänger. Das ist nützlich beim Einfügen und Löschen von Elementen an beliebigen Stellen innerhalb der Liste.“ (Ullenboom, 2009, S. 644).

[25] Eine Hashmap implementiert in Java ein internes Array Objekt, in welchem Schlüssel-Wert-Paare abgebildet werden. Aus einem Schlüssel wird mit einer Hash-Funktion ein Hashcode berechnet, der wiederum als

welcher einen Schlüssel mit einem Wert verbindet (vgl. Ullenboom, 2009, S. 644). Über das Konstrukt *LinkedList* werden die einzelnen Dimensions- und Gruppen-Objekte Teil einer Instanz der Klasse `DSSModel`. Instanzen der Klasse `Dimension` verfügen wiederum über zwei *LinkedList*-Objekte, welche Dimensionsausprägungen und Formeln mit der Instanz der Klasse `Dimension` verbinden. Während Dimensionsausprägungen keine Instanzen anderer Objekte in sich vereinigen, beinhaltet die Klasse `DimensionFormula` eine Vielzahl von Objekten, die einen assoziativen Speicher implementieren. Eine Formel besteht aus Verweisen auf Dimensionsausprägungen und logischen Operatoren. Die exakte Position der einzelnen Elemente innerhalb einer Formel ist ausschlaggebend für deren Semantik (siehe Kapitel 3.2.2). Bei einer normalen Subtraktion spielt es beispielsweise eine immanent wichtige Rolle, welche Zahl den Minuenden und welche den Subtrahenden darstellt. Diese und weitere Informationen werden anhand von *Hashmap*- und *LinkedList-Objekten* abgebildet, da diese eine sehr effiziente und performante Möglichkeit der Speicherung bieten. Auf Ebene des Programmiercodes entsteht letztendlich eine hierarchische, verkettete Struktur von Modellelementen. Die Klasse `Repository` ähnelt der Klasse `DSSModel` und enthält ergänzend einige `get`- bzw `set`-Methoden sowie das Element `ID`. Dieses ist nicht Teil eines `Repository`, da eine eindeutige Identifikation unterschiedlicher Repository-Objekte im Konzept nicht genutzt wird.

Einen gesonderten Fall im `Model`-package stellt die Klasse `FormulaCase` dar. Hier werden die im Konzept vorgestellten Fälle der Klassifizierung multidimensionaler Modellierung berechnet. Für jede Formel eines Modells wird ein Fall auf Basis der Klassifizierungskriterien *Grad*, *Homogenität* und *Simultanität* unter Berücksichtigung der Modellstruktur und -komponenten berechnet. Werden Änderungen im Modell vorgenommen, welche Auswirkungen auf die Fälle haben könnten, wird für jede Formel der aktuelle Fall durch Aufruf bzw. Instanziierung der Klasse `FormulaCase` neu berechnet. Wird in einem Modell z.B. die Definition einer Formel geändert oder eine neue Formel spezifiziert, werden die Fälle aller Modellformeln neu berechnet. Wird lediglich die Bezeichnung eines Modellelements, z.B. einer Dimension oder einer Dimensionsausprägung geändert, erfolgt keine erneute Berechnung der Fälle.

Index für das interne Array dient. Für die Nachfrage nach einem Schlüssel muss die Hash-Funktion umgekehrt gerechnet werden, um zum entsprechenden Schlüssel zu gelangen (Ullenboom, 2009, S. 694).

6.2.1.3 XML-package

Das package `xml` besteht aus vier Klassen, die für das Erzeugen und Einlesen von Modellen als XML-Dokument benötigt werden. Auch in diesem Fall müssen Modell und Repository trotz ihrer Ähnlichkeit wiederum getrennt voneinander behandelt werden, da die Struktur innerhalb der XML-Dokumente nicht vollständig identisch ist. In beiden Fällen werden die einzelnen Modellelemente mittels des *Java Document Object Model* (JDOM) in XML-Knoten bzw. XML-Elemente konvertiert. Dazu wird das Modell in einer Baumstruktur abgebildet (Speichern) bzw. die vorliegende Baumstruktur in die entsprechenden Modellelemente zerlegt (Einlesen).

6.2.2 Hauptfenster

Mod^{2}DSS bietet zwei unterschiedliche Hauptfenster an. Eins wird für die Definition von Elementen innerhalb eines Repository (siehe Abbildung 6-1) und ein weiteres für die Definition (und Komposition) eines Modells (siehe Abbildung 6-2) genutzt.

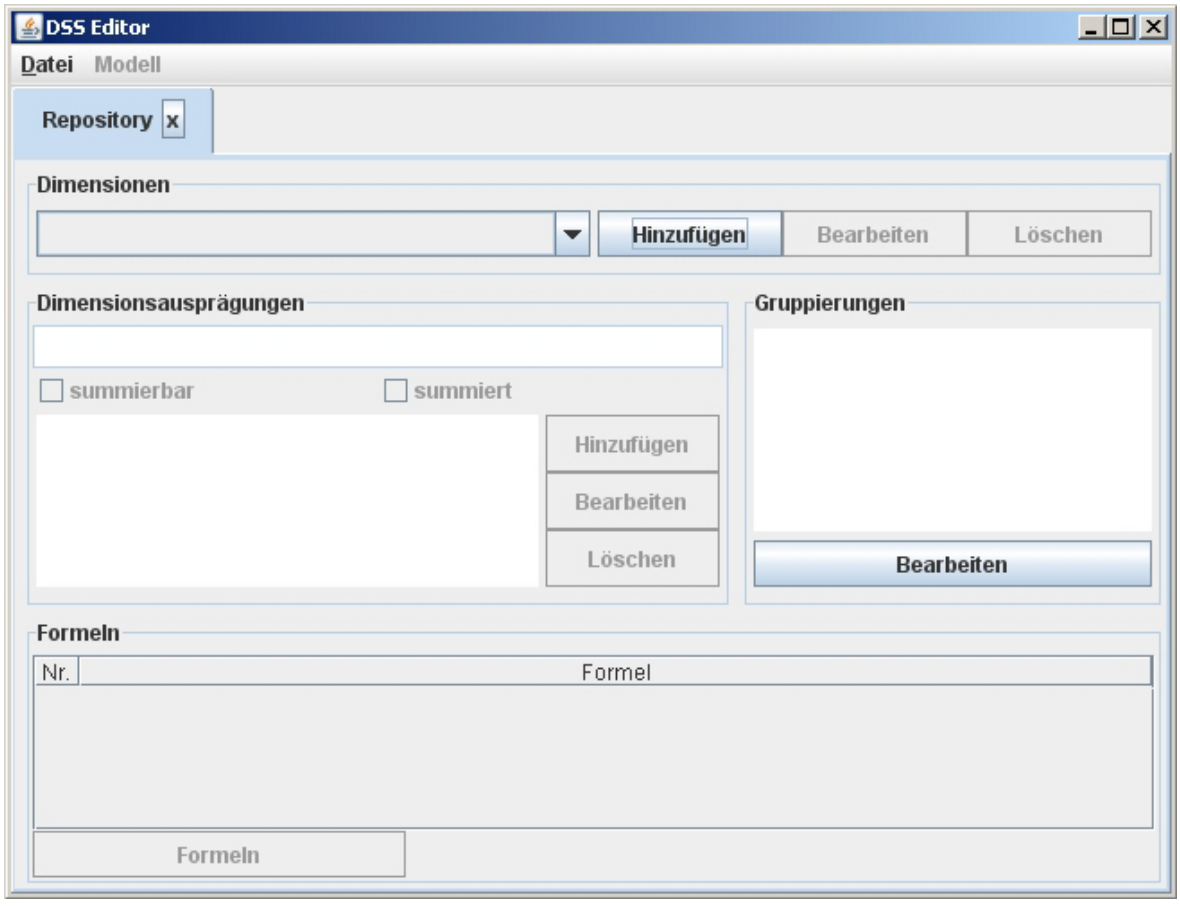


Abbildung 6-1: Repository-bezogenes Hauptfenster

Abbildung 6-1 stellt das Hauptfenster eines Repository dar. Da die einzelnen Dimensionen, Dimensionsausprägungen, Formeln und Gruppen nicht wie in einem Modell als individuel-

le Komposition von Elementen betrachtet werden, sondern jedes Element für sich allein existieren und für die Komposition von Modellen genutzt werden kann, sind Manipulationsmöglichkeiten für das Anlegen von Dimensionen und Dimensionsausprägungen direkt in das Hauptfenster integriert.

Abbildung 6-2 zeigt das Modell-bezogene Hauptfenster von Mod^{2}DSS. Nachdem ein oder mehrere neue Modelle angelegt oder bereits bestehende Modelle geöffnet wurden (via Menü *Datei*), werden diese jeweils als Reiter dargestellt. Der jeweilige Modellname dient hier zur Beschriftung der einzelnen Reiter. Die Abbildung zeigt demnach ein Modell mit der Bezeichnung *Evaluation_v4_transformed*.

Im Bereich *Optionen* des Hauptfensters hat der Benutzer die Möglichkeit weitere Fenster durch Anklicken von Schaltflächen aufzurufen. Mittels dieser Fenster können sämtliche Modellstrukturen (Dimensionen & Ausprägungen, Gruppen und Formeln) angelegt bzw. editiert werden.

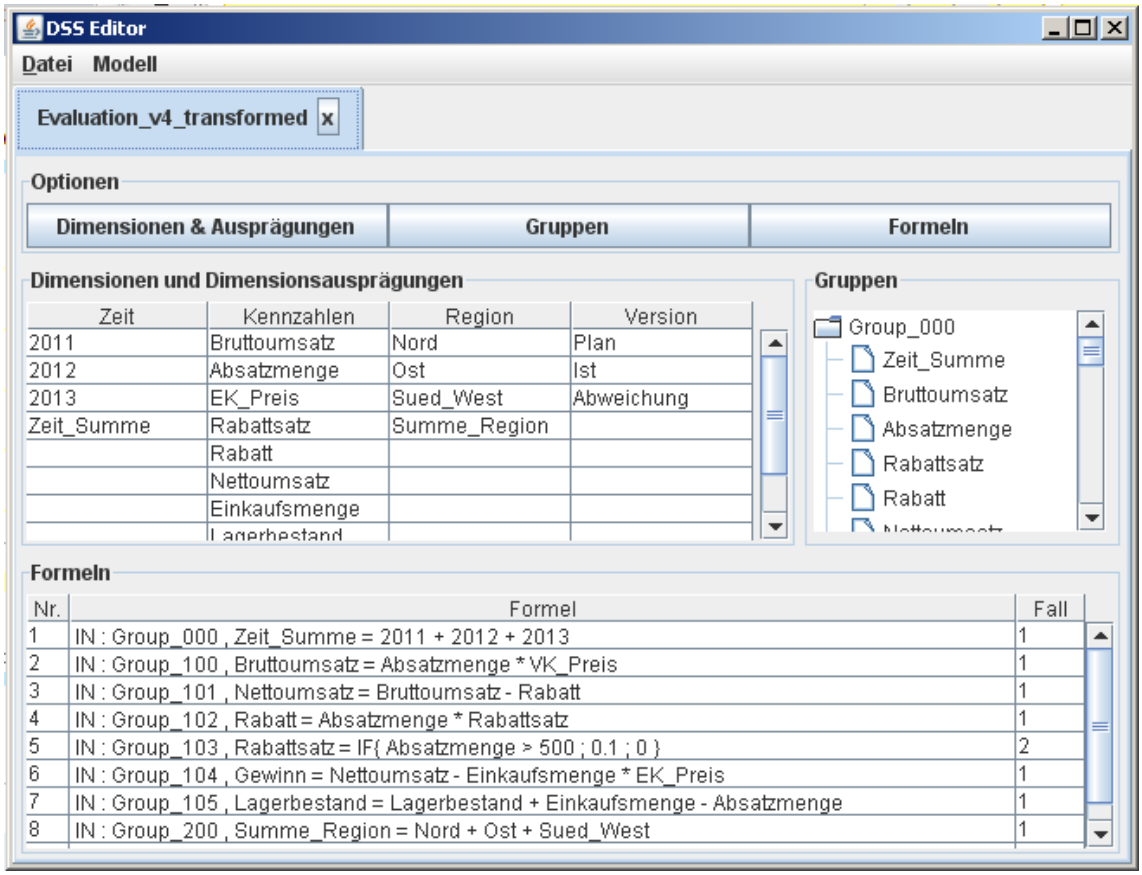


Abbildung 6-2: Modell-bezogenes Hauptfenster

Des Weiteren zeigt der Bereich unter den Optionsmöglichkeiten die Struktur des aktuellen Modells (ausgewählter Reiter bei mehreren gleichzeitig geöffneten Modellen), also sämtliche Dimensionen und die zugehörigen Ausprägungen, die während der Modellierung defi-

niert wurden. Jede Dimension des Modells wird als eine Tabellenspalte dargestellt. Die Methoden `setDimTable` und `createDimTable` in der Klasse `ModelContent` steuern die Ausgabe über ein `JTable`-Objekt mit einem Parameter des Typs `DefaultTableModel`. In den einzelnen Zeilen der Spalten werden die zugehörigen Ausprägungen der Dimensionen dargestellt. Direkt neben den Dimensionen und deren Ausprägungen werden die angelegten Gruppen in einem hierarchischen Baumdiagramm dargestellt. Dieser Baum wird durch die Methode `setGroupTree` erzeugt und mittels der Methode `setGroupTreeContent` gefüllt. Im unteren Teil des Hauptfensters wird dem Benutzer die Logik des Modells durch eine Auflistung aller modellierten Formeln präsentiert. Die Methoden `setSouthContent` bzw. `loadDimFormula` übernehmen diese Aufgaben. Des Weiteren nimmt Mod^{2}DSS basierend auf der Klassifizierung multidimensionaler Modellierung (siehe Abschnitt 4.3) eine Einordnung jeder Formel in den Gesamtkontext des Modells vor. Dies geschieht durch Instanziierung der Klasse `FormulaCase` je Formel eines Modells und wird in der gesonderten Spalte *Fall* visualisiert.

Im Weiteren wird nun näher auf die Fenster eingegangen, welche die Modellierung ermöglichen und über die Schaltflächen im Bereich *Optionen* erreicht werden.

6.2.3 Dimensionen und Dimensionsausprägungen

Abbildung 6-3 zeigt das Fenster *Dimensionseigenschaften*, welches über die Schaltfläche *Dimensionen & Ausprägungen* (siehe Abbildung 6-2) geöffnet wird:

Abbildung 6-3: Fenster Dimensionseigenschaften

Im unteren Teil des Fensters können Ausprägungen der im oberen Teil ausgewählten Dimension bearbeitet, angelegt oder gelöscht werden. Wie bereits im Konzept erläutert, können zu jeder Ausprägung Informationen bezüglich des Verhaltens einer Ausprägung in Bezug auf eine Summenformel hinterlegt werden, die im weiteren Verlauf des Modellierungsprozesses genutzt werden (siehe Kapitel 5.2.3). Dies findet über die Checkboxen *summierbar* und *summiert andere* statt.

In der oberen Hälfte des Fensters finden sich sämtliche Möglichkeiten wieder, die für das Anlegen und Bearbeiten von Dimensionen eines Modells notwendig sind. An dieser Stelle hat der Benutzer auch die Möglichkeit, auf ein bestehendes *Repository* zurückzugreifen, um komplette Dimensionen oder einzelne Ausprägungen und Formeln in das aktuell bearbeitete Modell zu importieren, anstatt sie neu anlegen zu müssen. Abbildung 6-4 zeigt das Fenster zum Einfügen von Elementen aus einem Repository:

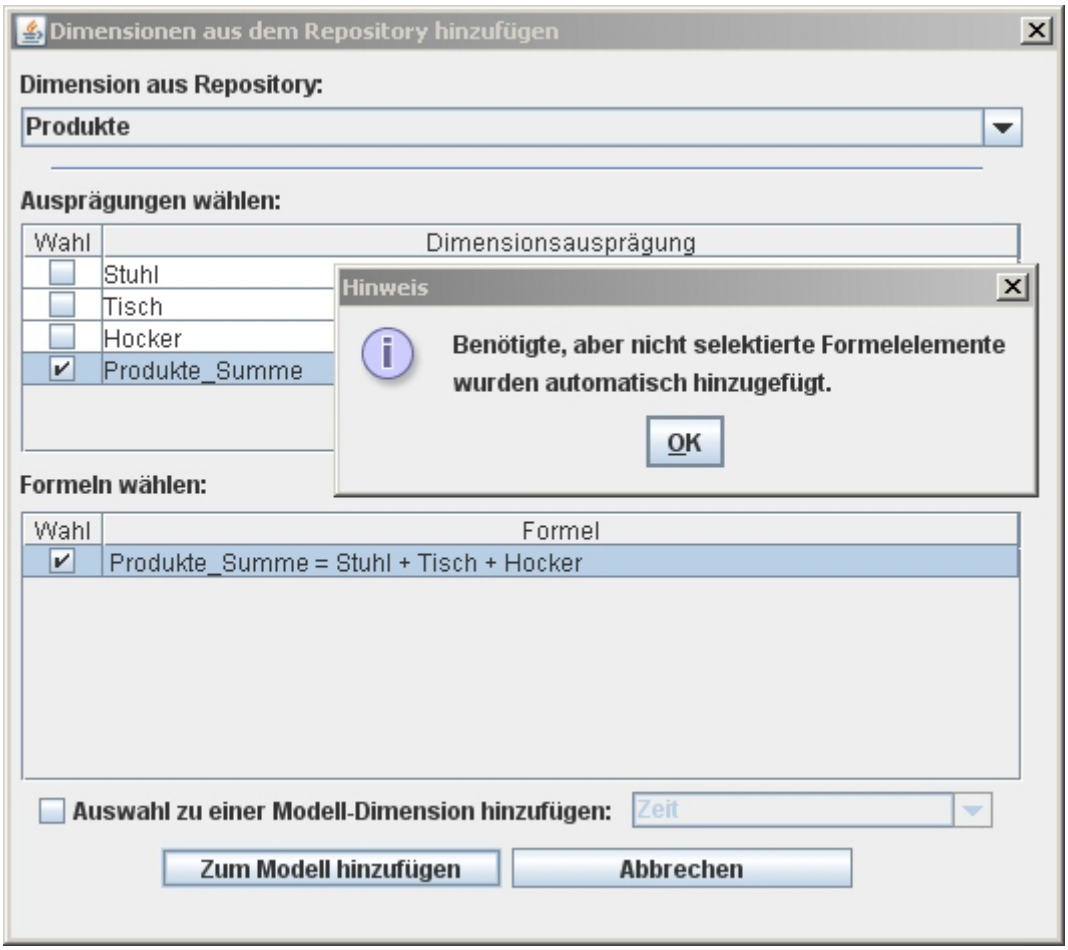


Abbildung 6-4: Dimensionen aus Repository einfügen

Das Hinweisfenster wird angezeigt, da die Schaltfläche *Zum Modell hinzufügen* betätigt wurde, die ausgewählten Elemente jedoch zu Inkonsistenzen im Modell führen würden. Hierfür werden die Inhalte des Modells und der einzufügenden Elemente abgeglichen. In diesem konkreten Fall wurden nicht alle Dimensionsausprägungen der Dimension *Produkte* ausgewählt, die als Variable in der zu importierenden Formel referenziert werden. Die drei nicht ausgewählten Dimensionsausprägungen *Stuhl*, *Tisch* und *Hocker* werden daher ebenfalls in das Modell importiert. Sofern nur ausgewählte Ausprägungen der Dimension Produkte importiert werden sollen, besteht im unteren Teil des Fensters die Möglichkeit das Anlegen einer neuen Dimension in das Modell zu verhindern und stattdessen eine vorhandene Modelldimension anzugeben, welcher die Elemente hinzugefügt werden.

6.2.4 Gruppen

Das dimensionsübergreifende Zusammenfassen von Ausprägungen zu Gruppen verschafft dem Benutzer die Möglichkeit, innerhalb der Definition von Formeln direkt mittels des Gruppennamens für Teile des Modells logische Zusammenhänge zu definieren. Abbildung

6-5 zeigt das Fenster, welches für diese Operation verwendet wird und über die Schaltfläche *Gruppen* (siehe Abbildung 6-2) erreicht wird.

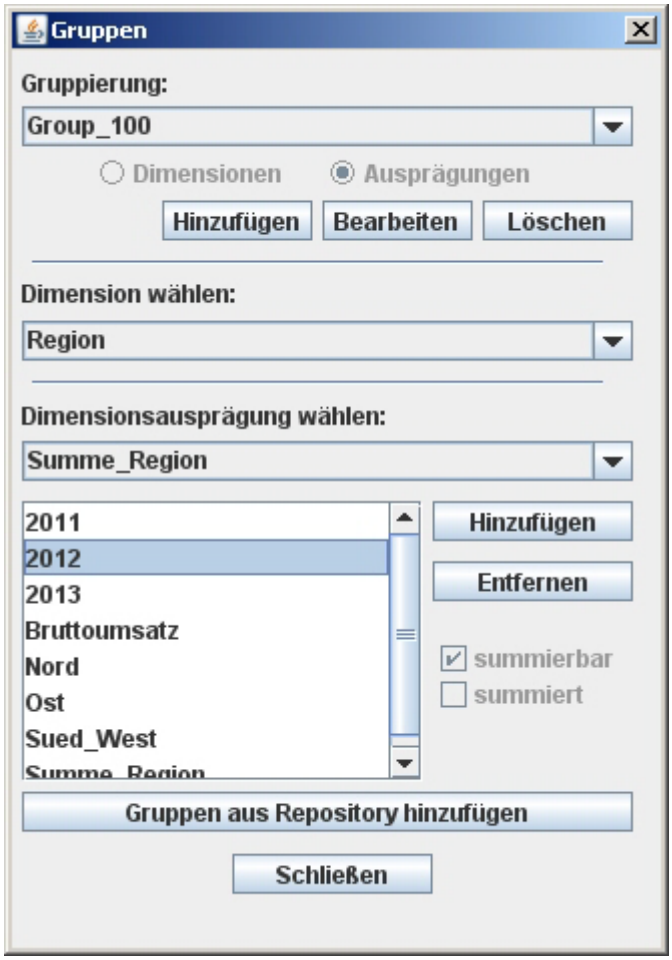


Abbildung 6-5: Fenster Gruppen

Es besteht die Möglichkeit, Dimensionsausprägungen oder ganze Dimensionen zu Gruppen zusammenzufassen, wobei die Möglichkeit der Gruppierung von Ausprägungen die deutlich häufiger verwandte ist. Diese Funktion wird im weiteren Verlauf des Modellierungsprozesses auch zum Abschluss der Modelltransformation genutzt, um die Anwendung der Formeln auf eine gewisse Zellmenge einzuschränken (siehe Kapitel 6.4). Eine simultane Gruppierung von Ausprägungen und Dimensionen innerhalb einer Gruppe ist nicht möglich (siehe Kapitel 5.2.4). Beim Anlegen einer neuen Gruppe muss der Benutzer anhand eines Radio-Button festlegen, ob die Gruppe Dimensionsausprägungen oder ganze Dimensionen enthalten soll (siehe Abbildung 6-5). Des Weiteren besteht auch hier wiederum die Möglichkeit, Gruppen aus einem Repository zu importieren, anstatt sie neu definieren zu müssen (siehe Abbildung 6-5: Schaltfläche *Gruppen aus Repository hinzufügen*).

Beim Löschen einer Dimensionsausprägung (siehe Abbildung 6-3) wurde eine Konsistenzfunktion in diesen Prozess integriert. Es wird geprüft, ob eine zu löschende Dimensions-

ausprägung Teil einer Gruppe ist. Ist dies der Fall, wird der Anwender darauf hingewiesen, dass diese auch aus Gruppen gelöscht wird, in denen sie referenziert wird.

6.2.5 Formeln

Das dritte und letzte Fenster zur Manipulation von Modellstrukturen wird über die Schaltfläche *Formeln* geöffnet (siehe Abbildung 6-2) und bietet dem Benutzer die Möglichkeiten, Formeln zu einem Modell hinzuzufügen, Formeln zu löschen oder sie zu ändern.

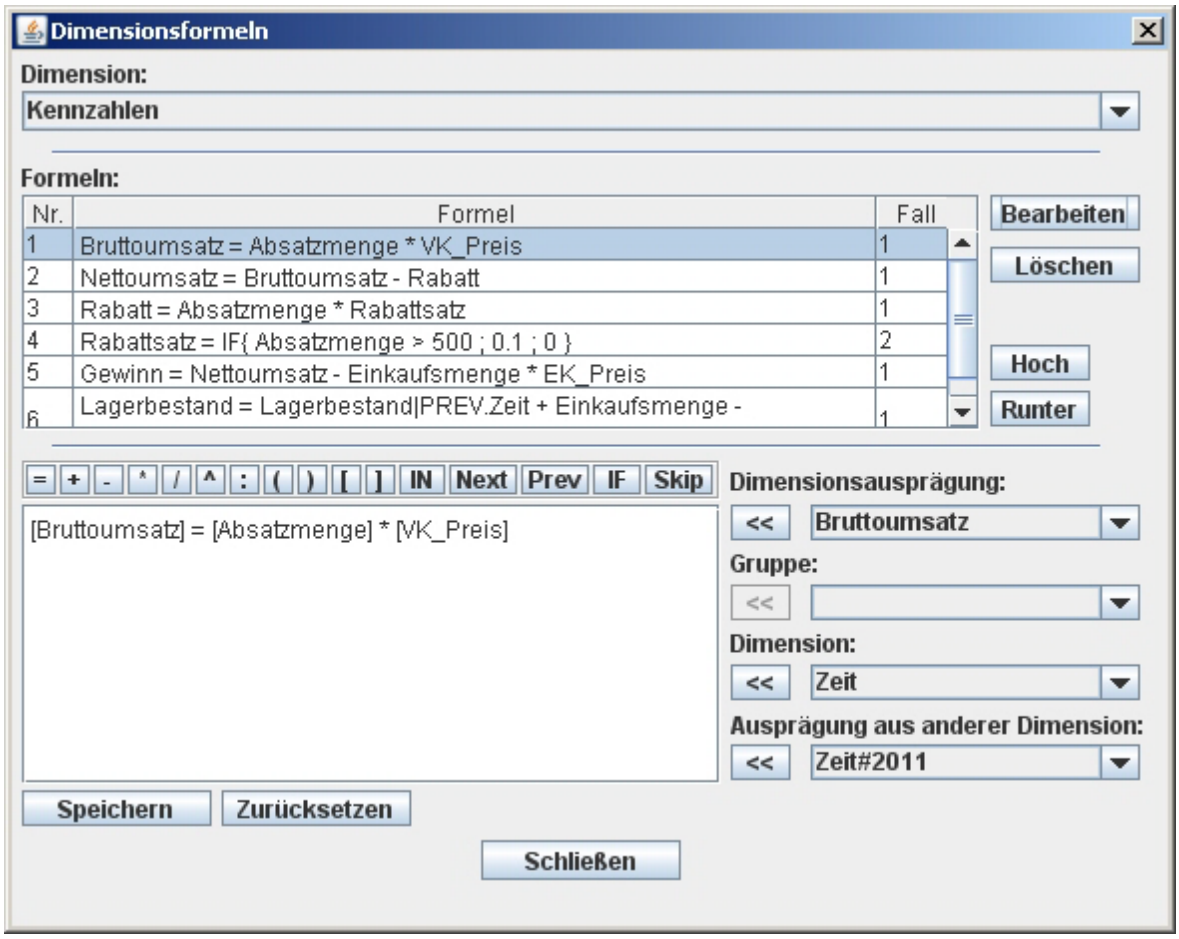


Abbildung 6-6: Fenster Formeln

Abbildung 6-6 zeigt das Fenster, welches eine Zweiteilung aufweist. In der oberen Hälfte werden dem Benutzer alle Formeln der im Drop-Down Menü gewählten Dimension (hier: *Kennzahlen*) angezeigt. Eine Formel ist immer Bestandteil einer Dimension. Eine freie Formel, die keiner Dimension zugeordnet ist, kann mittels des Modellierungseditors nicht herbeigeführt werden, da die Formel durch ihre Dimensionszugehörigkeit implizit Spezifikationsinformationen erhält. Als Konsistenz sichernde Maßnahme werden daher beim Löschen von Dimensionen auch die zugehörigen Formeln gelöscht. Im unteren Teil des Fensters befinden sich sämtliche Schaltflächen, die für das Anlegen oder Verändern von Formeln notwendig sind. Es ist sowohl möglich, die Formeln durch Bedienen der Schalt-

flächen und Drop-Down Menüs zusammenzustellen, als auch sie manuell in das Textfeld einzutragen. Im weiteren Verlauf des Modellierungsprozesses muss der Modellierungseditor anhand von Steuerelementen[26] die Modellelemente, z.B. Dimensionen, Ausprägungen oder auch logische Operatoren wie Plus- oder Minuszeichen, automatisch unterscheiden können. Daher werden Dimensionen und Dimensionsausprägungen immer in eckige Klammern geschrieben. Ausprägungen einer anderen Dimension, als der aktuell gewählten (bezogen auf die Dimension, der die Formel angehört; siehe Drop-Down Menü in Abbildung 6-6), werden extra gekennzeichnet: Zusätzlich zu den eckigen Klammern wird die Dimension, aus welcher die Ausprägung stammt, in das Element aufgenommen. Eine Raute trennt zudem Dimension und Dimensionsausprägung, z.B. [Zeit#2011] für eine Ausprägung *2011* aus der Dimension *Zeit*. Werden zwei Dimensionsausprägungen miteinander kombiniert, um einen Zellbereich im Sinne einer Teilmengen-differenzierten Kennzahlenspezifikation zu definieren, müssen diese durch einen Doppelpunkt getrennt werden. Eine IF-THEN-ELSE Klausel wird durch ein *IF* eingeleitet und die Bedingung, der *Then*-Wert und der *Else*-Wert folgen innerhalb einer geschweiften Klammer, jeweils getrennt durch ein Semikolon.

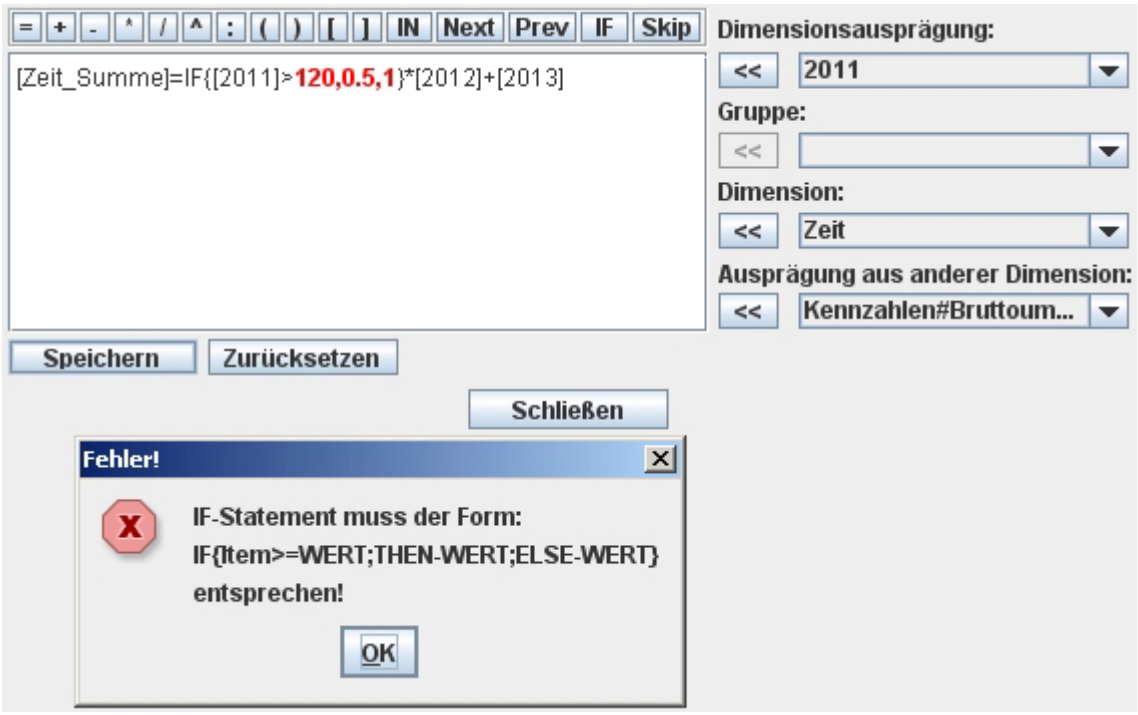


Abbildung 6-7: Unterstützung beim Modellieren von Formeln

[26] Ein Steuerelement ist in diesem Zusammenhang für die Abgrenzung semantisch unterschiedlicher Formelelemente da, die ohne dieses Element nicht automatisiert zu differenzieren wären.

Der Anwender wird im Modellierungsprozess insofern unterstützt, als dass ihm die korrekte Form eines IF-THEN-ELSE Statements angezeigt wird. Des Weiteren wird in jedem Fall der Teil einer Formel rot und fett markiert, der nicht den Konventionen genügt.

Das Löschen einer Dimensionsausprägung ist nicht nur im Hinblick auf Gruppen (siehe Abschnitt 6.2.4) interessant, sondern auch für eine Formel. Wird eine Ausprägung gelöscht, die (expliziter) Teil einer Formel ist, wird diese Formel deaktiviert.

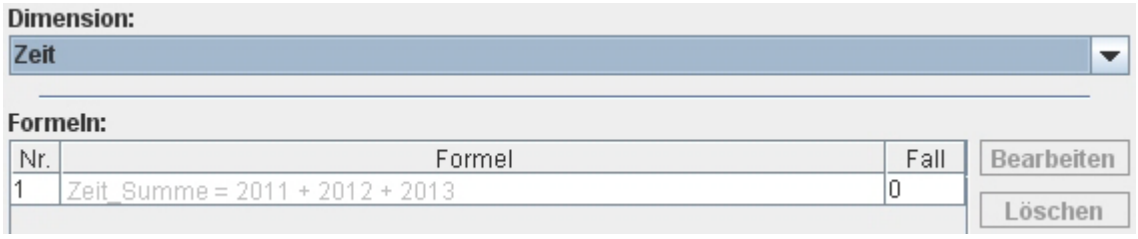


Abbildung 6-8: Deaktivierte Formel

Abbildung 6-8 zeigt ein Beispiel für eine deaktivierte Formel. In diesem Fall war das Löschen der Dimensionsausprägung *2011* der Grund der Deaktivierung.

Auch Formeln lassen sich aus einem Repository importieren. Da eine Formel immer Teil einer Modelldimension ist, wurde diese Option im Rahmen des Imports von Dimensionen umgesetzt (siehe Abbildung 6-4).

Das Ergebnis der ersten Modellierungsphase liegt in der Java Laufzeitumgebung in Form der Instanz `DSSModel` vor. Abschnitt 6.3 erläutert die XML-Schnittstelle, mittels derer ein Modell in ein XML-Dokument überführt wird.

6.3 XML-Schnittstelle

Die im `XML`-package bereitgestellten Klassen übernehmen die Aufgabe, die modellierten Strukturen als XML-Dokument zu speichern. Angestoßen wird dieser Prozess über den Menüeintrag *Modell speichern*, der im Hauptfenster (vgl. Abbildung 6-2) über den Eintrag *Datei* in der Menüleiste erreicht werden kann. Abbildung 6-9 visualisiert dieses Menü:

Abbildung 6-9: Modell speichern

Mittels des JDOM wird anhand der Java-Klassen `XMLExportModel` und `XMLExportRepository` unterschieden, ob ein Modell oder ein Repository gespeichert wird. Die XML-Repräsentationen dieser Konstrukte unterscheiden sich nur in den folgenden Punkten: Durch die Methode `save` wird das benötigte Wurzelelement <*dsseditor*> angelegt, gefolgt von einem Element <*model*> oder <*repository*>. Des Weiteren wird in einer Modellinstanz der Name des Repository im Tag <*from-repository*> gespeichert, sofern bei der Modellierung Inhalte eines Repositorys genutzt wurden. Es können je Modell lediglich Modellelemente eines Repository genutzt werden. Entsprechende Inhalte werden bei der Modellkomposition als Kopie in das Modell aufgenommen.

Im weiteren Verlauf werden nun alle Modellinhalte, die in Form von Java-Objekten vorliegen und irgendeine Auskunft über Logik oder Struktur des Modells geben, in XML-Elemente transformiert. Hierzu wird die hierarchische Struktur der Modellelemente in der Java-Laufzeitumgebung mittels verschachtelter Schleifen durchlaufen. Da auf Java-Ebene ein Dimensionsobjekt mittels *LinkedLists* und *Hashmaps* auf Dimensionsausprägungen und Formeln verweist, durchläuft die äußerste Schleife sämtliche Dimensionen und legt für jede ein XML-Element <*dimension*> mit den vorgefundenen Attributen an. Innerhalb dieser durchläuft eine weitere Schleife die vorhandenen Ausprägungen sowie anschließend eine Schleife die vorhandenen Formeln des Modells.

Insbesondere dieses Anlegen der Formeln ist jedoch ein aufwändiger Schritt und bedarf genauerer Erläuterung. Die flache Struktur eines XML-Dokuments muss die Objektstruktur in der Java-Umgebung abbilden. *Hashmaps* oder *LinkedLists* stehen als Konstrukte in XML nicht zur Verfügung und müssen durch adäquate Strukturen ersetzt werden. Innerhalb einer Formel ergibt sich beispielsweise das Problem, dass sich Dimensionsausprägungen im Bezugsteil und Logikteil einer Formel syntaktisch nicht unterscheiden und die Semantik der einzelnen Formelelemente maßgeblich durch deren Reihenfolge beeinflusst

wird. Um im Programmablauf, insbesondere beim Speichern und Einlesen von Modellen in XML-Format, einen einfacheren Zugriff auf einzelne Bestandteile einer Formel zu ermöglichen, wird diese Semantik durch die XML-Elemente <*scopeItem*> und <*mainItem*> mit in das Dokument aufgenommen. Ähnlich wird mit Ausprägungen verfahren, die den Skip-Bereich einer Formel repräsentieren. Hierfür wird das Steuerelement *Skip* innerhalb eines Tags <*formula-element*> eingebaut.

Die definierte Logik einer Formel wird durch diesen Prozess Schritt für Schritt abgebildet. Die Elter-Elemente der anzulegenden XML-Elemente ergeben sich aus der *Hashmap* `RangeIndex`, die Bestandteil des Java-Objekts `Formula` ist. Sie beschreibt jedes Element einer Formel, insbesondere ob es sich aus mehreren Dimensionsausprägungen zusammensetzt und welche dies sind. Dies ist für die Transformation zu beachten, da es im Sinne der Umsetzung der Teilmengen-differenzierten Kennzahlenspezifikation Elemente gibt, die Elter-Element mehrerer, zusammengehöriger Kind-Elemente sind, z.B. ob ein scopeItem Element aus mehreren Dimensionsausprägungen besteht. Die rekursive Methode `checkRange` wird hierfür am Ende eines jeden Schleifenzyklus, d.h. nach jeder Dimension, jeder Ausprägung und jeder Gruppe, die bei der Transformation einer Formel in das XML-Dokument eingefügt wurde, aufgerufen. Innerhalb der Methode wird geprüft, ob das aktuell zu transformierende Element zu demselben Elter-Element gehört, wie das vorhergehende Element. Ist dies der Fall, wird das Element, getrennt durch ein formula-element Element „:“, in das XML-Dokument eingefügt. Nun ruft sich die Methode mit den Parametern des nachfolgenden Elements selbst auf, um den Rekursionszyklus zu implementieren. Dies geschieht solange, bis das Elter-Element zweier aufeinanderfolgender Elemente voneinander abweicht und der nächste Zyklus der geschachtelten Schleifen (in Bezug auf Dimensionen und Dimensionsausprägungen) beginnt. Besteht ein mainItem Element einer Formel z.B. aus den Dimensionsausprägungen *Menge* und *2011*, wird das öffnende Tag eines mainItem Elements in das XML-Dokument eingefügt und die Referenz auf die Dimensionsausprägung *Menge* als Kind-Element des mainItem Elements in das XML-Dokument eingefügt. Anschließend wird mittels der Methode `checkRange` geprüft, ob die Ausprägungen *Menge* und *2011* dasselbe Elter-Element besitzen. Da dies in diesem Beispiel der Fall ist, weil beide Ausprägungen Teil desselben mainItem Elements sind, wird die Referenz auf die Dimensionsausprägung *2011* ebenfalls als Kind-Element des mainItem Elements in das XML-Dokument eingefügt. Sobald ein Formelelement ein anderes Elter-Element als das jeweils vorangegangene Element hat, wird das schließende

XML-Tag (im Beispiel würde dies dem Tag *</mainItem>* entsprechen) in das XML-Dokument eingefügt.

Sind alle Dimensionen, Dimensionsausprägungen und Formeln transformiert, werden die definierten Gruppen im XML-Dokument angelegt. Im nächsten Schritt folgen nun die Erläuterungen zur Umsetzung der zweiten Phase des konzeptuellen Modellierungsprozesses, der Transformationsphase, welche mittels Excel in Verbindung mit VBA umgesetzt ist.

6.4 Modelltransformation und Kompilierung

Nachdem die Modellstruktur mit dem Fokus auf Arithmetik und Modelllogik definiert wurde, ohne hierbei eventuell resultierende Schwierigkeiten zu berücksichtigen, welche sich aus der Konstellation von Modellstruktur und -logik ergeben (siehe Kapitel 3.2), stellt die Modelltransformation die Auflösung dieser Konflikte in den Vordergrund.

6.4.1 Erstellung eines Spreadsheet-Modells

Abbildung 6-10 zeigt das Hauptfenster des Modellierungseditors in einem Status, in welchem ein Modell gespeichert und damit in das XML-Format überführt worden ist:

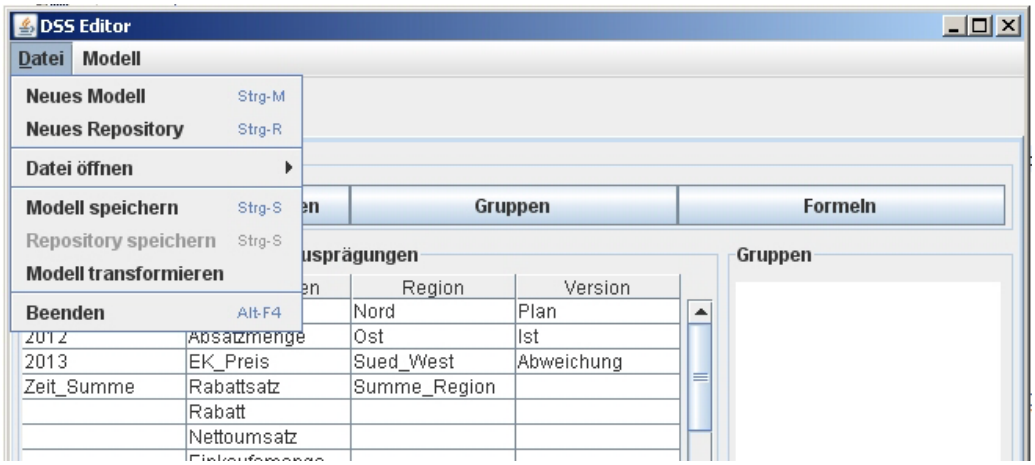


Abbildung 6-10: Start der Modelltransformation

Der zuvor ausgeblendete Menüeintrag *Modell transformieren* (siehe Abbildung 6-9) ist nun auswählbar und startet die zweite Phase des Modellierungsprozesses.

Durch Auswahl des Menüpunkts wird gemäß der gewählten Methodik (siehe Kapitel 6.1) die Datei *Transformator.xls*[27] geöffnet und ein darin hinterlegtes Makro ausgeführt. Die visuelle Unterstützung ist für den Anwender interessant, da er im Transformationsprozess

[27] Die Datei Transformator.xls muss sich im selben Verzeichnis wie die Java-Anwendung befinden. Ist dies nicht der Fall, muss der Benutzer den Transformator manuell öffnen.

in Form eines Dialogs nach der anzuwendenden Formelalternative der Zellen gefragt wird, in denen sich Formeln überschneiden. Durch die Visualisierung kann der Benutzer leichter einschätzen, an welcher Stelle im Modell er sich befindet. Es öffnet sich nach Auswahl des Menüpunkts das Startfenster der Transformation (siehe Abbildung 6-11):

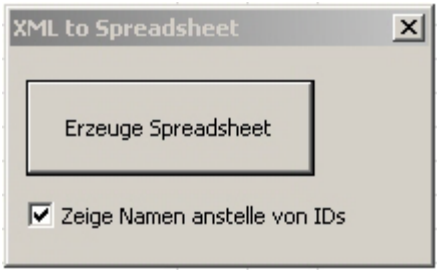


Abbildung 6-11: Startfenster Transformation

Die gesamte Transformationsphase gliedert sich über die XML-Schnittstelle an die Konfigurationsphase des Modellierungsprozesses an. Beim Start der Transformation werden die Dimensionen und Ausprägungen des zu transformierenden Modells innerhalb eines Excel-Spreadsheets[28] erzeugt.

Grundsätzlich ist festzustellen, dass das Makro anstatt der Bezeichnungen die *IDs* der Modellelemente (Dimensionen, Ausprägungen, Formeln und Gruppen) nutzt, da auf diese Weise eine Identifizierung der Objekte leichter möglich ist. Es wäre deutlich aufwändiger, einzelne Strings als numerische IDs miteinander zu vergleichen, sodass das Nutzen der IDs zu einer Verbesserung der Performance führt. Nachdem die Modellelemente im Spreadsheet erzeugt wurden, werden die IDs der Modellelemente durch deren Bezeichnungen bzw. Namen ersetzt, sofern die standardmäßig aktivierte Option nicht deaktiviert wurde (siehe Abbildung 6-11, *Zeige Namen anstelle von IDs*).

Aus Gründen der Übersichtlichkeit wird im Transformationsschritt das Modell auf einem einzigen Tabellenblatt (im Sinne von Excel) aufgebaut. Das Navigieren zwischen Tabellenblättern entfällt dadurch. Neben diesem Unterschied bezüglich der Dimensionalität sind Excel-Modelle Zell-basiert aufgebaut, d.h. für jede Modellzelle muss explizit eine Formel hinterlegt werden, sofern keine (Eingabe-)Daten repräsentiert werden sollen. Diesen konzeptuellen Unterschieden zu Mod^{2}DSS Modellen (Multidimensionalität und Vektor-basierte Formeln) muss insbesondere beim Aufbau des Excel-Modells Rechnung getragen werden. Microsoft bietet mit *PowerPivot* zwar eine Excel-Erweiterung an, die es erlaubt

[28] Wenn in diesem Zusammenhang von einem Spreadsheet die Rede ist, beschränkt sich der Begriff auf ein Modell, das mittels Microsoft Excel als Tabellenkalkulation definiert wurde. Dabei liegt der ausdrückliche Fokus auf der Anzahl zur Verfügung stehender Dimensionen. Diese ist in Excel auf drei sichtbare (Ordinate und Abszisse sowie die Möglichkeit verschiedene Arbeitsblätter zu verknüpfen) beschränkt.

multidimensionale Modelle zu verarbeiten, diese haben jedoch mit der Analyse großer Datenmengen (vgl. Horak et al., 2010, S. 5) einen anderen Fokus als MDSS. Sie bewegen sich im Kontext von OLAP-Systemen und sollen deshalb hier nicht näher betrachtet werden.

6.4.1.1 Projektion von Dimensionen

Die Abbildung multidimensional modellierter Modelle in flachen, Zell-basierten Spreadsheets wird durch die Projektion einzelner Dimensionen in die zur Verfügung stehenden zwei Dimensionen bewerkstelligt. Sind beispielsweise in einem Mod^{2}DSS Modell vier Dimensionen *Zeit*, *Regionen*, *Produkte* und *Kennzahlen* definiert, werden die ersten zwei jeweils durch Abszisse und Ordinate repräsentiert. Die übrig bleibenden zwei Dimensionen müssen nun in eine dieser beiden projiziert werden, da ein Excel-Spreadsheet keine weiteren Mechanismen oder Konzepte zur Verfügung stellt. Abbildung 6-12 visualisiert diesen Vorgang. Die Ausprägungen der Dimensionen *Kennzahlen* werden mit den Ausprägungen der Dimension *Regionen*, und die Ausprägungen der Dimension *Produkte* mit den Ausprägungen der Dimension *Zeit* kombiniert, so dass für diese Dimensionspaare alle Kombinationen von Ausprägungen abgedeckt sind (kartesisches Produkt). Die Zellmenge des zweidimensionalen Raums wird durch dieses Vorgehen so angepasst, dass alle Zellen des vierdimensionalen Modells im vorhandenen Modellraum existieren.

	A	B	C	D	E	F	G	H
1		Regionen	Kennzahlen					
2	Zeit			2011	2011	2012	2012	
3	Produkte			Produkt A	Produkt B	Produkt A	Produkt B	
4		Nord	Umsatz					
5		Nord	Menge					
6		Nord	Preis					
7		West	Umsatz					
8		West	Menge					
9		West	Preis					
10								

Abbildung 6-12: Projektion von Dimensionen ineinander (Beispiel)

Der markierte Bereich (umrandet) stellt die Modellzellen dar. Die Ausprägungen der Dimensionen *Zeit* und *Regionen* werden so oft vervielfältigt, wie die zu projizierende Dimension (*Produkte* bzw. *Kennzahlen*) an Ausprägungen umfasst. Bei vier Dimensionen mit

drei Mal zwei Ausprägungen je Dimension (*Zeit*, *Regionen* und *Produkte*) und drei Ausprägungen in der Dimension *Kennzahlen* ergeben sich 24 Modellzellen (2*2*2*3)[29].

Zur Realisation dieser Projektion werden auf VBA-Ebene die erste Hälfte aller im XML-Dokument enthaltenen Dimensionen der Ordinate und die restlichen Dimensionen der Abszisse zugeordnet (bei ungerader Zahl wird der Abszisse eine Dimension mehr zugeordnet, als der Ordinate). Das bedeutet, dass für jede Dimension, die der Ordinate zugeordnet ist, eine eigene Spalte reserviert wird, in deren Zeilen die Ausprägungen dieser Dimension geschrieben werden (*Zeilenbezeichner*). Um dies kenntlich zu machen, wird der Bezeichner der Dimension in die Zelle der ersten Zeile der betreffenden Spalte des Tabellenblattes geschrieben (vgl. Abbildung 6-12: Zelle A2 - Zeit). Alle nachfolgenden Dimensionen auf der Ordinate werden schrittweise den folgenden Spalten zugeordnet. Im weiteren Verlauf werden nun mittels einer Schleife sämtliche Ausprägungen der Dimensionen nacheinander in die Zeilen der betreffenden Spalte geschrieben. Für die Dimensionen, die der Abszisse zugeordnet werden, wird das gleiche Verfahren angewandt. Unterschieden werden hier lediglich nicht die Zeilen, sondern die Spalten, die nach und nach mit den Bezeichnern der Dimensionsausprägungen aufgefüllt werden (*Spaltenbezeichner*). Indem für Spalten- und Zeilenbezeichner jeweils eine ausreichend große Anzahl Zeilen bzw. Spalten reserviert werden, können durch dieses Verfahren prinzipiell beliebig viele Dimensionen in das Modell eingefügt werden. Es wird nach und nach ein zweidimensionaler Modellraum aufgebaut, der durch Kombination von Ausprägungen unterschiedlicher Dimensionen in Zeilen und Spalten sämtliche Kombinationen des im XML-Dokument definierten mehrdimensionalen Modells abbildet. Restriktiver Faktor ist in diesem Zusammenhang die unterstützte Anzahl von Zeilen und Spalten eines Spreadsheet.

Die Informationen der Attribute *summierbar* und *summiert* (siehe Kapitel 5.2.3) werden im Prototyp genutzt, um diese besonderen Zellen eines Modells zu markieren. Entsteht eine Zelle durch Kombination einer Dimensionsausprägung mit dem Attribut „summiert=ja“ und einer Dimensionsausprägung mit dem Attribut „summierbar=nein“, visualisiert das Makro die entsprechende Modellzelle, indem es sie grau ausfüllt.

Das Abbilden Vektor-basierter Formeln in einer Zell-basierten Spreadsheet-Anwendung, lässt sich durch Vervielfältigung der Formeln lösen. Die Logik der Vektor-basierten Formel wird auf alle Zellen angewandt, die aus der Ausprägung bzw. der Kombination von

[29] Dieser Beispielrechnung liegt die Annahme zu Grunde, dass keine Modellzellen im Sinne heterogener Strukturen ausgeschlossen sind und alle Ausprägungen uneingeschränkt miteinander kombiniert werden können, ohne dass semantische Widersprüche entstehen.

Ausprägungen entstehen, für die sie definiert ist (scopeItem), abgesehen von der Einschränkung eines Skip-Konstrukts. Das Wissen, welche Dimensionsausprägung an welcher Modellzelle beteiligt ist, liegt im Excel-Spreadsheet jedoch nicht explizit vor. Daher muss dies zuvor bestimmt werden.

6.4.1.2 Ranges von Dimensionsausprägungen

Für das Speichern der Zellbereiche, an denen eine Dimensionsausprägung beteiligt ist, wird im VBA-Makro das von Excel angebotene Objekt `Range` verwendet. Es stellt eine Referenz auf eine bestimmte Zellmenge bzw. einen definierten Zellbereich dar (vgl. Can-Weber und Wendel, 2010, S. 221). In Kombination mit einem Objekt `Names`, mittels dessen einem Zellbereich respektive einem Objekt `Range` ein Bezeichner zugewiesen wird (vgl. Can-Weber und Wendel, 2010, S. 477), kann für jede Dimensionsausprägung die Zellmenge des Modells bestimmt werden, an der sie beteiligt ist. Hierfür wird jeweils ein `Names`-Objekt pro Dimensionsausprägung angelegt, welches auf ein `Range`-Objekt verweist, das den Zellbereich dieser Ausprägung definiert. Für jede Modellzelle – der Modellraum bestimmt sich über die *Spalten-* bzw. *Zeilenbezeichner*, in welche die Dimensionsausprägungen eingetragen wurden – wird festgestellt, welche Dimensionsausprägungen in den betreffenden Spalten der *Spaltenbezeichner* bzw. Zeilen der *Zeilenbezeichner* stehen. Im Anschluss wird die Zelle zu denjenigen Zellbereichen (`Range`-Objekte) hinzugefügt, die über die `Names`-Objekte erreichbar sind, die den in den *Zeilen-* und *Spaltenbezeichnern* vorgefundenen Dimensionsausprägungen entsprechen. Im vorliegenden Beispiel (vgl. Abbildung 6-12) würde sich die Situation für Zelle *D4* wie folgt darstellen:

- Spaltenbezeichner: Zeile 2 und 3, Spalte D
- Zeilenbezeichner: Spalte B und C, Zeile 4
- beteiligte Dimensionsausprägungen: 2011, Produkt A, Umsatz, Nord

Die Zelle *D4* würde demzufolge den `Range`-Objekten zugewiesen werden, die über die `Names`-Objekte 2011, Produkt A, Umsatz und Nord referenziert werden. Im Endeffekt liegt nach diesem Prozess für jede Dimensionsausprägung die Information vor, an welchen Modellzellen sie beteiligt ist. Dies stellt eine Basisinformation dar, auf welcher der weitere Programmablauf aufbaut.

6.4.1.3 Ranges von Formeln

Nach wie vor fehlt die Information, auf welche Modellzellen die definierten Formeln angewandt werden sollen. Der nächste Schritt der Transformation fokussiert die Identifikation dieser Zellen. Diese Information wird wiederum im Spreadsheet-Modell als `Range`-Objekt gespeichert. Ein Zugriff ist über ein `Names`-Objekt mit der ID der Formel als Bezeichner möglich. Für die Feststellung der entsprechenden Zellen wird die Schnittmenge der `Range`-Objekte der Dimensionsausprägungen, aus denen das scopeItem Element einer Formel besteht, gebildet. Ist dies abgeschlossen wird kontrolliert, ob die Formel über einen Skip-Teil verfügt. Die Zellmenge, die ein Skip-Teil repräsentiert, wird gegebenenfalls auf dieselbe Art und Weise ermittelt, wie die eines scopeItem Elements. Abschließend werden diejenigen Zellen, die in der Zellmenge des Skip-Teils und der Zellmenge des scopeItem Elements enthalten sind, aus dem `Range`-Objekt der Formel entfernt. Auf die restlichen Zellen ist die Formel anzuwenden.

Des Weiteren muss bei dieser Berechnung noch unterschieden werden, ob der Zellbereich mittels eines *IN-Statements* oder über die Angabe von Dimensionsausprägungen definiert ist. Das erste scopeItem Element eines IN-Statements repräsentiert einen bestimmten Zellbereich, in der Regel in Form einer Gruppe. Um diesen zu ermitteln, wird für Dimensionsausprägungen derselben Dimension (nur innerhalb einer Gruppe können mehrere Dimensionsausprägungen einer Dimension enthalten sein) die Vereinigungsmenge der entsprechenden `Range`-Objekte (der Ausprägungen) gebildet. Anschließend wird die Schnittmenge dieser Teilmengen (eine Teilmenge je Modelldimension) gebildet und in einem `Range`-Objekt gespeichert. Der Zellbereich des zweiten scopeItem Elements wird wiederum durch Bilden der Schnittmenge der `Range`-Objekte der Dimensionsausprägungen ermittelt. Abschließend wird die Schnittmenge des ersten und des zweiten scopeItem Elements gebildet. Das Ergebnis stellt den Zellbereich im Modell dar, auf welchen die Formel Anwendung finden soll.

Nach diesem Berechnungsschritt bestehen `Names`- und `Range`-Objekte für jede Formel und für jede Dimensionsausprägung des Modells. Diese Informationen werden bei der Auswahl der anzuwendenden Formel je Modellzelle genutzt. Die angewandte Methodik erläutert der folgende Abschnitt 6.4.2.

6.4.2 Formelauswahl und Formeltransformation

Nachdem die Struktur eines Modells in einem Spreadsheet-Modell abgebildet ist, müssen die Formeln eines XML-Dokument-basierten Mod^{2}DSS Modells in eine Excel-kompatible

Form überführt werden. Der grundlegende Unterschied zwischen Mod²DSS und Excel ist die Art der Definition logischer Zusammenhänge bzw. Rechenvorschriften. Während im Mod²DSS Modell durch sprechende Variablen (i.d.R. Dimensionsausprägungen) dem Benutzer die Logik einer Formel semantisch intuitiv verständlich präsentiert wird, werden in einem Excel-Spreadsheet lediglich einzelne Zellen miteinander in Relation gesetzt. Die Semantik dieser Zellen ist im Transformationsschritt nur durch das Einführen von Dimensionsausprägungen bestimmt und somit kein Bestandteil eines Excel-inhärenten Konzepts. Die *Zeilen-* und *Spaltenbezeichner* sind dabei lediglich beschreibender Natur. Es bedarf daher auch hier einer Umwandlung der Formeldefinitionen, um diese wie gewünscht in Excel abbilden zu können.

Hierfür wird Zelle für Zelle überprüft, ob im Modell mehrere Formeln vorliegen, welche auf diese Zelle angewandt werden können. Je Zelle wird hierfür kontrolliert, ob eine Überschneidung mit den `Range`-Objekten der Formeln vorliegt. Ist dies der Fall, wird die `ID` der jeweiligen Formel in einem `Array` gespeichert. Sofern nach der Kontrolle aller Formeln mehrere `IDs` in dem `Array` vorhanden sind, wird der Benutzer gebeten eine Auswahl der anzuwendenden Formel der aktuell untersuchten Zelle zu treffen (vgl. Abbildung 6-13). Es öffnet sich ein Fenster, welches Informationen über die Dimensionsausprägungen der Zelle (linkes Textfeld) und über die zur Auswahl stehenden Formeln (Drop-Down Menü) enthält. Im Drop-Down Menü zur Auswahl der anzuwendenden Formel steht diese in Form der im Mod²DSS hinterlegten Logik. Im XML-Dokument verbirgt sich diese Information hinter dem Element formula-string. Sollte eine der zur Auswahl stehenden Formeln eine Gruppe referenzieren, werden dem Benutzer die Dimensionsausprägungen der Gruppen aufgelistet (rechtes Textfeld). Ist keine Gruppe in den auswählbaren Formeln referenziert, wird das rechte Textfeld ausgeblendet.

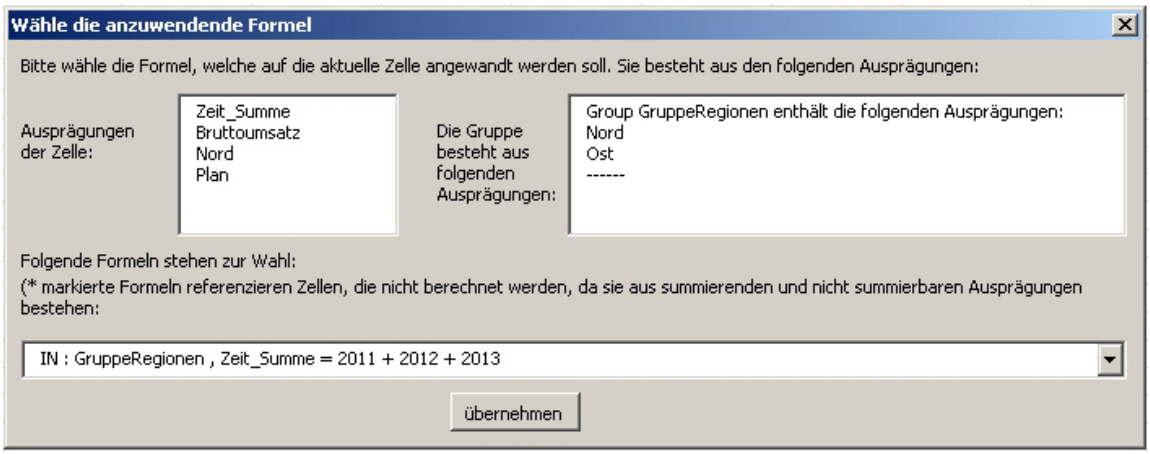


Abbildung 6-13: Auswahl der anzuwendenden Formel je Zelle

Hat der Benutzer eine Auswahl getroffen, muss für die Erstellung des anwendbaren Excel-Modells (Modellkompilierung) im nächsten Schritt die eigentliche Logik der Formel auf die betroffenen Excel-Zellen angewandt werden. Hierfür ist lediglich der Logikteil der ausgewählten Formel relevant.

Um die Zellen des Modells zu identifizieren, die durch die Formel in Beziehung gesetzt werden sollen, wird für jedes mainItem Element bestimmt, welcher Zelle es entspricht. Hierfür werden wieder die zuvor berechneten Range-Objekte je Dimensionsausprägung genutzt. Da jede Zelle eines Modells aus genau einer Dimensionsausprägung je vorhandener Modelldimensionen besteht, lässt sich eine Zelle im Spreadsheet-Modell identifizieren, indem man die Schnittmenge aller Range-Objekte der Dimensionsausprägungen bildet. Besteht ein mainItem Element aus weniger Dimensionsausprägungen, als das Modell Dimensionen aufweist (dies ist aufgrund des Konzepts Vektor-basierter Formeln der Regelfall), wird für die fehlenden Dimensionen jeweils die Ausprägung der Zelle herangezogen, für welche die Formel definiert wird (die aktuell im Auswahlprozess behandelte Zelle). Das folgende Beispiel soll diese Funktion noch einmal verdeutlichen:

	A	B	C	D	E	F	G	H
1		Regionen	Kennzahlen					
2	Zeit			2011	2011	2012	2012	
3	Produkte			Produkt A	Produkt B	Produkt A	Produkt B	
4		Nord	Umsatz					
5		Nord	Menge					
6		Nord	Preis					
7		Nord	Umsatz					
8		Nord	Menge					
9		Nord	Preis					
10								

Abbildung 6-14: Beispiel eines Spreadsheet-Modells in Excel

In dem Beispiel, das in Abbildung 6-14 dargestellt ist, existiere eine Formel, die den *Umsatz* (scopeItem Element) als Produkt aus *Menge* (erstes mainItem Element) und *Preis* (zweites mainItem Element) definiert. Die Modellzelle *D4* besteht aus den Dimensionsausprägungen *Nord*, *Umsatz*, *2011* und *Produkt A*. Sie geht aus der Schnittmenge der Range-Objekte dieser Dimensionsausprägungen hervor. Da für die Dimensionsausprägung *Umsatz* eine Formel hinterlegt ist, muss bestimmt werden, welche Zellen in diesem Fall den Dimensionsausprägungen *Menge* und *Preis* entsprechen. Um zu bestimmen, welcher Modellzelle in diesem Fall die Ausprägung *Menge* entspricht, wird die Schnittmenge der *Range*-Objekte *Menge*, *2011*, *Nord* und *Produkt A* gebildet. Die Dimensionsausprägungen *2011*, *Nord* und *Produkt A* sind dabei die entsprechenden Ausprägungen der Di-

mensionen *Zeit*, *Regionen* und *Produkte*, für die im ersten mainItem Element der Formel (*Menge*) keine einschränkenden Dimensionsausprägungen spezifiziert sind. Es ergibt sich die Modellzelle *D5*. Für die Bestimmung der entsprechenden Modellzelle für das mainItem Element *Preis*, wird analog die Schnittmenge der *Range*-Objekte *Preis*, *2011*, *Nord* und *Produkt A* gebildet. Es ergibt sich die Zelle *D6*. Für die Darstellung der Formel Umsatz = Menge * Preis würde in die Zelle *D4* (*Umsatz*) des Excel-Spreadsheets die Formel *D5 * D6* (*Menge * Preis*) eingefügt werden.

Die logischen Operatoren (+, -, *, /), welche jeweils als XML-Element formula-element vorliegen, werden in der Reihenfolge der Definition der Formelelemente im XML-Dokument in die Formel der Zelle eingefügt. Genauer betrachtet werden müssen wiederum die Elemente PREV, NEXT sowie IF-THEN-ELSE Verkettungen. PREV bzw. NEXT erfordern in Bezug auf die Methodik zur Feststellung einer Zelle im Spreadsheet-Modell, dass es eine korrespondierende Dimension gibt, in welcher nicht die Dimensionsausprägung berücksichtigt wird, die im scopeItem Element vertreten ist, sondern der jeweilige Vorgänger (PREV) bzw. Nachfolger (NEXT). In der Regel ist die korrespondierende Dimension eine Zeitdimension, so dass durch NEXT bzw. PREV Bezüge zwischen Zeitperioden innerhalb eines Modells aufgebaut werden können (siehe Kapitel 5.2.5). Ein Beispiel hierfür wäre die Berechnung einer Kennzahl *Lagerbestand*, welche den *Lagerbestand einer vorhergehenden Periode* bei der Berechnung der aktuellen Periode berücksichtigt. Damit der Benutzer möglichst uneingeschränkt modellieren kann und es keine Dimensionen innerhalb eines Modells gibt, denen eine Sonderaufgabe zukommt, kann als korrespondierende Dimension jegliche Modelldimension angegeben werden. IF-THEN-ELSE Bedingungen sind innerhalb des Prototyps auf eine genau vorgegebene Struktur eingeschränkt (vgl. Kapitel 5.2.5). Dieser Ausdruck wird in Excel durch die *WENN*- bzw. *If*-Funktion (abhängig von der eingestellten Sprache im Excel-Spreadsheet) ersetzt, welche dieselbe Struktur aufweist.

Bis hierher sind sämtliche Funktionen bzw. Mechanismen auf VBA-Ebene erklärt, die für den generellen Aufbau eines Spreadsheet-Modells auf Basis eines Mod^{2}DSS-Modells nötig sind. Im nächsten Abschnitt 6.4.3 wird erläutert, wie die Transformationsphase abgeschlossen und das überschneidungsfreie Modell gespeichert wird.

6.4.3 Speichern des überschneidungsfreien Modells

Beim Durchlaufen des Modells Zelle für Zelle werden einige Informationen gespeichert, welche insbesondere für das Speichern des Excel-Modells als XML-Dokument relevant

sind. In einem weiteren Tabellenblatt des Excel-Spreadsheet wird für alle mit mindestens einer Formel belegten Zellen festgehalten, aus welchen Dimensionsausprägungen sie bestehen, welche Formel der Benutzer ausgewählt hat und welche weiteren Formeln zur Auswahl standen (sofern dies der Fall war). Abbildung 6-15 zeigt anhand eines frei gewählten Beispiels ein solches Tabellenblatt. In den ersten vier Spalten (A bis einschließlich D) werden den vier Dimensionen des Beispielmodells entsprechend die Dimensionsausprägungen je Zelle, in Spalte E die vom Benutzer gewählte und in den Spalten F bis H die übrigen zur Verfügung stehenden Formeln dieser Zellen gespeichert.

H20

	A	B	C	D	E	F	G	H	I	J
1	0	1	2	3					1	8
2	**Dimension 1**	**Dimension 2**	**Dimension 3**	**Dimension 4**	**Formel 1**	**Formel 2**	**Formel 3**	**Formel 4**	**DimItems of Dim 0 und Formel _000**	**DimItems of Dim 1 und Formel _000**
3	Zeit_Summe	Bruttoumsatz	Nord	Abweichung	_000	_100	_300		Zeit_Summe	Bruttoumsatz
4	Zeit_Summe	Absatzmenge	Nord	Abweichung	_000	_300				Absatzmenge
5	Zeit_Summe	Rabattsatz	Nord	Abweichung	_000	_103	_300			Rabattsatz
6	Zeit_Summe	Rabatt	Nord	Abweichung	_000	_102	_300			Rabatt
7	Zeit_Summe	Nettoumsatz	Nord	Abweichung	_000	_101	_300			Nettoumsatz
8	Zeit_Summe	Einkaufsmeng	Nord	Abweichung	_000	_300				Einkaufsmenge
9	Zeit_Summe	Lagerbestand	Nord	Abweichung	_000	_105	_300			Lagerbestand
10	Zeit_Summe	Gewinn	Nord	Abweichung	_000	_104	_300			Gewinn

Abbildung 6-15: „Logging“ von Informationen der Formelauswahl

Im weiteren Verlauf werden diese Informationen nun für die Umformung der Zell-basierten Formeldefinition des Excel-Spreadsheets in die Vektor-basierte Formeldefinition der konzeptuellen Mod^{2}DSS Modelle ausgewertet. Zuerst werden die Spalten mit den Dimensionsausprägungen und Formeln (in Abbildung 6-15 die Spalten A bis H) entsprechend der gewählten Formel sortiert (im Beispiel Spalte E). Dies führt dazu, dass alle Zellen, die mit derselben Formel belegt sind hintereinander in einem Block gespeichert werden. Dies ist für die weiteren Auswertungen relevant, da der prozedurale Programmablauf des VBA-Codes nicht auf individuell definierte Objekte zurückgreifen und auf sie Methoden anwenden kann, wie es in Java der Fall ist. Um im XML-Dokument des transformierten Excel-Modells nicht jede Zelle, auf die eine Formel angewandt werden soll, einzeln speichern zu müssen, wird folgende Methode angewandt:

Für jede Formel wird identifiziert, welche Dimensionsausprägungen pro Dimension von der Formel berührt werden. Die Ergebnisse werden wiederum in den Spalten des Tabellenblatts abgespeichert, die auf die Spalten für die Formeln folgen (im Beispiel der Abbildung 6-15 sind dies die Spalten ab I). Für *Dimension 2* wird demnach die Formel mit der ID *000* auf Zellen angewandt, die aus den Dimensionsausprägungen *Bruttoumsatz*, *Absatzmenge*, *Rabattsatz*, *Rabatt*, *Nettoumsatz*, *Einkaufsmenge*, *Lagerbestand* und *Gewinn* bestehen. Anschließend wird basierend auf diesen Informationen berechnet, welche Zellen aus diesen Dimensionsausprägungen kombiniert werden können, was einer Art kartesischem Produkt entspricht. Im XML-Dokument des Excel-Spreadsheet werden diese Dimensionsauspräg-

gungen in einem XML-Element Group gespeichert. Dieses Element wird in der Definition der Formel im scopeItem Element mit einem IN-Statement referenziert. Stößt der Transformator in einem scopeItem Element auf eine solche Klausel (IN gefolgt von Group) interpretiert dieser die Inhalte als kartesisches Produkt, so dass die Formel auf sämtliche Zellen angewandt wird, die aus den Kombinationsmöglichkeiten der in der Gruppe enthaltenen Dimensionsausprägungen hervorgehen. Da es jedoch möglich ist, dass bei der Kombination der Dimensionsausprägungen des Group-Elements Zellen berücksichtigt werden, für welche ursprünglich die Formel nicht ausgewählt wurde, findet ein Abgleich mit eben diesen Zellen statt (im Beispiel Abbildung 6-15 die Spalten A bis D). Taucht eine Zelle, die aus den Kombinationen der Dimensionsausprägungen der Modelldimensionen und Formeln hervorgeht (im Beispiel Abbildung 6-15 die Spalten ab I) in diesen Zelldefinitionen nicht auf, wird diese Zelle im Skip-Teil der Formel berücksichtigt. Der Transformator übergeht bei der Interpretation eines solchen Elements die entsprechenden Zellen für die jeweilige Formel.

Das Resultat dieser komplexen Methode ist ein XML-Dokument, welche ein überschneidungsfreies Modell basierend auf der Formelauswahl des Benutzers darstellt. Das resultierende Modell entspricht nach wie vor den Anforderungen der Multidimensionalität und besteht aus Vektor-basierten Formeln, obwohl es während der Transformation hilfsweise in ein flaches, zweidimensionales Excel-Modell umgewandelt wurde.

6.5 Limitationen des Prototyps

Prototypen ermöglichen es, Konzepte zu analysieren und auf Schwachstellen zu untersuchen, die ohne eine Implementierung nicht oder nur unter sehr großem Aufwand aufgedeckt werden können. Limitationen sind ein zentrales Merkmal prototypischer Applikationen (vgl. Naumann und Jenkins, 1982, S. 32). Da die Implementierung sämtlicher Funktionen und Eigenschaften eines Konzepts i.d.R. sehr zeitaufwändig ist, werden lediglich die zentralen Funktionen und Teilkonzepte in der prototypischen Realisation eines Konzepts umgesetzt. In Bezug auf den Prototyp Mod^{2}DSS sind ebenfalls einige Funktionen und Konzepte, die in den Kapiteln 4 und 5 dargestellt sind, nicht oder lediglich in vereinfachter Form implementiert.

Der Modellierungseditor bietet während der Modellierung kein Konstrukt zum Ausschluss von Modellzellen an. Inhomogenität bzw. heterogene Strukturen können in dieser Modellierungsphase daher lediglich durch die Einschränkung des Bezugsteils von Formeln abge-

bildet werden. Hinsichtlich des speziellen Operators PREV bzw. NEXT (siehe Abschnitt 5.2.5) findet weder eine automatische Ermittlung des Skip-Teils für die erste oder letzte Dimensionsausprägung der korrespondierenden Dimension statt, noch wird die Definition einer Formel für diese Ausprägung angeboten. Beim Anlegen der Modellstruktur muss des Weiteren die vorgegebene Beschränkung von zehn Dimensionen beachtet werden. Außerdem können die Reihenfolgen von Dimensionen, Dimensionsausprägungen und Formeln derzeit nicht verändert werden. Der Anwender muss über dies hinaus sicherstellen, dass nicht zwei oder mehrere Modellelemente unterschiedlicher Elementtypen denselben Bezeichner erhalten. Dies könnte zur Folge haben, dass z.B. innerhalb einer Formeldefinition nicht unterschieden werden könnte, ob eine Dimension oder eine Dimensionsausprägung definiert wird.

Das Repository, welches Bestandteil der Konfigurationsphase des Konzepts ist, wird im Prototyp nur rudimentär umgesetzt. Im Prototyp werden die Inhalte eines Repository lediglich in einem XML-Dokument gespeichert, deren Struktur sehr stark denen eines XML-Dokuments eines Modells gleicht. Weitergehende Überlegungen, wie z.B. die referenzielle Integration von Repository und Modell oder die Verfügbarkeit eines Repository über Web-Technologien wurden explizit nicht verfolgt.

Bezüglich der XML-Schnittstelle ist anzumerken, dass die Umsetzung des ID-Konzepts im Modellierungseditor und der Transformationskomponente nicht der XML Schema-Konvention entspricht. Für IDs wird aus Gründen der Performance Steigerung im Gegensatz zur XML Schema-Konvention eines Strings im Prototyp ein numerischer Datentyp verwendet. Die Vergleiche numerischer Werte bedeuten für die Weiterverarbeitung von Modellinformationen in der Transformationskomponente mittels numerischer Datentypen weniger Aufwand.

Eine Effizienz steigernde Maßnahme spiegelt sich in Bezug auf das Konzept in der gemeinsamen Implementierung von Transformations- und Kompilierungskomponente wider. Aufgrund der guten Integration von VBA und Excel kann bereits im Zuge der Transformation eine Kompilierung der transformierten Modellspezifikationen mit Excel als Zielsystem stattfinden. Hierzu wird ein Spreadsheet-Modell innerhalb eines Tabellenblatts in Microsoft Excel aufgebaut, welches z.B. für Sensitivitäts- oder What-If Analysen zur Verfügung stünde. Die Umsetzung der Kompilierung innerhalb von Excel führt jedoch dazu, dass simultane Bezüge für die Übersetzung der transformierten Inhalte nicht berücksichtigt werden. Dies ist der Fall, da zirkuläre Bezüge in Excel-Modellen nicht berücksichtigt wer-

den können, sondern in einem vorhergehenden Schritt aufgelöst werden müssten. Bei der Implementierung von Mod^{2}DSS wurde auf die Umsetzung dieses sehr aufwändig umzusetzenden Schrittes verzichtet.

Konsistenzinformationen werden im Prototyp nur rudimentär, z.B. beim Löschen von Modellelementen, generiert und ausgewertet. Die Transformationskomponente muss daher in jedem Fall selbst die Überschneidungen von Formeln innerhalb einzelner Zellen ermitteln, anstatt die Modellierungsfälle der Formeln auszuwerten.

Während der Transformationsphase kann der Anwender nicht zurück in die Konfigurationsphase (Modellierungseditor) springen. Es besteht lediglich die Möglichkeit, am Ende der Transformationsphase des Modellierungsprozesses zurück in die Konfigurationsphase zu springen. Während der Transformation können die Auswirkungen der unterschiedlichen Formeln auf den Rest des Modells zudem nicht analysiert werden und es besteht in der Dialogkomponente keine Möglichkeit, die Formelauswahl auf einen definierbaren Zellbereich anzuwenden. Jede Zelle, die mit mehr als einer Formel belegt ist, muss daher vom Anwender in einem Dialog spezifiziert werden. Eine getätigte Formelauswahl kann darüber hinaus nur durch Neustart der gesamten Transformationsphase wieder geändert werden.

Abschließend ist festzustellen, dass sämtliche Einschränkungen des Prototyps für die Beurteilung von Machbarkeit und Evaluation als nicht gravierend anzusehen sind. Dies ist der Fall, da der Kern des erstellten Konzepts implementiert wurde und lediglich Funktionen unberücksichtigt geblieben sind, die zu einer Verbesserung der Bedienbarkeit und einer intensiveren Unterstützung des Anwenders führen würden.

7 Evaluation

Kapitel 6 hat den entwickelten Prototyp dargestellt und ist dabei insbesondere auf die angewandten Techniken sowie die Entwicklungsumgebung eingegangen. Die eigentliche Anwendung des Prototyps steht im vorliegenden Kapitel im Fokus. Anhand eines konkreten Modellierungsbeispiels wird das erarbeitete Konzept hinsichtlich des Potenzials zur nachhaltigen Verbesserung der Qualität definierter Modelle getestet. Zu diesem Zweck wurde eine Modellierungsaufgabe von zwei Studierendengruppen gelöst. Eine Gruppe nutzte für die Bearbeitung der Aufgabe das Spreadsheet-Werkzeug Microsoft Excel und eine weitere Studierendengruppe nutzte den entwickelten Prototyp Mod^{2}DSS, so dass die Ergebnisse bezüglich der Modellqualität verglichen werden konnten.

7.1 Rahmenbedingungen und Vorgehen

Die Evaluation des Prototyps wurde im Rahmen der Übungsveranstaltung zur Vorlesung *Management Support Systeme 2 – Decision Support Syste*me im Sommersemester 2011 am Fachbereich Wirtschaftswissenschaften der Universität Osnabrück durchgeführt. Sie wurde im Sinne eines *Proof of Concept* durchgeführt und nutzt die prototypische Applikation für die Beurteilung des Konzepts. Es nahmen 23 Studierende teil, von denen 14 im Diplomstudiengang *Betriebswirtschaftslehre* (mindestens sechstes Fachsemester), acht im Bachelorstudiengang *Wirtschaftswissenschaft* (mindestens sechstes Fachsemester) sowie einer im Bachelorstudiengang *Information Systems* (Fachsemester acht) eingeschrieben waren. Als Räumlichkeit stand ein Computer-Pool des Fachbereichs zur Verfügung, so dass jedem Teilnehmer ein Computer zur Verfügung stand. Die Teilnehmer befassten sich im Rahmen der Vorlesung explizit mit dem Themengebiet *Modellgetriebene (Spreadsheet-basierte) Decision Support Systeme* und haben diesbezüglich auch den Bereich der *multidimensionalen Modellierung* kennengelernt. Aus diesem Grund ist festzuhalten, dass sämtliche Teilnehmer bereits über einschlägige Vorkenntnisse verfügten.

Die Evaluation bestand aus einer praktischen Modellierungsaufgabe und einem anschließenden Fragebogen (siehe Anhang I: Fragebogen). Mittels des Fragebogens wurden Informationen zu den Themenbereichen *Vorkenntnisse zum Thema Modellierung*, *Modellierungsprozess* und *Bedienbarkeit*, sowie eine Schilderung *subjektiver Eindrücke* erhoben. Die Fragen hatten dabei alle einen engen Bezug zu der vorausgehenden Modellierungsaufgabe.

Für die Bearbeitung der Modellierungsaufgabe hat jeder Teilnehmer zu Beginn aus einer Urne einen Zettel mit einer Teilnehmernummer gezogen. Die Teilnehmer eins bis zwölf bearbeiteten die Aufgabe anhand des Prototyps Mod^{2}DSS, während die Teilnehmer 13 bis 23 das Spreadsheet-Werkzeug Microsoft Excel benutzten.

Zu Beginn der Evaluation haben sämtliche Teilnehmer eine kurze Einführung in die Bedienung beider Werkzeuge erhalten. Es wurde anhand von Beispielen auf sämtliche Funktionen und Konzepte des entwickelten Prototyps Mod^{2}DSS sowie von Microsoft Excel eingegangen, die für die Bearbeitung der anschließenden Modellierungsaufgabe relevant waren. Da in der Modellierungsaufgabe typische multidimensionale Aspekte wie Dimensionen und Ausprägungen vorhanden waren, die in Excel nicht in dieser oder einer vergleichbaren Form vorhanden sind, wurde im Laufe der Erläuterungen explizit auf mögliche Lösungsstrategien für die Bearbeitung derartiger Probleme mit Excel eingegangen. Es wurden die Möglichkeiten dargestellt, Dimensionen ineinander zu projizieren (vgl. Abschnitt 6.4.1.1), wie es im Transformationsschritt des Prototyps automatisiert geschieht, verschiedene Tabellenblätter als Dimension zu nutzen und durch Modellierung mehrerer Tabellen auf einem Tabellenblatt Dimensionen zu simulieren. Jeder Teilnehmer erhielt des Weiteren diese Hinweise für die Nutzung des Modellierungseditors (siehe Anhang II: Modellierungshinweise Prototyp) sowie für Microsoft Excel (siehe Anhang III: Modellierungshinweise Excel) als Stichpunkt-artige Zusammenfassung. Dies stellte sicher, dass alle Teilnehmer die nötigen Kenntnisse besitzen und eine Bearbeitung der Aufgabe nicht an der mangelnden Kenntnis des Werkzeugs scheitern konnte.

Im folgenden Abschnitt 7.2 wird nun auf die Inhalte der Evaluation eingegangen.

7.2 Modellierungsaufgabe

Die zu bearbeitende Modellierungsaufgabe war für alle Teilnehmer gleich, unabhängig davon ob eine Bearbeitung mittels Microsoft Excel oder Mod^{2}DSS erfolgt ist. Damit jeder Teilnehmer die Aufgabe bearbeiten konnte, wurde ein Modell gewählt, das möglichst knapp gehalten ist, ohne dabei jedoch auf eine der typischen Modellierungssituationen zu verzichten. Tabelle 2 gibt einen Überblick der Modellstruktur der Aufgabe. Insgesamt waren in vier Dimensionen 18 Dimensionsausprägungen zu definieren:

Tabelle 2: Struktur der Modellierungsaufgabe

Dimensionen	**Dimensionsausprägungen**
Zeit	2011, 2012
Kennzahlen	Bruttoumsatz, Absatzmenge, EK_Preis, Rabattsatz, Rabatt, Nettoumsatz, Einkaufsmenge, Lagerbestand, VK_Preis, Gewinn
Regionen	Nordost, Südwest, Region_Summe
Version	Plan, Ist, Abweichung

Zusätzlich zu diesen strukturellen Aspekten sollte das Modell mit einigen logischen Zusammenhängen versehen werden. Der Großteil der Modelllogik ist in der Dimension Kennzahlen zu finden. Nur die Zeitdimension sollte keine Dimensionsausprägungen enthalten, die mit einer Formel berechnet werden sollten. Dies führte im Endeffekt zu einigen Modellzellen, die mit zwei oder maximal drei Formeln belegt sein können. Die logischen Zusammenhänge stellt Tabelle 3 dar:

Tabelle 3: Logische Aspekte der Modellierungsaufgabe

Dimensionsausprägung	**Berechnung / Logik**
Bruttoumsatz	Absatzmenge multipliziert mit dem VK_Preis
Rabattsatz[30]	sofern die Absatzmenge größer als 500 ist, wird ein Rabattsatz von 10% gesetzt, alles darunter bleibt bei 0%
Rabatt	Bruttoumsatz multipliziert mit dem Rabattsatz
Nettoumsatz	Bruttoumsatz reduziert um den gewährten Rabatt
Gewinn	Nettoumsatz abzüglich der Einkaufsaufwendungen (Einkaufsmenge multipliziert mit dem EK-Preis)

[30] Das Modell soll den Umsatz eines einzelnen Kunden widerspiegeln.

Dimensionsausprägung	**Berechnung / Logik**
Lagerbestand	Lagerbestand der vorausgehenden Periode, addiert um die Einkaufsmenge, abzüglich der Absatzmenge (jeweils der aktuellen Periode)
Region_Summe	Summe von Nordost und Südwest
Abweichung	Ist- abzüglich Plan-Wert

Aus Tabelle 3 wird ersichtlich, dass lediglich die Definition der Dimensionsausprägung *Rabattsatz* nicht mittels der Grundrechenarten Addition, Subtraktion und Multiplikation abgebildet werden konnte. Im Falle des Rabattsatzes ist für die Umsetzung im Prototyp eine *IF-THEN-ELSE* Klausel vorgesehen. Microsoft Excel bietet mit der *WENN-Funktion* (engl. if) ein sehr ähnliches Konstrukt an. Hinsichtlich der Ausprägung *Lagerbestand* musste ein Bezug auf einen Wert der vorausgehenden Periode definiert werden (siehe Kapitel 5.2.5: PREV). Für Zellen der Dimensionsausprägungen *Lagerbestand* und *2012* sollte dementsprechend der Wert der Zelle des *Lagerbestands* und *2011* als Ausgangsgröße dienen.

Insgesamt beschrieben die vier Dimensionen mit 18 Ausprägungen einen Modellraum, der aus 180 Zellen bestand. Zellen, in denen eine Summenformel auf eine nicht summierbare Ausprägung traf, z.B. Region_Summe und VK_Preis, konnten unterschiedlich interpretiert werden. Einerseits konnten sie durch eine weitere Formel, die beispielsweise einen gewichteten Durchschnitt berechnete, spezifiziert werden, andererseits wurde ein Auslassen der Formel ebenfalls nicht als Fehler interpretiert, da der Prototyp bei der Kompilierung des konzeptuellen Modells in ein Excel Spreadsheet-Modell solche Zellen ebenfalls auslässt (vgl. Abschnitt 6.4.1.1).

7.3 Fehlermetrik

Für die Auswertung der Ergebnisse wurde eine Fehlermetrik definiert, die eine Aussage über die Feststellung der Fehler macht. Die Metrik beschränkt sich auf die Dimension *Anzahl fehlerhafter Zellen*. Eine Zelle ist fehlerhaft, wenn:

- sie durch eine logisch falsche Formel definiert wird,
- eine aus der Aufgabenstellung hervorgehende Formelspezifikation fehlt,

- eine Formelspezifikation vorgenommen wurde, obwohl diese aus der Aufgabenstellung für die entsprechende Zelle nicht hervorgeht,
- oder wenn sie sich fälschlicherweise nicht im Modell wiederfindet.

Eine denkbare Unterscheidung von Fehlern kann nur dann sinnvoll stattfinden, wenn der gesamte Erstellungsprozess eines Modells erfasst wird. Selbst bei dieser aufwändigen Methode können einzelne Fehler jedoch häufig nur richtig klassifiziert werden, indem der Modellersteller befragt wird (vgl. Panko und Sprague, 1998, S. 342). Aus diesem Grund wurde im Rahmen der Evaluation des Prototyps keine Unterscheidung oder Klassifizierung nach der Schwere eines Fehlers vorgenommen, so dass eine denkbare Berücksichtigung differenzierter Gewichtungen je Fehlerklasse nicht in die Auswertung eingeflossen ist.

Die Modellierungsfehler wichen in Bezug auf das benutzte Werkzeug sehr stark voneinander ab, da vor allem bei der Modellierung der Aufgabe mittels Microsoft Excel der Anwender viele Möglichkeiten hatte, die nicht vorhandenen multidimensionalen Konzepte durch geeignete Konstruktionen zu ersetzen oder zu simulieren. Aber auch bei der Nutzung von Mod^{2}DSS gab es Situationen, die unterschiedlich interpretiert werden konnten und für die mehrere vertretbare Lösungen vorhanden waren. Die Fehlermetrik definiert daher in Abschnitt 7.3.1 Fehler, die bei der Nutzung beider Werkzeuge auftreten konnten und erläutert in den folgenden Abschnitten 7.3.2 und 7.3.3 Besonderheiten der Werkzeuge, die für die Beurteilung von Fehlern und deren Entstehung relevant sind.

7.3.1 Allgemeine Fehler

Ein typischer Fehler, der bei der Bearbeitung der Aufgabe in beiden Werkzeugen aufgetreten ist, ist das Auslassen von Spezifikationen. Dieser Fehler wurde z.B. auch von Panko und Sprague oder Rajalingham et al. in ihren Klassifizierungsvorschlägen von Fehlerarten in Spreadsheet-Modellen definiert (vgl. Panko und Sprague, 1998, S. 342; Rajalingham et al., 2000, S. 25f.). Dies konnte Dimensionen, Dimensionsausprägungen oder auch Formeln umfassen, die nicht spezifiziert wurden. Im Falle von Dimensionen und deren Ausprägungen führte dies in jedem Fall dazu, dass das Modell bestimmte Zellen nicht beinhaltete, die entsprechend als fehlerhaft zu werten waren. Das Auslassen von Formeln führte jedoch nicht dazu, dass jede Zelle, die mit dieser Formel laut Aufgabe hätte belegt sein können, als falsch definiert zu bewerten waren. Dies lag daran, dass die Berechnung des Zellwertes im Sinne des Anwenders durch andere Formeln stattfinden konnte. Fehlerhaft zu bewerten waren demnach nur solche Zellen, die durch die fehlende Formeldefinition als leere Zelle bzw. als Zelle interpretiert wurde, die Eingabedaten enthält.

War eine Formel falsch definiert, fehlte beispielsweise die Variable eines Summanden oder wurde eine falsche Variable verwendet, so waren alle Zellen, auf welche diese Formel angewandt wurde, als fehlerhaft zu bewerten. Dies galt auch für den Fall, dass eine korrespondierende Dimension für einen PREV-/NEXT-Operator falsch angegeben wurde. In der Regel ist in diesen Fällen eine Zeitdimension gefordert (siehe Kapitel 5.2.5), wie auch in der Modellierungsaufgabe hinsichtlich des *Lagerbestandes*. Bezüglich der Spezifikation dieser Formel hat die Aufgabenstellung keine Angabe darüber gemacht, wie die Zellen des *Lagerbestandes* für das Jahr *2011* definiert werden sollten. Aus diesem Grund wurden sowohl Zellen des *Lagerbestandes* für das Jahr *2011* als korrekt spezifiziert gewertet, die durch *keine* Formel spezifiziert wurden, als auch Zellen, die mit der Formel *Einkaufsmenge – Absatzmenge* (ein Wert des Lagerbestands einer vorausgehenden Periode 2010 war nicht Teil der Modellierungsaufgabe) belegt waren.

7.3.2 Besonderheiten der Prototyp-Modelle

Bei der Fehlerdefinition für Modelle, die mit dem Prototyp Mod^{2}DSS modelliert wurden, gab es spezifische Eigenschaften, die zu fehlerhaften Zellspezifikationen führen konnten.

Die erste spezifische Eigenschaft ist die Anwendung der definierten Vektor-basierten Formeln. Diese Art der Formeldefinition unterscheidet sich grundlegend von Zell-basierten Formeln, wie sie in Microsoft Excel zu finden sind (vgl. Kapitel 3.2.2). Dies führte dazu, dass einerseits weniger Einzelspezifikationen vorgenommen werden mussten, sich andererseits aber Fehler in einer Formel auf große Teile des Modells auswirken konnten. Dies gilt generell für die Modellierung unter Berücksichtigung des Konzepts der Multidimensionalität.

Sehr drastische Auswirkungen auf das finale Spreadsheet-Modell hatte auch die falsche Nutzung der Attribute *summierbar* und *summiert* von Dimensionsausprägungen. Wurde eines der beiden Attribute einer Ausprägung der Modellierungsaufgabe falsch gesetzt, wurden unter Umständen größere Teilmengen des Modells ausgeschlossen. Modellzellen, die aufgrund eines solchen Fehlers ausgeschlossen wurden, waren als fehlerhafte Zellen zu interpretieren und dementsprechend gewertet (gezählt).

7.3.3 Besonderheiten der Excel-Modelle

Spreadsheet-Modelle werden in Microsoft Excel Zell-basiert definiert und es stehen maximal drei Objekte zur Repräsentation von Dimensionen für die Modellierung zur Verfügung (Ordinate, Abszisse und Tabellenblätter; vgl. Abschnitt 6.4.1).

Excel bietet dem Anwender mit der Funktion *Ausfüllen* eine Unterstützung im Modellerstellungsprozess an, welche ähnliche Auswirkungen hat, wie es bei Vektor-basierten Formeln der Fall ist. Mittels dieser Funktion lassen sich Formeln von Zellen auf Zellbereiche übertragen. Mittels absoluter und relativer Verweise lässt sich dabei steuern, ob während des Kopiervorgangs die Zellbezüge entsprechend der Zeile bzw. Spalte des Kopierziels im Spreadsheet-Modell angepasst werden.

Es wurde in der Aufgabenstellung keine Angabe über das Ausschließen von Zellen in Excel-Modellen gemacht. Dies führte dazu, dass sowohl *leere* als auch *farblich markierte* Zellen nicht als Fehler gewertet.

Der folgende Abschnitt 7.4 stellt die Ergebnisse der Evaluation dar und bezieht dabei die Ergebnisse des Fragebogens mit ein.

7.4 Ergebnisse und Interpretation

7.4.1 Grundgesamtheit

Die Grundgesamtheit für die Auswertung der Ergebnisse bestand aus insgesamt 22 Modellen, da ein Excel-Modell einen Dateifehler aufwies, so dass es für die Auswertung nicht mehr geöffnet werden konnte. Es flossen zwölf Modelle in die Evaluation ein, die mit dem Prototyp Mod^{2}DSS definiert und zehn Modelle, die direkt in Excel erstellt wurden. Vier von allen Modellen enthielten sehr viele Fehler hinsichtlich der Modellstruktur, teilweise fehlte eine beträchtliche Anzahl von Zellen. Hier handelte es sich um zwei mittels des Prototyps und zwei mittels Excel erstellte Modelle. In den zwei betreffenden Prototyp-Modellen wurden die Attribute summiert und summierbar der Dimensionsausprägungen falsch gesetzt. Dies führte in einem Fall zu 38 und in einem weiteren zu 74 fälschlicherweise ausgeschlossenen Modellzellen. Die zwei mittels Excel erstellten Modelle wiesen ebenfalls eine fehlerhafte Modellstruktur auf, die in beiden Fällen auf ein falsches Vorgehen beim Lösen des Problems der Mehrdimensionalität zurückzuführen ist. In einem Fall fehlten daher 120 und in einem weiteren 140 der 180 zu modellierenden Zellen. Eine sinnvolle Interpretation der Qualität war bei diesen Modellen nicht möglich, sodass diese vier Ausreißer nicht in die Berechnung der Ergebnisse eingeflossen sind. Die Aussagekraft der Ergebnisse wäre insgesamt negativ beeinflusst worden, da in lediglich drei der übrigen Modelle jeweils sechs Zellen aufgrund falscher Attributwerte (summierbar und summiert) unbegründet ausgeschlossen wurden. Die restlichen Modelle waren strukturell nicht zu beanstanden. Die im Folgenden dargestellten Ergebnisse basieren also auf der Auswertung

von acht mittels Excel und zehn mittels des Prototyp Mod²DSS erstellten Modellen. Hinsichtlich der Auswertung der Fragebögen ist zu bemerken, dass die Antworten *aller* Teilnehmer in die Auswertung eingeflossen sind, auch sofern deren Modelle nicht in der Auswertung der weiteren Ergebnisse berücksichtigt wurden.

7.4.2 Ergebnisse der Modellauswertung

Die Ergebnisse der Modellauswertung stellen im Folgenden die beobachtete Modellqualität mittels ausgewählter Kennzahlen dar. Die zentrale Kennzahl für die Beurteilung der unterschiedlichen Qualität der Modelle stellt die *absolute Anzahl der fehlerhaft spezifizierten Modellzellen* dar. Die zehn in die Auswertung einbezogenen Mod²DSS-Modelle wiesen eine durchschnittliche Anzahl von 9,4 nicht korrekt definierten Zellen auf. In den acht begutachteten Excel-Modellen liegt die durchschnittliche Anzahl bei 25,38 fehlerhaften Zellen. In Bezug auf die Gesamtzahl der Modellzellen entspricht dies einem Wert von 5,22 % (Mod²DSS) bzw. 14,09 % (Excel). Hauptursache der Modellierungsfehler sind mangelhafte Formeldefinitionen. Abbildung 7-1 visualisiert diese Ergebnisse anhand eines Säulendiagramms:

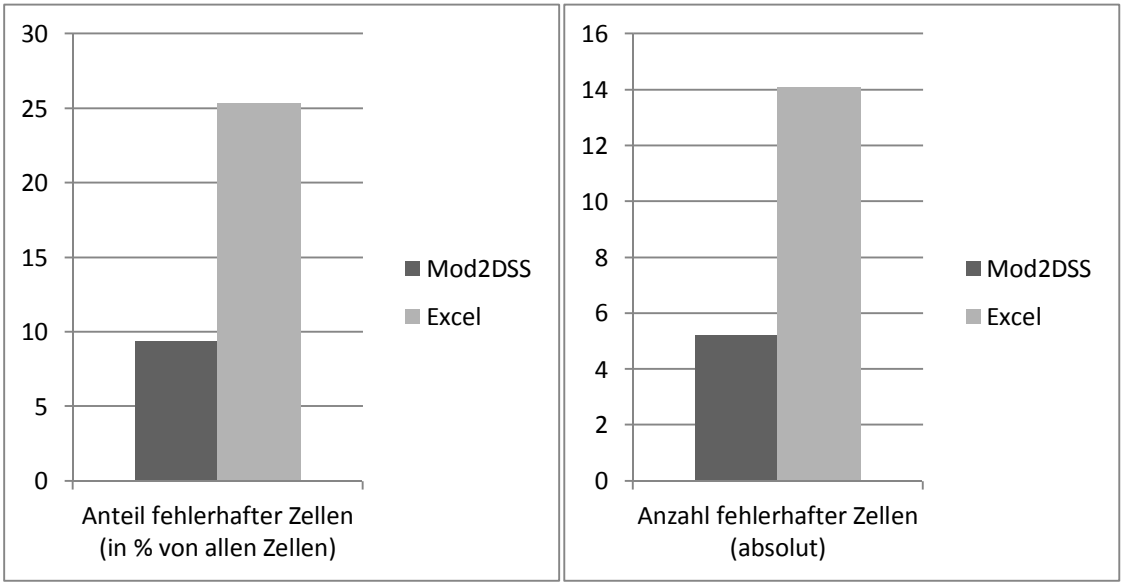


Abbildung 7-1: Fehlerhafte Zellen je Modellierungswerkezug

Überlegungen zur statistischen Signifikanz der beobachteten Ergebnisse müssen aufgrund der beschränkten Stichprobe verworfen werden. Das Ergebnis kann dennoch als Indiz gewertet werden, dass die Fehleranfälligkeit der Spreadsheet-Modellierung durch den Einsatz des erstellten Konzepts positiv beeinflusst wird, da die Werte hinsichtlich der unterschiedlichen Werkzeuge sehr stark voneinander abweichen, sich insgesamt jedoch in einem ähn-

lichen Bereich bewegen, wie es z.B. in einer Studie über Fehler in Spreadsheet Modellen von Janvrin und Morrison (Janvrin und Morrison, 1996) der Fall ist.

Das Ergebnis kann des Weiteren dazu beitragen, die Bereitschaft und das Vertrauen der Benutzer hinsichtlich des Einsatzes von MDSS generell zu steigern, da die Fehleranfälligkeit von Spreadsheet-Modellen durch den Einsatz von Mod^{2}DSS entschärft wird.

Hinsichtlich der Auswertung aller Modelle ist des Weiteren herauszustellen, dass Werkzeug-unabhängig kein Fehler gefunden werden konnte, der sich auf lediglich eine Modellzelle beschränkt. Dies wird sicherlich durch die Aufgabenstellung beeinflusst, die ein mehrdimensionales Modell vorgibt. Eine multidimensionale Aufgabenstellung reicht jedoch nicht aus, um den zu vermutenden Vorteil, dass sich Formelfehler bei Zell-basierter Modellierung auf kleinere Teile eines Modells auswirken, auszuschließen. Es wäre trotzdem zu vermuten, dass die Anfälligkeit für Fehler, die sich auf größere Teile des Modells auswirken, bei Zell-basierter Formeldefinition in Excel geringer ist. Dies konnte nicht bestätigt werden. Hinsichtlich der durchschnittlich beeinflussten Zellen eines Modellierungsfehlers konnte festgestellt werden, dass sich ein Fehler in einem Excel-Modell auf 14,47 Zellen und in Mod^{2}DSS-Modellen auf 10,44 Zellen auswirkt. Als Grund hierfür kann eine fehlerhafte Nutzung der Funktion *Ausfüllen* sowie *absoluter* und *relativer Zellbezüge* ausgemacht werden. Die Zell-basierte Modellierung von Formeln in Microsoft Excel-Modellen wird durch das Kopieren einer Formel in weitere Zellen eines Modells zu einer Art Vektor-basierter Formeldefinition. Dies wurde dafür genutzt, um bei Zell-basierter Formeldefinition in Excel auf den Grund der fehlerhaften Spezifikation einer bzw. mehrerer Zellen zu schließen. Wiederholte sich ein (charakteristischer) Fehler in mehreren Zellen immer wieder und änderte sich gleichzeitig lediglich eine Zellreferenz, wurde dieselbe Fehlerursache unterstellt. Eine potenziell mögliche, aber unwahrscheinliche weitere Fehlerursache wurde nicht in Betracht gezogen. Über dieses Ergebnis hinaus könnte die Vermutung angestellt werden, dass sich die Auswirkung von Modellierungsfehlern in Mod^{2}DSS-Modellen durch Anwendung Vektor-basierter Formeln durch den Transformationsschritt gemindert werden, da in diesem Schritt der Zellbereich, auf welchen eine Formel angewandt wird, sich in der Regel verkleinert. Ein Beweis dieses Zusammenhangs ist aufgrund der beschränkten Grundgesamtheit jedoch nicht möglich.

7.4.3 Ergebnisse der Fragebogenauswertung

Die Ergebnisse der Fragenbogenauswertung stellen sich wie folgt dar: Zehn der elf Teilnehmer, die die Aufgabe mit Excel bearbeiteten, gaben an zuvor schon einmal aktiv mit

Excel Erfahrungen gesammelt und bereits relative und/oder absolute Zellbezüge in Excel genutzt zu haben. Sechs hatten bereits Erfahrung mit der Verknüpfung von Tabellenblättern, was auf Erfahrung im Lösen komplexer Probleme mittels Excel schließen lässt. Zehn der zwölf Teilnehmer, die die Aufgabe anhand des Prototyps bearbeiteten, gaben an, zuvor schon einmal mit einem multidimensionalen Modellierungswerkzeug gearbeitet zu haben. Acht von ihnen hatten aktive Erfahrung in der Modellierung mehrdimensionaler Modelle und sieben sind bereits mit Vektor-basierten Formeln in Berührung gekommen. Zusammen mit der kurzen Einführung in die benötigten Konzepte der beiden Modellierungswerkzeuge im Vorfeld der Aufgabenbearbeitung führen diese Ergebnisse zu der Annahme, dass der Erfahrungsschatz der Teilnehmer sich auf einem ähnlichen Niveau bewegte und die Vergleichbarkeit der Ergebnisse nicht durch unterschiedliche Vorkenntnisse beeinträchtigt wurde.

Die Bearbeitungsdauer der Modellierungsaufgabe, welche von den Teilnehmern persönlich erfasst wurde, lag hinsichtlich der Mod^{2}DSS-Modelle bei durchschnittlich 30,7 Minuten und in Bezug auf die mittels Excel erstellten Modelle bei 40,56 Minuten. Das subjektive Empfinden des Modellierungsaufwandes auf einer Skala von 1 bis 5 (1=sehr hoch, 3=indifferent, 5=sehr niedrig) passt zu diesem Ergebnis. Anwender des Prototyps gaben im Durchschnitt einen niedrig bis sehr niedrig empfundenen Aufwand an (4,33), während die Excel-Anwendergruppe durchschnittlich einen subjektiv erhöhten Aufwand angab (2,82). Die generell voneinander getrennten Modellierungsschritte beim Anlegen von Modellstruktur und -logik haben alle Anwender für gut bzw. wünschenswert empfunden (1,95). Es wurde des Weiteren klar, dass die subjektive Einschätzung des Aufwandes für Pflege und Erweiterung eines Modells bei Nutzung von Excel im Vergleich zum Prototyp größer wäre. 18 Teilnehmer (78,26 %) sahen einen größeren Aufwand bei der Nutzung von Excel, vier (17,39 %) erkannten keinen Unterschied und ein Teilnehmer schätzte den Aufwand bei Nutzung des Prototyps größer ein.

Das subjektive Empfinden des wahrgenommenen Aufwands sowie die reale Bearbeitungsdauer sind konsistent und legen nahe, dass die Bedienbarkeit des Prototyps gegenüber Excel Vorteile mit sich bringt. Die Auswertung des Fragebogenteils D zur Bedienbarkeit der Modellierungswerkzeuge bestätigt dies und wird in Abbildung 7-2 dargestellt:

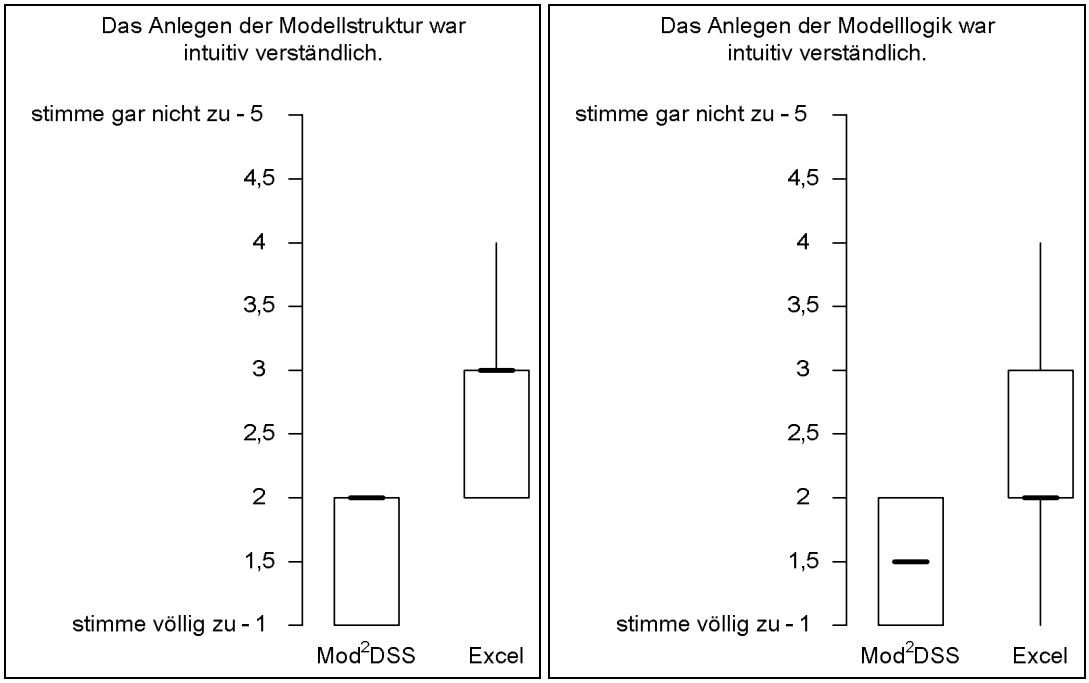


Abbildung 7-2: Empfundener Modellierungsaufwand

Auf einer Skala von 1 bis 5 (1=stimme völlig zu, 3=indifferent, 5=stimme gar nicht zu) wurde sowohl das Anlegen der Modellstruktur (durchschnittlich 1,67 gegenüber 2,82), als auch die Formeldefinition (durchschnittlich 1,5 gegenüber 2,45) von den Mod^{2}DSS-Anwendern gegenüber den Excel-Anwendern als intuitiver bewertet.

In der persönlichen Einschätzung (Teil E) finden sich ebenfalls einige Kommentare, die dies unterstützen. Ein Mod^{2}DSS-Anwender schrieb beispielsweise:

> *Die Modellierung war meines Erachtens wesentlich einfacher und schneller durchzuführen [im Vergleich zu Excel]. Außerdem glaube ich, dass aufgrund der Übersichtlichkeit auch Fehler - trotz schnellerer Modellierung - vermieden werden.*

Sicherlich ist auch bei der Beurteilung der Bedienbarkeit in Bezug auf die Aufgabenstellung zu berücksichtigen, dass diese explizit ein multidimensionales Modell vorgibt. Jedoch ist der typische Anwendungsfall für ein Excel-Modell auch nicht auf zwei Dimensionen beschränkt. Es ist typisch, dass auch in einem Excel-Modell mehrere Perioden, Kennzahlen, Produkte und Versionen betrachtet werden. Daher können die Ergebnisse durchaus für die Evaluation dieser Eigenschaft der prototypischen Applikation herangezogen werden. Eine beispielhafte Beurteilung eines Excel-Anwenders lautete:

> *Ohne eine Schulung [Vorstellung der Konzepte] hätte ich mich mit dem Modellieren sehr schwer getan. Der Zeitaufwand war unerwartet hoch und die kleinteiligen Arbeitsschritte verwirrten mich.*

7.4.4 Limitationen und Resümee

Die Fehleranfälligkeit der unterschiedlichen Modellierungsansätze in Microsoft Excel- und Mod^{2}DSS-Modellen war der zentrale Motivationsaspekt für die Erarbeitung des in den Kapiteln 4 und 5 dargestellten konzeptuellen Modellierungsansatzes für MDSS. Neben dieser zentralen Kennzahl (fehlerhafte Zellen) sowie den weiteren vorgestellten Punkten Bedienbarkeit und Verständlichkeit, hätten in der Evaluation weitere Aspekte geprüft werden können. Das Erfassen unterschiedlicher Gründe für das Auftreten von Fehlern, in Verbindung mit einer strukturierten Befragung der Nutzer während des Modellierungsprozesses (siehe Abschnitt 7.3), könnte möglicherweise aufschlussreiche Erkenntnisse über die Ausgestaltung der Sprachelemente und des Bedienungskonzepts liefern. Des Weiteren wäre die Erfassung und Evaluation sozialer Merkmale, wie z.B. Alter und Geschlecht sowie eine größere, breiter gestreute Grundgesamtheit, die unterschiedliche Ausprägungen hinsichtlich Bildung und Erfahrung im Bereich multidimensionaler Modellierung aufweist, wünschenswert.

Insgesamt führt die Interpretation der Evaluationsergebnisse zu einer positiven Beurteilung des entwickelten Konzepts. Die Ergebnisse rechtfertigen die Überlegung der konzeptuellen Modellierungsebene und können als Indizien gewertet werden, weitere Forschungsarbeiten im beschriebenen Themenfeld zu platzieren. In die Beurteilung fließt dabei auch ein, dass zentrale Funktionen, wie z.B. die Bereitstellung von Konsistenzinformationen nur zu einem kleinen Teil in der prototypischen Applikation umgesetzt sind. Stünden sämtliche Funktionen in der Form, wie sie in Kapitel 4 und 5 skizziert werden zur Verfügung, wäre die Unterstützungsleistung, die dem Anwender bereitgestellt würde, deutlich größer. Dies lässt ein ausgesprochen positiv zu beurteilendes Unterstützungspotenzial erwarten.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Eingangs der vorliegenden Arbeit wurde als Ziel die Konzeption und prototypische Implementierung einer konzeptuellen Modellierungsebene für MDSS formuliert. Der Anwender sollte in die Lage versetzt werden, ein generisches, multidimensionales Modell zu definieren und dieses teil-automatisiert in ein Ziel-Anwendungssystem zu überführen. Während des Modellierungsprozesses sollte der Anwender mit Informationen bezüglich der Konsistenz der modellierten Zusammenhänge versorgt werden, um auf diese Weise die beobachtete Fehleranfälligkeit von Spreadsheet-Modellen positiv zu beeinflussen.

Im ersten Teil der Arbeit wurde in den Kapiteln 2 und 3 die theoretische Basis für die Erstellung des Konzepts geschaffen. Aufbauend auf einer Definition und der Klassifizierung von DSS der SIGDSS wurden ein Begriffsverständnis sowie eine Klassifizierung von MDSS erarbeitet. *Modellgetrieben* impliziert in diesem Zusammenhang die Berücksichtigung eines Modellierungsprozesses zur Erstellung eines Modells, der zentralen Komponente eines MDSS. Kapitel 2.2 unterbreitete eine Beschreibung gängiger Modellbegriffe und leitete aus diesen abschließend ein dieser Arbeit zugrunde liegendes Verständnis ab. Anschließend wurde dargelegt, wie im Rahmen dieser Arbeit der Begriff *Konzept* bzw. *konzeptuelle Modellierung* zu interpretieren ist. Die Einführung einer konzeptuellen Modellierung hat im Themenbereich der Datenmodellierung dazu geführt, dass die Qualität von Datenbankapplikationen gesteigert und die Modellierung unabhängig von den Anwendungsprogrammen und damit auch proprietären Modellierungskonzepten durchgeführt werden konnten. Analog hierzu wurde mit der Einführung einer konzeptuellen Modellierungsebene für MDSS das Ziel verfolgt, auch für diese Systeme eine Anwendungssystemunabhängige Modellierung zu ermöglichen, die aufgrund der Unabhängigkeit von proprietären Werkzeugkonzepten das Abbilden realer Zusammenhänge akzentuiert. Als Resultat dieser strukturierten Vorgehensweise sollte die Modellqualität nachhaltig positiv beeinflusst werden.

Es wurde eine differenzierte Betrachtung von MDSS und insbesondere eine Unterscheidung von Systemklassen anhand von Modelltypen vorgenommen. MDSS, für welche die konzeptuelle Modellierungsebene entwickelt wurde, weisen charakteristische Eigenschaften auf. Sie berücksichtigen multidimensionale Modelle, die sich eingehend mit Relationen

und der Formulierung logischer Zusammenhänge zwischen zwei oder mehreren Modellvariablen befassen. Diese logischen Verknüpfungen bestehen i.d.R. nicht aus komplexen mathematischen Algorithmen, sondern Verknüpfungen, welche der Grundrechenarten genügen. Für die Implementierung der Modelle wird in der Regel ein Spreadsheet-Werkzeug herangezogen.

Kapitel 3 erläuterte im ersten Teilabschnitt die Fehleranfälligkeit von Spreadsheet-Modellen, da Spreadsheet-Werkzeuge vermehrt für die Implementierung modellgetriebener DSS genutzt werden. Fehlerhafte Modelle gefährden die sinnvolle Anwendbarkeit eines MDSS, beeinträchtigen dadurch die Unterstützungsleistung, die einem Entscheidungsträger zur Verfügung gestellt werden kann und müssen daher unbedingt bei der Formulierung des Konzepts berücksichtigt werden. Da Vektor-basierte Formeln, Dimensionen, Dimensionsausprägungen und Zellen zentrale Bestandteile der betrachteten MDSS darstellen, beschäftigte sich Abschnitt 3.2 im Anschluss mit den Anforderungen multidimensionaler Modelle und beschrieb deren Eigenschaften. Basierend auf diesen Ausführungen konnten Anforderungen einer konzeptuellen Modellierungsebene für MDSS in Kapitel 3.3 abgeleitet werden. Diese umfassen eine *freie Modellierung*, *generische Dimensionalität*, *Nichtprozeduralität*, *Simultanität* und eine *Teilmengen-differenzierte Kennzahlenspezifikation*. Freie Modellierung drückt aus, dass der Modellierer lediglich von seinem Modellierungszweck und den realen Begebenheiten in Form von weiteren Modellelementen abhängig ist. Generische Dimensionalität stellt den Aspekt heraus, dass jede Modelldimension dieselben strukturellen und funktionalen Möglichkeiten bereitstellt, Nichtprozeduralität bedeutet, dass der Modellierer die logisch korrekte Berechnungsreihenfolge von Modellformeln außer Acht lassen und stattdessen sach-logisch zusammenhängende Einheiten aufeinander folgend spezifizieren kann. Simultanität bezeichnet die Situation in einem Modell, in der zwei endogene Modellvariablen eine Schleife bilden, die nicht mittels prozeduraler Rechenoperationen gelöst werden kann. Teilmengen-differenzierte Kennzahlenspezifikation bedeutet, dass eine Modellvariable in unterschiedlichen Zellen eines Modells durch alternative Formelspezifikationen berechnet werden kann. Kapitel 3.4 untersuchte bestehende konzeptuelle Modellierungsansätze aus dem Themenkomplex datengetriebener Systeme. Diese ließen aufgrund ähnlicher Konzepte und Charakteristika eine möglichst hohe Adaptionswahrscheinlichkeit erhoffen. Die Beurteilung der Qualität der Ansätze wurde anhand der zuvor dargestellten Anforderungen einer konzeptuellen Modellierungsebene vorgenommen. Nach einer zusammenfassenden Darstel-

lung des State-of-the-Art wurde ersichtlich, dass keiner der bestehenden Ansätze für die Umsetzung einer konzeptuellen Modellierungsebene für MDSS geeignet ist.

Teil II der Arbeit unterbreitete einen Vorschlag für ein Konzept einer konzeptuellen Modellierungsebene für MDSS, das die im ersten Teil der Arbeit identifizierten Anforderungen der adressierten Systeme umfassend berücksichtigt. Kapitel 4 gab einen Überblick über das Konzept und stellte dabei insbesondere die Teilung in *Konfigurations-*, *Transformations-* und *Kompilierungsphase* vor. Konfigurations- und Transformationsphase bilden zusammen einen Modellierungsprozess, der das Erstellen eines konzeptuellen Modells ermöglicht. Die Teilung des Modellierungsprozesses in zwei Phasen wurde vorgenommen, um dem Anwender eine natürliche, freie und nichtprozedurale Modellierung zu ermöglichen, welche die Qualität von Modellen positiv beeinflussen kann. In der Konfigurationsphase werden sämtliche Modellelemente (Dimensionen, Dimensionsausprägungen, Gruppen und Formeln) definiert und die Überlagerung von Formeln innerhalb des Modells bewusst zugelassen. Die Transformation als zweite Phase des Modellierungsprozesses stellt die Konsistenz der modellierten Strukturen in Bezug auf deren Eindeutigkeit sicher, indem sämtliche Überschneidungen von Formeln innerhalb von Modellzellen in einem systemunterstützten Benutzerdialog beseitigt werden. Im dritten Teil des Konzepts, der Kompilierungsphase, werden die überschneidungsfreien, konzeptuellen Modelle in proprietäre Modellinstanzen übersetzt, sodass sie von Anwendungssystemen wie z.B. Microsoft Excel interpretiert und für Analysen anwendbar gemacht werden können. Auf diese Weise wird die Integration der konzeptuellen Modellierungsebene in bestehende Systemarchitekturen sichergestellt und mit konzeptuellen Modellen können mehrere unterschiedliche Modellzwecke verfolgt werden, die indirekt mit diesen Ziel-Anwendungssystemen verbunden sind.

In jedem Teil des Konzepts sollte der Anwender zudem mit Informationen versorgt werden, die sich seinem aktuellen Bedarf entsprechend anpassen. Konsistenzinformationen in Form der zwei Kriterien *Eindeutigkeit* und *Vollständigkeit* können den Anwender bei der Erstellung eines widerspruchsfreien Modells (Eindeutigkeit) unterstützen und dem unbeabsichtigten Fehlen von Formeldefinitionen (Vollständigkeit) vorbeugen. Damit diese Konsistenzinformationen automatisiert und effektiv bestimmt werden können, wurde eine Klassifizierung multidimensionaler Modellierungsfälle vorgeschlagen. Anhand der drei binär ausgestalteten Kriterien *Grad*, *Homogenität* und *Simultanität* beteiligter Dimensionen, deren Kombination zu acht Modellierungsfällen führt, können die Formeln innerhalb eines Modells klassifiziert werden. Dies geschieht in Abhängigkeit von der gesamten Mo-

delllogik, so dass die Klassifizierung einer Formel innerhalb eines Modells von den übrigen Formeln beeinflusst wird. Liegen z.B. heterogene Strukturen durch Teilmengen-differenzierte Kennzahlenspezifikationen vor (Fälle 2, 4, 6 und 8), kann automatisiert überprüft werden, ob sämtliche Intervalle durch eine Formel definiert sind und den Anwender ggf. auf inkonsistente Spezifikationen aufmerksam machen.

Kapitel 5 widmete sich der detaillierten Erläuterung der konzeptuellen Modellierungssprache, die anhand von XML umgesetzt wurde. Diese gibt den Rahmen, die Objekte und deren zulässige Kombination für die Repräsentation der konzeptuellen Modelle vor. Eine XML Schema-Definition beinhaltet die Regeln, die bei der Modellierung eingehalten werden müssen. Sie gibt vor, dass ein Modell aus Dimensionen, Dimensionsausprägungen, Formeln und Gruppen besteht, welche Modellelemente auf welche Weise miteinander verknüpft werden können, welche Modellelemente andere Elemente beinhalten dürfen und in welcher Reihenfolge dies geschehen darf. Der Modellierungsprozess, der durch den Modellierungseditor und die Transformationskomponente repräsentiert wird, implementiert diese XML Schema-Definition und stellt somit sicher, dass jede Modellinstanz konform zu dieser ist.

Der dritte und abschließende Teil der Arbeit erläuterte die prototypische Implementierung Mod^{2}DSS des Konzepts und die Evaluation im Sinne eines *Proof of Concept*. Die für einen Prototyp charakteristischen Limitationen werden eingangs des sechsten Kapitels erläutert. Daran schließt sich die Darstellung der technischen Umsetzung an, die auf Basis unterschiedlicher Technologien erfolgte. Der Modellierungseditor wurde in der Programmiersprache Java umgesetzt, für die Implementierung der Transformationsschnittstelle wurde die in Microsoft Excel integrierte Programmiersprache VBA genutzt und die Speicherung der konzeptuellen Modelle findet in einem XML-Dokument statt. Die gute Integration von VBA und Excel ermöglichte es mit vergleichsweise geringem Aufwand, die Kompilierungskomponente des dreiteiligen Konzepts beispielhaft für prinzipiell beliebige andere Ziel-Anwendungssysteme innerhalb der Transformationskomponente umzusetzen. Bei der Modelltransformation in überschneidungsfreie Strukturen wird im Hintergrund ein Spreadsheet-Modell in Microsoft Excel aufgebaut, das für weitere Entscheidungsunterstützungsfunktionen genutzt werden kann.

Nach der ausführlichen Beschreibung von Mod^{2}DSS wurde in Kapitel 7 die Evaluation beschrieben, die im Rahmen der Übungsveranstaltung zur Vorlesung *Management Support Systeme 2 – Decision Support Syste*me im Sommersemester 2011 am Fachbereich Wirt-

schaftswissenschaften der Universität Osnabrück durchgeführt wurde. Basierend auf einer erstellten Fehlermetrik zeigten die Ergebnisse, dass der Ansatz einer konzeptuellen Modellierungsebene für MDSS Potenzial für die positive Beeinflussung der Modellqualität birgt. Die durchschnittliche Anzahl fehlerhafter Zellen in Modellen war bei Mod^{2}DSS-Modellen fast neun Prozentpunkte niedriger im Vergleich zu Excel-Modellen. Sehr positiv ist ebenfalls festzuhalten, dass die zu befürchtende Auswirkung auf größere Teile eines Modells aufgrund der Vektor-basierten Formeln nicht festgestellt werden konnte. Die durchschnittliche Anzahl von Modellzellen, auf die sich eine fehlerhafte Formel auswirkt, war bei Mod^{2}DSS-Modellen sogar vier Zellen niedriger.

Insgesamt lassen die Ergebnisse den Schluss zu, dass die analoge Übertragung einer konzeptuellen Modellierungsebene aus dem Bereich der Datenmodellierung auf den Bereich MDSS ein zielführender Ansatz ist. Aus den Beschreibungen des Konzepts sowie des Prototyps ergeben sich gleichwohl weitere Fragen, die in zukünftigen Forschungsprojekten zu untersuchen sind. Der nächste Abschnitt unterbreitet einige Ideen für weitere Forschungsaktivitäten, die sich an diese Arbeit anschließen könnten.

8.2 Zukünftiger Forschungsbedarf

Zukünftiger Forschungsbedarf besteht vor allem hinsichtlich der Ausgestaltung der konzeptuellen Modellierungssprache. Während die grundlegenden Konzepte und Modellelemente im Rahmen dieser Arbeit vorgestellt wurden und feststehen, bietet die Ausgestaltung einzelner Attribute vielfältige Möglichkeiten, die spezifizierten Modelle mit Informationen anzureichern, die während der Transformations- und Kompilierungsphase genutzt werden können. Die Attribute *summiert* und *summierbar* stellen im Rahmen dieser Arbeit nur eine erste Möglichkeit dar.

Des Weiteren wäre eine adäquate Unterstützung des Anwenders bei der Modellierung der Attribute summierbar und summiert hilfreich. Beispielsweise könnten die Formeldefinitionen innerhalb eines Modells automatisiert analysiert werden. Auf diesem Weg könnte versucht werden, die Werte der beiden Attribute hinsichtlich einzelner Elemente abzuleiten oder zumindest dem Anwender nützliche Informationen zur Bestimmung der korrekten Werte bereitzustellen.

Die Verknüpfung von Modellen untereinander, im Sinne eines *multi-cube* Ansatzes, könnte in einem weiteren Schritt zu einer Erhöhung der flexiblen Anwendbarkeit der konzeptuellen MDSS-Modelle führen. Der Modellierer könnte in die Lage versetzt werden, reale Be-

gebenheiten in einzelne Teil-Modelle zu zerlegen, ohne jedoch auf die Gesamtheit der modellierten Zusammenhänge verzichten zu müssen. An dieser Stelle könnte im selben Zug eine erweiterte Nutzung eines Repository in Erwägung gezogen werden. Sofern einzelne (Teil-)Modelle bereits komponiert im Repository vorliegen würden (Ansatz einer *Modellbank* aus der klassischen DSS-Architektur, siehe Abbildung 2-2), könnte die Wiederverwendbarkeit von Modellelementen unter Umständen erhöht werden.

Für die Ableitung von Konsistenzinformationen ist die Klassifizierung multidimensionaler Modellierung (siehe Kapitel 4.3) hinsichtlich der Interpretation der unterschiedlichen Modellierungsfälle zu untersuchen. Die dargestellten Beispiele zur Auswertung der Konsistenzinformationen sind nicht vollständig und bieten unter Umständen weitere Möglichkeiten zur Ableitung nützlicher Informationen. So könnte z.B. untersucht werden, ob die Informationen über den Grad und die Homogenität einer simultanen Formel im Kontext eines Modells für den Vorschlag einer Strategie zur Auflösung dieser Bezüge genutzt werden kann. Des Weiteren müsste untersucht werden, wie sich der Informationsbedarf des Modellierers in den unterschiedlichen Teilen des Konzepts darstellt und wie dieser Informationsbedarf bestmöglich bedient werden kann. Es könnte diesbezüglich z.B. untersucht werden, ob Hinweise auf eine möglicherweise unvollständige Modellierung erst am Ende der Konfigurationsphase gegeben werden sollten, da diese Inkonsistenz durch weitere Spezifikationen aufgehoben werden könnte oder ob es zielführender wäre den Anwender sofort auf diese Zusammenhänge aufmerksam zu machen.

Hinsichtlich der Kompilierungskomponente des Konzepts stellt die Übersetzung der konzeptuellen Modelle in ausführbare Excel-Modelle nur ein Beispiel eines möglichen Anwendungssystems dar. Da die Erstellung einer konzeptuellen Modellierungsebene für MDSS sich am Themenbereich Datenmodellierung orientiert hat, wäre eine Verknüpfung von konzeptuellen MDSS-Modellen und datengetriebenen Systemen reizvoll. Forschungsarbeiten könnten sich in diesem Zusammenhang z.B. damit beschäftigen, inwiefern konzeptuelle MDSS-Modelle für die Erstellung einer OLAP-Applikation genutzt werden können. Formeln und logische Verknüpfungen könnten hierfür z.B. innerhalb von ETL-Prozessen abgebildet werden.

Anhang

I Fragebogen

Evaluation eines Modellierungswerkzeuges zur multidimensionalen Modellierung

Liebe Studierende,

in den vergangenen zwölf Monaten ist am Lehrstuhl für Management Support und Wirtschaftsinformatik im Rahmen eines Forschungsprojektes ein Prototyp eines Modellierungswerkzeuges zur multidimensionalen Modellierung entwickelt worden. Neben einem Modellierungstest, welcher die Modellierung des entwickelten Werkzeuges und eine freie Modellierung eines Excel-Sheets gegenüberstellt, soll eine kurze schriftliche Befragung durchgeführt werden. Ziel dieser Befragung ist, genauere Kenntnisse darüber zu gewinnen, ob das Werkzeug mit seinen Modellierungsfunktionalitäten eine adäquate Alternative zu bestehenden Lösungen darstellen kann.

Die Befragung wird anonym durchgeführt und alle Daten werden ausschließlich aggregiert ausgewertet (Ausnahme: Teil E) und nicht anderweitig verwendet.

A Allgemeine Angaben

A1	**Teilnehmernummer:**	_____

(für die Zuordnung von Fragebogen und Modell)

A2	**Modellierungswerkzeug**	☐	Excel	☐	multidimensionales Werkzeug
A3	**beanspruchte Zeit:**	_____			

(in Minuten, bezogen auf die Modellerstellung)

B Modellierungskenntnisse

B1	**Ich habe schon einmal mit Excel gearbeitet und Zellen miteinander verknüpft (z.B. durch eine Summe oder eine Multiplikation)**	☐	ja	☐	nein
B2	**Ich habe schon einmal in einem Excel-Modell Tabellenblätter verknüpft**	☐	ja	☐	nein
B3	**Ich habe in Excel schon einmal relative und / oder absolute Zellbezüge genutzt**	☐	ja	☐	nein
B4	**Ich habe schon einmal Modelle mit mehr als zwei Dimensionen modelliert**	☐	ja	☐	Nein
B5	**Ich habe schon einmal Modelle mit vektoriellen Formeln erstellt**	☐	ja	☐	Nein
B6	**Ich habe schon einmal mit einem anderen multidimensionalen Werkzeug gearbeitet**	☐	ja	☐	Nein

C Fragen zum Modellierungsprozess

C1	Ich habe den Modellierungsaufwand mit dem benutzten Werkzeug (siehe A2) als sehr groß empfunden.								
☐	**stimme völlig zu**	☐	**stimme eher zu**	☐	**ich bin indifferent**	☐	**stimme kaum zu**	☐	**stimme gar nicht zu**
C2	Modellstrukturen (Dimensionen & -ausprägungen) in einem separaten Schritt zur Logik (Formeln) zu modellieren, erleichtert dem Benutzer den Modellierungsprozess.								
☐	**stimme völlig zu**	☐	**stimme eher zu**	☐	**ich bin indifferent**	☐	**stimme kaum zu**	☐	**stimme gar nicht zu**
C3	Ich denke der Aufwand bestehende Modelle zu erweitern (z.B. um eine Dimension oder eine Ausprägung) ist in folgendem Werkzeug größer:								
☐	**Excel**			☐	**gleich groß**	☐	**Modellierungswerkzeug**		
C4	Ich denke der Aufwand bestehende Modelle zu warten (Änderung bestehender Modellgrößen) ist in folgendem Werkzeug größer:								
☐	**Excel**			☐	**gleich groß**	☐	**Modellierungswerkzeug**		

D Fragen zur Bedienbarkeit

D1	Die Navigation beim Anlegen der Modellstruktur (Dimensionen & -ausprägungen) war im benutzten Werkzeug (siehe A2) intuitiv verständlich.								
☐	**stimme völlig zu**	☐	**stimme eher zu**	☐	**ich bin indifferent**	☐	**stimme kaum zu**	☐	**stimme gar nicht zu**
D2	Die Navigation beim Anlegen der Modelllogik (Formeln) war im benutzten Werkzeug (siehe A2) intuitiv verständlich.								
☐	**stimme völlig zu**	☐	**stimme eher zu**	☐	**ich bin indifferent**	☐	**stimme kaum zu**	☐	**stimme gar nicht zu**

E Persönliche Einschätzung

E1	Bitte schildere kurz deine Eindrücke, die du während des Modellierungsprozesses gewonnen hast und gehe dabei vor allem auf die Vor- und Nachteile bei der Modellierung, die Bedienbarkeit und den Aufwand für die Einarbeitung ein:

II Modellierungshinweise Prototyp

Schritt 1: Ein neues Modell erstellen

- Via Datei → neues Modell gelangt man zur Aufforderung dem neuen Modell einen Namen zu geben: Bitte tragt hier „Modell + Teilnehmernummer“ ein, z.B. Modell99.

Schritt 2: Dimensionen anlegen

- Unter dem Punkt „Dimensionen und Ausprägungen“ kann man neue Dimensionen und dazugehörige Ausprägungen anlegen.
- Klickt unter dem Punkt „Dimensionen des Modells“ auf „Hinzufügen“ und gebt den Namen der Dimension ein (z.B. Zeit).

Schritt 3: Dimensionsausprägungen anlegen

- Via „Hinzufügen“ (unter Dimensionsausprägungen) können Ausprägungen angelegt werden.
- **WICHTIG:** alle Ausprägungen, die einen **summierten** Wert darstellen (z.B. Region_Summe), müssen als **„summiert“** gekennzeichnet sein.
- Alle Ausprägungen, die durch eine Summenformel ohne inhaltliche Widersprüche aufsummiert werden können, (dies ist in den meisten Fällen so, z.B. bei Mengen) müssen als **„summierbar“** gekennzeichnet sein.
- Alle Ausprägungen, für die eine Aufsummierung widersinnig wäre (z.B. macht es keinen Sinn Preise, Quoten oder Rabattsätze zu summieren), **dürfen nicht „summierbar“** gekennzeichnet sein.

Schritt 4: Formeln anlegen

- Unter dem Punkt „Formeln“ kann das Modell mit Logik gefüllt werden.

- Zuerst muss die Dimension der Ausprägung gewählt werden, für welche eine Formel definiert werden soll (oberes Drop-Down-Menü).
- Unter dem Punkt „Dimensionsausprägungen“ wählt ihr die Ausprägung und über einen Klick auf „<<“ schiebt ihr diese in das Formelfenster links.
- Achtet darauf, dass immer zuerst die Ausprägung, für die eine Formel gelten soll, gefolgt von einem „=“ modelliert wird (z.B. [Region_Summe] = [Ost] + [West]
- Ist eine Formel fertig modelliert muss sie „gespeichert“ werden.
- Dezimaltrennzeichen ist ein Punkt, also z.B. „0.1“.

Schritt 5: Modell speichern

- Zum Ende des Modellierungsprozesses speichert ihr das Modell via „Datei“ → „Modell speichern“ unter dem Namen „Modell + Teilnehmernummer“ als xml-Datei ab.

Schritt 6: Modell transformieren

Via „Datei“ → „Modell transformieren“ könnt ihr nun das Modell (xml-Datei aus Schritt 5) in ein Excel-Sheet transformieren. Wählt die Excel-Datei „Transformator“ aus, die ihr heruntergeladen habt und speichert das Modell am Ende des Prozesses (build model Dialog mit ok bestätigen) unter dem Namen „Modell + Teilnehmernummer + trans“ ab.

III Modellierungshinweise Excel

Zellen

- jedes Excel-Modell ist Zell-basiert, d.h. dass Formeln immer nur Auswirkung auf die eine Zelle haben, in welcher sie definiert sind

generelle Struktur

- Da Excel nur zwei Dimensionen anbietet, muss man sich für das Modellieren eines mehrdimensionalen Problems eine geeignete Strategie einfallen lassen; generell gibt es mehrere mögliche Lösungsansätze.
- Neben den zwei offensichtlichen Dimensionen (Ordinate und Abszisse) gibt es die Möglichkeit mehrere Tabellenblätter miteinander zu verknüpfen: z.B. addiert der Ausdruck auf Tabellenblatt 1 „=B1+Tabelle2!C1“ die Zelle „B1“ aus Tabellenblatt 1 und die Zelle „C1“ aus Tabellenblatt 2. Der Name der Tabellenblätter wird vor die Zellbezeichnung, gefolgt von einem Ausrufezeichen, gesetzt.
- Eine weitere Möglichkeit besteht in der Projektion von Dimensionen ineinander. Bei drei vorliegende Dimensionen A, B und C und dem Vorhaben Dimension A in Dimension B zu projizieren, wird jede Dimensionsausprägung der Dimension B so oft vervielfältigt und mit den Ausprägung von Dimension B kombiniert, bis jede Kombination der Dimensionsausprägungen aus Dimension A und B besteht.
- Des Weiteren können Dimensionen auch durch „Vervielfältigung des Modells“ simuliert werden. Dies bietet sich insbesondere bei einer geringen Anzahl an Ausprägungen an. Wenn Dimension C nur drei Ausprägungen enthält, wird das Modell drei Mal unter- oder nebeneinander in einem Tabellenblatt erstellt („kopiert“), für jede Ausprägung von Dimension C also ein eigenes Sub-Modell.

Zellbezüge

- In Excel gibt es die Möglichkeiten relativer und absoluter Zellbezüge, um Zellen miteinander in Beziehung zu setzen.

- Relative Bezüge passen sich beim Kopieren von Formeln an, so dass Zeilen- und Spaltenwert entsprechend der neuen Zelle geändert werden, während absolute Bezüge sich auf eine konkrete Zelle beziehen und beim Kopieren nicht angepasst werden.
- Mittels des „$“-Zeichens lässt sich diese Anpassung sowohl für den Zeilen-, als auch den Spaltenwert unterbinden. Aus dem relativen Bezug „C5“ würde der absolute Bezug „C5“ werden, bei dem weder Spalte noch Zeile der Zelle „C5“ beim Kopieren angepasst werden. Würde man beispielsweise „$C5“ schreiben, würde der Spaltenwert („C“) sich nicht ändern, der Zeilenwert jedoch angepasst werden

WENN-Funktion

- Die Funktion bildet ein IF-THEN-ELSE Konstrukt ab: An der ersten Stelle nach der öffnenden Klammer steht die Prüfung, an der zweiten der Dann-Wert, welcher eingesetzt wird, sofern die Prüfung positiv abgeschlossen wurde und an der dritten Stelle folgt der Sonst-Wert, welcher bei negativer Prüfung eingesetzt wird.
- Beispiel =WENN(A2>500;0,1;0)

Literaturverzeichnis

Alpaydin, E.: Maschinelles Lernen, München: Oldenbourg, 2008.

Alter, S.: Decision Support Systems: Current Practice and Continuing Challenges, Reading, Massachusetts: Addison-Wesley Pub., 1980.

Alter, S.: Information Systems: A Management Perspective, 2. Aufl., Menlo Park, California: Benjamin/Cummings Pub. Co., 1996.

Anthony, R. N.: Planning and Control Systems: A Framework for Analysis, Boston, Massachusetts: Studies in Management Control, Harvard Business School Press, 1965.

Balzer, W.: Die Wissenschaft und ihre Methoden: Grundsätze der Wissenschaftstheorie - ein Lehrbuch, 2. völlig überarb. Aufl., Freiburg u.a.: Verlag Karl Alber, 2009.

Balzert, H.: Lehrbuch der Software-Technik - Software Entwicklung, 2. Aufl., Heidelberg: Spektrum Akademischer Verlag, 2000.

Bauer, A. und Günzel, H.: Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung, 2. überarb. und akt. Aufl., Heidelberg: Dpunkt-Verl, 2004.

Becker, J., Rosemann, M. und Schütte, R.: Grundsätze ordnungsmäßiger Modellierung; In: Wirtschaftsinformatik, Bd. 37, 1995, S. 435–445.

Böhnlein, M. und Ulbrich-vom Ende, A.: Ein konzeptuelles Modell für die Erstellung multidimensionaler Datenstrukturen; In: Gesellschaft für Informatik, Fachgruppe 5.10 Informationssystem-Architekturen (Hrsg.), Rundbrief 8. Jahrgang, Heft 1, 2001, S. 25–58.

Bonczek, R. H., Holsapple, C. W. und Whinston, A. B.: Foundations of Decision Support Systems, New York, New York: Academic Press, 1981.

Bretzke, W.-R.: Der Problembezug von Entscheidungsmodellen, Tübingen: Mohr Siebeck, 1980.

vom Brocke, J.: Referenzmodellierung - Gestaltung und Verteilung von Konstruktionsprozessen, Berlin: Logos-Verlag, 2003.

Buhl, A.: Grundkurs VBA: Einführung in die Programmentwicklung mit Visual Basic for Applications in Excel, 3. völlig überarb. Aufl., München u.a.: Oldenbourg, 2005.

Bulos, D.: OLAP Database Design - A New Dimension; In: Database Programming & Design, 9 Bd., Nr. 6, 1996, nachgedruckt in: Chamoni, P., Gluchowski, P. (Hrsg.): Analytische Informationssysteme - Data Warehouse, On-Line Analytical Processing, Data Mining, 2. Aufl., Berlin u.a.: Springer, 1998, S. 251–261.

Bulos, D. und Forsman, S.: Getting started with ADAPT - OLAP Database Design, San Rafael, California: Symmetry Corporation - White Paper, 1998. (http://www.symcorp.com/downloads/ADAPT_white_paper.pdf - abgerufen am: 28.07.2012)

Cabibbo, L. und Torlone, R.: Querying Multidimensional Databases; In: S. Cluet und R. Hull (Hrsg.): Database Programming Languages - 6th International Workshop, DBPL-6, Estes Park, Colorado: Springer, 1998a, S. 319–335.

Cabibbo, L. und Torlone, R.: A Logical Approach to Multidimensional Databases; In: H.-J. Schek, G. Alonso, F. Saltor und I. Ramos (Hrsg.): Advances in Database Technology — EDBT'98, Valencia, Spanien: Springer, 1998b, S. 183–197.

Cabibbo, L. und Torlone, R.: From a Procedural to a Visual Query Language for OLAP; In: 10th IEEE International Conference on Scientific and Statistical Database Management (SSDBM-98), Capri, Italien: IEEE Computer Society, 1998c, S. 74–83.

Can-Weber, M. und Wendel, T.: Microsoft Excel 2010 Programmierung - Das Handbuch, Köln: O'Reilly Verlag, 2010.

Caulkins, J. P., Morrison, E. L. und Weidemann, T.: Spreadsheet Errors and Decision Making; In: Journal of Organizational and End User Computing, Bd. 19, Nr. 3, 2007, S. 1–23.

Chen, P. P.-S.: The Entity-Relationship Model - Towards a Unified View of Data; In: ACM Transactions on Database Systems, Bd. 1, Nr. 1, 1976, S. 9–36.

Codd, E. F., Codd, S. B. und Salley, C. T.: Providing OLAP to User-Analysts: An IT Mandate, Ann Arbor, Michigan: E.F. Codd & Associates, 1993.

Cragg, P. B. und King, M.: Spreadsheet Modelling Abuse: An Opportunity for OR?; In: Journal of the Operational Research Society, Bd. 44, Nr. 8, 1993, S. 743–752.

Davies, I., Green, P., Rosemann, M., Indulska, M. und Gallo, S.: How Do Practitioners Use Conceptual Modeling in Practice?; In: Data & Knowledge Engineering, Bd. 58, Nr. 3, 2006, S. 358–380.

Devlin, B.: Data Warehouse: From Architecture to Implementation, Reading, Massachusetts u.a.: Addison-Wesley, 1997.

Donovan, J. J. und Madnick, S. E.: Institutional and Ad Hoc DSS and Their Effective Use; In: ACM SIGMIS Database, Bd. 8, Nr. 3, 1977, S. 79.

Evjen, B., Sharkey, K., Thangarathinam, T., Kay, M., Vernet, A. und Ferguson, S.: Professional XML, Indianapolis, Indiana: Wiley Publishing, 2007.

Ferstl, O. K. und Sinz, E. J.: Grundlagen der Wirtschaftsinformatik, 6. überarb. und erw. Aufl., München: Oldenbourg, 2008.

Filler, A.: Elementare Lineare Algebra Linearisieren und Koordinatisieren, Heidelberg: Spektrum Akademischer Verlag, 2011.

Fischer, J., Dangelmaier, W., Natansky, L. und Suhl, L.: Bausteine der Wirtschaftsinformatik: Grundlagen und Anwendungen, 4. Aufl., Berlin: Erich Schmidt Verlag, 2008.

Flanagan, D.: Java in a Nutshell (deutsche Ausgabe für Java 1.4), 4. Aufl., Beijing u.a.: O'Reilly Verlag, 2003.

Frank, U.: Möglichkeiten und Grenzen einer objektorientierten Modellierungslehre; In: Tagungsband der STJIA'97, Erfurt, 1997, S. 96–102.

Frank, U.: Modelle als Evaluationsobjekt: Einführung und Grundlegung; In: L. J. Heinrich und I. Häntschel (Hrsg.): Evaluation und Evaluationsforschung in der Wirtschaftsinformatik, München u.a.: Oldenbourg, 2000, S. 339–352.

Frank, U.: Towards a Pluralistic Conception of Research Methods in Information Systems Research; In: ICB Research Report No. 7, Essen: Universität Duisburg-Essen, 2006.

Frank, U.: Ein Vorschlag zur Konfiguration von Forschungsmethoden in der Wirtschaftsinformatik; In: F. Lehner und S. Zelewski (Hrsg.): Wissenschaftstheoretische Fundierung und wissenschaftliche Orientierung in der Wirtschaftsinformatik, Berlin: GITO-Verlag, 2007, S. 155–184.

Gabriel, R. und Gluchowski, P.: Grafische Notationen für die semantische Modellierung multidimensionaler Datenstrukturen in Management Support Systemen; In: Wirtschaftsinformatik, Bd. 40, Nr. 6, 1998, S. 493–512.

Gabriel, R., Gluchowski, P. und Pastwa, A.: Data Warehouse & Data Mining, Herdecke; Witten: W3L-Verlag, 2009.

Gabriel, R. und Röhrs, H.-P.: Datenbanksysteme: Konzeptionelle Datenmodellierung und Datenbankarchitekturen, Berlin u.a.: Springer, 1994.

Galletta, D., Abraham, D., El Louadi, M., Lekse, W., Pollalis, Y. A. und Sampler, J. L.: An Empirical Study of Spreadsheet Error-finding Performance; In: Accounting, Management and Information Technologies, Bd. 3, Nr. 2, 1993, S. 79–95.

Galletta, D. F., Hartzel, K. S., Johnson, S. E., Joseph, J. L. und Rustagi, S.: Spreadsheet Presentation and Error Detection: An Experimental Study; In: Journal of Management Information Systems, Bd. 13, Nr. 3, 1997, S. 45–63.

Gelhoet, M.: Simulationsbasierte Unterstützung bei der Gestaltung von Entscheidungsalgorithmen: Konzeption und prototypische Realisierung am Beispiel der Ressourcenallokation im Hochschulwesen, Dissertation, Universität Osnabrück, 2010.

Glinz, M.: A Lightweight Approach to Consistency of Scenarios and Class Models, IEEE Compuer Society, 2000, S. 49–58.

Gluchowski, P., Gabriel, R. und Dittmar, C.: Management Support-Systeme und Business intelligence: Computergestützte Informationssysteme für Fach- und Führungskräfte, 2. vollst. überarb. Aufl., Berlin u.a.: Springer, 2008.

Golfarelli, M., Maio, D. und Rizzi, S.: Conceptual Design of Data Warehouses from E/R Schemes; In: Proceedings of the Thirty-First Hawaii International Conference on System Sciences, Kona, Hawaii: IEEE Computer Society, 1998a, S. 334–343.

Golfarelli, M., Maio, D. und Rizzi, S.: The Dimensional Fact Model: A Conceptual Model for Data Warehouses; In: International Journal of Cooperative Information Systems, Bd. 7, Nr. 2-3, 1998b, S. 215–247.

Gorry, G. A. und Scott Morton, M. S.: A Framework for Management Information Systems; In: Sloan Management Review, Bd. 13, Nr. 1, 1971, S. 55–70.

Green, P.: Use of Information Systems Analysis and Design (ISAD) Grammars in Combination in Upper CASE Tools – An Ontological Evaluation; In: K. Siau, Y. Wand und J. Parsons (Hrsg.): Proceedings of the 2nd CAiSE/IFIP8.1 International Workshop on the Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'97), Barcelona, 1997, S. 1–12.

Green, P. und Rosemann, M.: An Ontological Analysis of Integrated Process Modelling; In: M. Jarke und A. Oberweis (Hrsg.): Advanced Information Systems Engineering, Berlin u.a.: Springer, 1999, S. 225–240.

Hackathorn, R. D. und Keen, P. G. W.: Organizational Strategies for Personal Computing in Decision Support Systems; In: MIS Quarterly, Bd. 5, Nr. 3, 1981, S. 21–27.

Hahne, M.: Mehrdimensionale Datenmodellierung für analyseorientierte Informationssysteme; In: P. Chamoni und P. Gluchowski (Hrsg.): Analytische Informationssysteme - Business Intelligence-Technologien und -Anwendungen, 4. vollst. überarb. Aufl., Heidelberg u.a.: Springer, 2010, S. 229–258.

Harold, E. R. und Means, W. S.: XML in a Nutshell: A Desktop Quick Reference., Sebastopol, California: O'Reilly Verlag, 2004.

Heinrich, L. J., Heinzl, A. und Roithmayr, F.: Wirtschaftsinformatik-Lexikon, 7. vollst. überarb. und erw. Aufl., München u.a.: Oldenbourg, 2004.

Herden, O.: Eine Entwurfsmethodik für Data Warehouses, Dissertation, Universität Oldenburg, 2001.

Herden, O. und Harren, A.: Die ODAWA-Methodik für den Entwurf von Data-Warehouse-Datenbanken; In: Informatik Forschung und Entwicklung, Bd. 19, Nr. 2, 2004, S. 87–96.

Holsapple, C. W. und Whinston, A. B.: Decision Support Systems - A Knowledge-based Approach, St. Paul, Minnesota: West Publishing Company, 1996.

Holten, R.: Entwicklung von Führungsinformationssystemen: Ein methodenorientierter Ansatz, Wiesbaden: Dt. Univ.-Verl.; Gabler, 1999.

Holthuis, J.: Der Aufbau von Data Warehouse-Systemen: Konzeption - Datenmodellierung - Vorgehen, Wiesbaden: Dt. Univ.-Verl.; Gabler, 1999.

Horak, S., Evers, S. und Lauber, M.: Business Intelligence vs. Self Service Business Intelligence: Was leisten Tools wie PowerPivot 2010 und XLCubed?, München: GRIN Verlag, 2010.

Janvrin, D. und Morrison, J.: Factors influencing risks and outcomes in end-user development, IEEE Computer Society, 1996, S. 346–355.

Keen, P. G. W. und Scott Morton, M. S.: Decision Support Systems: An Organizational Perspective, Reading, Massachusetts: Addison-Wesley, 1978.

Kemper, H.-G., Mehanna, W. und Unger, C.: Business Intelligence - Grundlagen und praktische Anwendungen, 2. erg. Aufl., Wiesbaden: Friedr. Vieweg & Sohn Verlag, 2006.

Kimball, R. und Ross, M.: The Data Warehouse Toolkit - The Complete Guide to Dimensional Modeling, 2. Aufl., New York: Wiley Publishing, 2002.

Kosiol, E.: Modellanalyse als Grundlage unternehmerischer Entscheidungen; In: Zeitschrift für betriebswirtschaftliche Forschung, 1961, S. 318–334.

Kreutzer, W.: Grundkonzepte und Werkzeugsysteme objektorientierter Systementwicklung - Stand der Forschung und Anwendung; In: Wirtschaftsinformatik, Bd. 32, Nr. 3, 1990, S. 212–227.

Krüger, G.: Handbuch der Java-Programmierung: Standard-Edition Version 6, 5. Aufl., München u.a.: Addison-Wesley, 2009.

Law, A. M. und Kelton, W. D.: Simulation Modeling and Analysis, New York: McGraw-Hill, 2000.

Lehner, F., Maier, R. und Hildebrand, K.: Wirtschaftsinformatik: Theoretische Grundlagen, München u.a.: Hanser, 1995.

Lenze, B.: Basiswissen Lineare Algebra, Herdecke u.a.: W3L-Verlag, 2006.

March, S. T. und Hevner, A. R.: Integrated Decision Support Systems: A data Warehousing Perspective; In: Decision Support Systems, Bd. 43, Nr. 3, 2007, S. 1031–1043.

McGill, T. J. und Klobas, J. E.: The Role of Spreadsheet Knowledge in User-developed Application Success; In: Decision Support Systems, Bd. 39, Nr. 3, 2005, S. 355–369.

Mertens, P.: Die Wirtschaftsinformatik auf dem Weg zur Unternehmensspitze – alte und neue Herausforderungen und Lösungsansätze; In: W. Uhr, W. Esswein und E. Schoop (Hrsg.): Wirtschaftsinformatik 2003: Medien - Märkte - Mobilität, Band 1, Heidelberg: Physica, 2003, S. 49–74.

de Molière, F.: Prinzipien des Modellentwurfs - Eine modelltheoretische und gestaltungsorientierte Betrachtung, Dissertation, Technische Hochschule Darmstadt, 1984.

Naumann, J. D. und Jenkins, A. M.: Prototyping: The New Paradigm for Systems Development; In: MIS Quarterly, Bd. 6, Nr. 3, 1982, S. 29–44.

Nguyen, T. B., Tjoa, A. M. und Wagner, R. R.: An Object Oriented Multidimensional Data Model for OLAP; In: H. Lu und A. Zhou (Hrsg.): Web-Age Information Management, Berlin u.a.: Springer, 2000, S. 69–82.

Oehler, K.: Corporate Performance Management mit Business Intelligence Werkzeugen, München u.a.: Hanser, 2006.

Olson, J. R. und Nilsen, E.: Analysis of the Cognition Involved in Spreadsheet Software Interaction; In: Human-Computer Interaction, Bd. 3, Nr. 4, 1987, S. 309–349.

Ortlieb, C. P.: Heinrich Hertz und das Konzept des mathematischen Modells; In: Proceedings of the Symposium for History of Science - Heinrich Hertz (1857–1894) and the Development of Communication, Hamburg, 2008, S. 53–70.

Panko, R. R.: What We Know About Spreadsheet Errors, University of Hawai'i - College of Business Administration, 2008.
(http://panko.shidler.hawaii.edu/ssr/Mypapers/whatknow.htm - abgerufen am: 28.07.2012)

Panko, R. R. und Halverson, R. P. J.: Spreadsheets on trial: a survey of research on spreadsheet risks, IEEE Computer Society, 1996, S. 326–335.

Panko, R. R. und Sprague, R. H. J.: Hitting the Wall: Errors in Developing and Code Inspecting a "Simple" Spreadsheet Model; In: Decision Support Systems, Bd. 22, Nr. 4, 1998, S. 337–353.

Powell, S. G., Baker, K. R. und Lawson, B.: A Critical Review of the Literature on Spreadsheet Errors; In: Decision Support Systems, Bd. 46, Nr. 1, 2008, S. 128–138.

Powell, S. G., Baker, K. R. und Lawson, B.: Impact of Errors in Operational Spreadsheets; In: Decision Support Systems, Bd. 47, Nr. 2, 2009, S. 126–132.

Power, D. J.: Decision Support Systems: Concepts and Resources for Managers, Westport, Connecticut: Quorum Books, 2002.

Power, D. J. und Sharda, R.: Model-driven Decision Support Systems: Concepts and Research Directions; In: Decision Support Systems, Bd. 43, Nr. 3, 2007, S. 1044–1061.

Rajalingham, K.: A Revised Classification of Spreadsheet Errors, Greenwich, UK, 2005, S. 185–199.

Rajalingham, K., Knight, B. und Chadwick, D.: Classification of Spreadsheet Errors, Greenwich, UK, 2000, S. 23–34.

Rauh, O. und Stickel, E.: Konzeptuelle Datenmodellierung, Stuttgart: Teubner, 1997.

Repenning, A. und Sumner, T.: Agentsheets: A Medium for Creating Domain-oriented Visual Languages; In: Computer, Bd. 28, Nr. 3, 1995, S. 17–25.

Rieger, B.: Wissensbasierte Erweiterungen von Planungssprachen; In: D. Ehrenberg, H. Krallmann und B. Rieger (Hrsg.): Wissensbasierte Systeme in der Betriebswirtschaft: Grundlagen, Entwicklung, Anwendungen, Berlin: Erich Schmidt Verlag, 1990, S. 251–266.

Rieger, B.: Der rechnerunterstützte Arbeitsplatz für Führungskräfte, Habilitationsschrift, Technische Universität Berlin, 1994.

Riel, A.: Object-Oriented Design Heuristics, Boston, Massachusetts: Addison-Wesley, 1996.

Rieper, B.: Betriebswirtschaftliche Entscheidungsmodelle: Grundlagen, Herne u.a.: Verlag Neue Wirtschafts-Briefe, 1992.

Saaty, T. L.: The Analytic Hierarchy Process: Planning, Setting Priorities, Resource Allocation, New York u.a.: McGraw-Hill, 1980.

Sapia, C., Blaschka, M., Höfling, G. und Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm; In: Y. Kambayashi, D.-L. Lee, E. Lim, M. Mohania und Y. Masunaga (Hrsg.): Advances in Database Technologies, Berlin u.a.: Springer, 1999, S. 105–116.

Saylor, M., Bedell, J. und Rodenberger, T.: Optimizing VLDB Systems for Relational OLAP; In: Data Based Advisor, Bd. 15, Nr. 3, 1997, S. 39–43.

Schelp, J.: Konzeptionelle Modellierung mehrdimensionaler Datenstrukturen analyseorientierter Informationssysteme, Dissertation, Universität St. Gallen, 2000.

Schultewolter, C.: Konzeptuelle Modellierung für modellgetriebene Decision Support Systeme; In: H. Baars und B. Rieger (Hrsg.): Forschungskolloquium Business Intelligence 2009 der GI Fachgruppe 5.8 - Perspektiven der betrieblichen Management- und Entscheidungsunterstützung, Dortmund: CEUR Workshop Proceedings, Vol. 542, 2009, S. 15–31.

Schultewolter, C.: Towards a Framework for the Generic Specification of Model-driven Decision Support Systems: Classification Criteria of Model Relationships; In: R. H. J. Sprague (Hrsg.): Proceedings of the 43rd Hawaii International Conference on System Sciences 2010 (HICSS-43), Los Alamitos, Calofornia: IEEE Computer Society, 2010a, S. 1–10.

Schultewolter, C.: Klassifizierung von Modellierungsfällen in Modellgetriebenen Decision Support Systemen; In: M. Schumann, L. M. Kolbe, M. H. Breitner und A. Frerichs (Hrsg.): Multikonferenz Wirtschaftsinformatik 2010, Göttingen: Universitätsverlag Göttingen, 2010b, S. 1113–1124.

Schütte, R.: Grundsätze ordnungsmässiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle, Wiesbaden: Gabler, 1998.

Schwarzer, B. und Krcmar, H.: Wirtschaftsinformatik: Grundlagen betrieblicher Informationssysteme, 4. überarb. Aufl., Stuttgart: Schäffer-Poeschel, 2010.

Scott Morton, M. S.: State of the Art of Research in Management Support Systems, Cambridge, Massachusetts: Center for Information Systems Research (CISR)

107, Sloan School of Management, Massachusetts Institute of Technology, 1983.

Sharda, R., Barr, S. H. und McDonnell, J. C.: Decision Support System Effectiveness: A Review and an Empirical Test; In: Management Science, Bd. 34, Nr. 2, 1988, S. 139–159.

Simon, H. A.: Administrative Behavior: A Study of Decision-making Process in Administrative Organization, New York: Macmillan, 1947.

Simon, H. A.: The New Science of Management Decision, überarb. Aufl., Englewood Cliffs, New Jersey: Prentice-Hall, 1977.

Sinz, E. J.: Modellierung; In: P. Mertens, A. Back, J. Becker, W. König, H. Krallmann, B. Rieger, A.-W. Scheer, D. Seibt, P. Stahlknecht, H. Strunz, R. Thome und H. Wedekind (Hrsg.): Lexikon der Wirtschaftsinformatik, 3. Aufl., Berlin u.a.: Springer, 1997.

Sprague, R. H. J. und Carlson, E. D.: Building Effective Decision Support Systems, Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

Sprague, R. H. J. und Watson, H. J.: Decision Support Systems: Putting Theory into Practice, 3. Aufl., London: Prentice Hall, 1993.

Sprague, R. H. J. und Watson, H. J.: Decision Support for Management, Upper Saddle River, New Jersey: Prentice-Hall, 1996.

Stachowiak, H.: Allgemeine Modelltheorie, Wien u.a.: Springer, 1973.

Staehle, W. H.: Kennzahlen und Kennzahlensysteme: Ein Beitrag zur modernen Organisationstheorie, Dissertation, München 1967.

Staehle, W. H.: Kennzahlen und Kennzahlensysteme als Mittel der Organisation und Führung von Unternehmen, Wiesbaden: Gabler, 1969.

Stegmüller, W.: Probleme und Resultate der Wissenschaftstheorie und analytischen Philosophie, Berlin u.a.: Springer, 1974.

Thomas, O.: Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation; In: Veröffentlichungen des Instituts für Wirtschaftsinformatik im Deutschen Forschungszentrum für Künstliche Intelligenz, Nr. 184, Saarbrücken: Universität des Saarlandes, 2005.

Totok, A.: Modellierung von OLAP- und Data-Warehouse-Systemen, Wiesbaden: Dt. Univ.-Verl.; Gabler, 2000.

Totok, A. und Jaworski, R.: Modellierung von multidimensionalen Datenstrukturen mit ADAPT - Ein Fallbeispiel, Institut für Wirtschaftswissenschaften der Technischen Universität Braunschweig, 1998.

Turban, E., Sharda, R. und Delen, D.: Decision Support and Business Intelligence Systems, 9. internat. Aufl., Upper Saddle River, New Jersey: Prentice Hall, 2011a.

Turban, E., Sharda, R., Delen, D. und King, D.: Business Intelligence: A Managerial Approach, Boston, Massachusetts: Prentice Hall, 2011b.

Ullenboom, C.: Java ist auch eine Insel, 8. Aufl., Bonn: Galileo Press, 2009.

Vetter, M.: Aufbau betrieblicher Informationssysteme mittels objektorientierter, konzeptioneller Datenmodellierung, Stuttgart: Teubner, 1991.

Vogt, C.: Dokument-Austauschformate und -Auszeichnungssprachen; In: P. Rechenberg und G. Pomberger (Hrsg.): Informatik-Handbuch, 4. akt. und erw. Aufl., München u.a.: Hanser, 2006.

Wand, Y. und Weber, R.: Information Systems and Conceptual Modeling: A Research Agenda; In: Information Systems Research, Bd. 13, Nr. 4, 2002, S. 363–376.

Weber, K.: Mehrkriterielle Entscheidungen, München u.a.: Oldenbourg, 1993.

Weber, R. und Zhang, Y.: An Analytical Evaluation of NIAM'S Grammar for Conceptual Schema Diagrams; In: Information Systems Journal, Bd. 6, Nr. 2, 1996, S. 147–170.

Wilde, T. und Hess, T.: Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung; In: T. Hess (Hrsg.): Institut für Wirtschaftsinformatik und Neue Medien, Arbeitsbericht Nr.2/2006, München: Ludwig-Maximilians-Universität München, 2006.

Wilde, T. und Hess, T.: Forschungsmethoden der Wirtschaftsinformatik; In: Wirtschaftsinformatik, Bd. 49, Nr. 4, 2007, S. 280–287.

Winter, R.: Analytische Informationssysteme aus Managementsicht: Unternehmensweite Informationslogistik und analytische Prozessunterstützung; In: P. Chamoni und P. Gluchowski (Hrsg.): Analytische Informationssysteme: Business Intelligence-Technologien und -Anwendungen, 4. vollst. überarb. Aufl., Heidelberg u.a.: Springer, 2010.

Witte, T.: Simulationstheorie und ihre Anwendung auf betriebliche Systeme, Wiesbaden: Betriebswirtschaftlicher Verlag Gabler, 1973.

Witte, T., Deppe, J. F. und Born, A.: Lineare Programmierung: eine Einführung für Wirtschaftswissenschaftler in 12 Lehreinheiten, Wiesbaden: Gabler, 1975.

Woll, A.: Wirtschaftslexikon, 9. völlig überarb. und erw. Aufl., München u.a.: Oldenbourg, 2000.

Zwicker, E., Jahnke, B. und Preßmar, D. B.: Simultane und rekursive Gleichungssysteme in der Kosten- und Leistungsrechnung; In: IT-gestützte betriebswirtschaftliche Entscheidungsprozesse, Wiesbaden: Gabler, 2001.