2009

EIN ALLGEMEINER ANSATZ ZUR ABLEITUNG VON ABHÄNGIGKEITSANALYSEN AUF UNTERNEHMENSARCHITEKTURMODELLEN

Stephan Kurpjuweit
Universität St. Gallen

Stephan Aier
Universität St. Gallen

EIN ALLGEMEINER ANSATZ ZUR ABLEITUNG VON ABHÄNGIGKEITSANALYSEN AUF UNTERNEHMENSARCHITEKTURMODELLEN

Stephan Kurpjuweit, Stephan Aier[1]

Kurzfassung
Der Nutzen von Unternehmensarchitekturmodellen liegt unter anderem in der Bereitstellung bedarfsgerechter Abhängigkeitsanalysen. Die praktische Herausforderung besteht darin, beliebige Analysen auf Instanzen von vorab nicht vollständig bekannten, unternehmensspezifischen Architekturmetamodellen durchführen zu können. Der Beitrag entwirft darum eine allgemeine und formale Lösung dieses Problems, welche sich für die Implementierung in Unternehmensarchitekturmanagement-Werkzeugen eignet.

1. Einleitung

Nach ANSI/IEEE Std 1471-2000 ist Architektur definiert als ”the fundamental organization of a system, embodied in its components, their relationships to each other and the environment” [8]. Die Unternehmensarchitektur ist die fundamentale Strukturierung einer Organisation (Unternehmen, öffentliche Institution etc.) und bildet die Gestaltungsobjekte der Organisation und deren Beziehungen in Modellen ab. Die Bandbreite der Gestaltungsobjekte reicht dabei von der Strategie, über die Aufbau- und Ablauforganisation bis hin zu Elementen der Daten- und Softwarearchitektur sowie der technischen Infrastruktur [3, 13]. Aufbauend auf der Dokumentation der Strukturen einer Organisation können Unternehmensarchitekturmodelle nach [10] in sehr unterschiedlichen Szenarien (z. B. Produkteinführung, IT-Konsolidierung, Mergers&Acquisitions) genutzt werden, in denen die wechselseitigen Abhängigkeiten der Gestaltungsobjekte von Bedeutung sind. Typische Fragen in diesem Zusammenhang sind: „Welche Produkte werden durch welche Prozesse unter Zuhilfenahme welcher Applikationen erstellt?“, „Welche Dienstleistungen sind betroffen bei Ausfall eines Serverclusters“ oder „Wie ist der Beitrag einzelner Geschäftseinheiten oder Geschäftsprozesse zu den Unternehmenszielen?“. Zur Beantwortung dieser Fragen werden die betreffenden Gestaltungsobjekte gefiltert, in Beziehung gesetzt und übersichtlich z. B. als Matrix dargestellt [2].

Die praktische Herausforderung besteht dann darin, genau die Abhängigkeitsanalysen zur Verfügung zu stellen, die für die Stakeholder den größten Nutzen erzeugen. Die Erfahrung aus Praxisprojekten zeigt jedoch drei Dinge: (1) Statt starrer Standardauswertungen wünschen sich die Stakeholder flexible „Navigationsfunktionalitäten“ zur interaktiven Erschließung der strukturellen Zusam-

[1] Universität St. Gallen, Müller-Friedberg-Strasse 8, CH-9000 St. Gallen

menhänge in der Unternehmensarchitektur. (2) Nicht für jede Situation respektive jede Organisation kann das gleiche Metamodell verwendet werden. (3) Existierende Ansätze und Werkzeuge zur Modellierung und Analyse der Unternehmensarchitektur basieren auf mehr oder weniger starren Metamodellstrukturen. Komplexere Abhängigkeitsanalysen sind auf dieser Struktur meist „fest verdrahtet“ und können nur sehr begrenzt und mit hohem Aufwand angepasst werden. Dies führt zu der Anforderung an zukünftige Werkzeuge, flexible Abhängigkeitsanalysen generisch auf Basis eines flexiblen Metamodells zu unterstützen. Dabei sind eine Reihe von Herausforderungen, wie inkonsistente Semantiken von Hierarchien unterschiedlicher Gestaltungsobjekte, verschiedene Formen der Abhängigkeiten oder transitive Abhängigkeiten über eine Vielzahl von Gestaltungsobjekten hinweg, zu berücksichtigen.

Das Ziel dieses Beitrags ist es, basierend auf bestehenden relationalen Ansätzen eine allgemeine formale Lösung für die Ableitung von Abhängigkeitsanalysen zu entwickeln. Die Formalisierung unserer Lösung ist dabei notwendig, um sie eindeutig zu spezifizieren und in entsprechenden Werkzeugen implementieren zu können. Der Ansatz geht dabei von möglichst wenigen Grundannahmen aus, um die Anwendbarkeit mit möglichst vielen Ansätzen zur Modellierung der Unternehmensarchitektur sicherzustellen (Abschnitt 2). Wir beschränken uns in der Darstellung auf zweidimensionale Abhängigkeitsanalysen, welche die Abhängigkeiten zwischen je zwei Typen von Gestaltungsobjekten analysieren und die Basis darstellen für komplexere Abhängigkeitsanalysen (Abschnitt 3). Dabei berücksichtigen wir Anforderungen, die sich in Praxisfallstudien [1] als relevant herausgestellt haben: reflexive Beziehungsklassen (Abschnitt 4), hierarchische Verfeinerung (Abschnitt 5) sowie die aggregierte Betrachtung von Abhängigkeiten (Abschnitt 6). Die Effektivität der Lösung soll durch Praxisbeispiele belegt werden.

2. Metamodell

Dem Ansatz liegt folgende Annahme zugrunde: Unternehmensarchitekturen werden als Modelle in einer Modellierungssprache beschrieben. Ein Metamodell ist ein Modell einer Modellierungssprache [6, 9, 12]. Es spezifiziert die verfügbaren Arten von Gestaltungsobjekten, die Beziehungen zwischen den Gestaltungsobjekten sowie Konsistenzbedingungen für die Verwendung von Gestaltungsobjekten und Beziehungen [11]. In Anlehnung an [5] definieren wir ein *Metamodell (M)* als ein 3-Tupel: $M = (C, T, R)$, wobei gilt:

- C ist eine Menge von *Gestaltungsklassen* (kurz: *Klassen*).
- T ist eine Menge von *Beziehungstypen* (kurz: *Typen*). Für jeden Typ $t \in T$ existiert ein Umkehrtyp $t^{-1} \in T$ mit $(t^{-1})^{-1} = t$.
- $R \subseteq C \times C \times T$ ist eine Menge von *Beziehungsklassen* zwischen Gestaltungsklassen. Ein Element $(c_1, c_2, t) \in R$ ist dabei eine *Beziehungsklasse* des Typs t zwischen den Klassen c_1 und c_2.

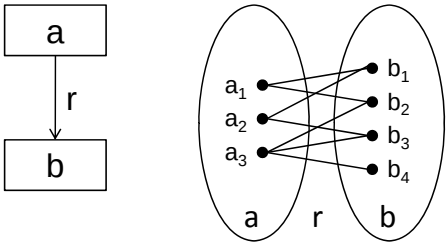
Die Granularität der Beziehungstypen ist beliebig: In den Extremfällen existiert in dem Metamodell nur ein Typ für alle Beziehungsklassen oder ein eigener Typ für jede Beziehungsklasse. [5] enthält ein Beispiel mit einem relativ feingranularen System an Beziehungstypen. In all unseren Praxisfallstudien waren die in Tabelle 1 dargestellten drei Beziehungstypen Assoziation, Aggregation und Generalisierung mit ihren jeweiligen Umkehrtypen ausreichend, wobei Assoziationen aussagekräftige Namen erhielten.

Tabelle 1: Beziehungstypen

	$t \in T$	$t^{-1} \in T$
Assoziation	Assoziation	Assoziation
Aggregation	Teil-Ganzes-Beziehung	Ganzes-Teil-Beziehung
	Ganzes-Teil-Beziehung	Teil-Ganzes-Beziehung
Generalisierung	Generalisierung	Spezialisierung
	Spezialisierung	Generalisierung

3. Zweidimensionale Abhängigkeitsanalysen (2dA)

Zahlreiche Strukturfragen zur Analyse der Unternehmensarchitektur sind vom Typ „Wie ist der Zusammenhang zwischen den Instanzen der Gestaltungsklassen a und b?“ (z. B. „Durch welche Applikationen werden die einzelnen Geschäftsprozesse unterstützt?“). Im einfachsten Fall sind die beiden Klassen $a, b \in C$ im Metamodell direkt durch eine Beziehungsklasse r verbunden. In diesem Fall kann eine 2dA unmittelbar aus den Instanzen der Beziehungsklasse r abgeleitet und beispielsweise als Matrix dargestellt werden (siehe Abbildung 1).



(a) Metamodell (b) Modell

	b_1	b_2	b_3	b_4
a_1	X	X		
a_2	X		X	
a_3		X	X	X

(c) zweidimensionale Abhängigkeitsanalyse (als Matrix)

Abbildung 1: Beispiel einer einfachen zweidimensionalen Abhängigkeitsanalyse

Sind zwei Klassen $c_1, c_2 \in C$ nicht direkt durch eine Beziehungsklasse verbunden, muss eine transitive Beziehungsklasse aus den Beziehungsklassen des Metamodells abgeleitet werden. [5] hat hierzu Vorarbeiten geleistet, die in diesem Abschnitt aufgezeigt und erweitert werden. Wir definieren zunächst einen Kompositionsoperator für Beziehungstypen $\otimes : T^* \times T^* \rightarrow T^*$. Nachdem die Komposition zweier Beziehungstypen $t_1, t_2 \in T$ einen Beziehungstyp $t_3 \notin T$ ergeben kann, der nicht in dem Metamodell verwendet wird, definiert T^* eine Erweiterung von T um alle Beziehungstypen, die bei der Komposition von Beziehungstypen aus T^* resultieren können. Tabelle 2 spezifiziert einen einfachen Kompositionsoperator für das Typsystem aus Tabelle 1, welches wir um den Typ „undefiniert“ ergänzen. Es besteht die Möglichkeit, den Kompositionsoperator zu detaillieren (z. B. indem die Kombination aus Aggregation und Generalisierung zu einer Beziehung des Typs „Verfeinerung“ komponiert wird.).

Tabelle 2: Kompositionsoperator für Beziehungstypen

t_1	t_2	$t_1 \otimes t_2$
Assoziation	*	Assoziation
*	Assoziation	Assoziation
Teil-Ganzes-Beziehung	Teil-Ganzes-Beziehung	Teil-Ganzes-Beziehung
Ganzes-Teil-Beziehung	Ganzes-Teil-Beziehung	Ganzes-Teil-Beziehung
Generalisierung	Generalisierung	Generalisierung
Spezialisierung	Spezialisierung	Spezialisierung
sonst		undefiniert

Legende: „*“ steht als Platzhalter für alle Typen, „sonst“ steht als Platzhalter für alle nicht aufgeführten Paarungen

Wir definieren rekursiv die Mengen der 2-Schritt Beziehungsklassen R^2, der n-Schritt Beziehungsklassen R^n und die transitive Hülle R^* der Menge der Beziehungsklassen[2]:

1a) $\forall c_1, c_2, c_3 \in C \forall t_1, t_2 \in T : (c_1 \neq c_2) \wedge (c_2 \neq c_3) \wedge (c_1, c_2, t_1) \in R \wedge (c_2, c_3, t_2) \in R$
$\Rightarrow (c_1, c_3, t_1 \otimes t_2) \in R^2$

1b) $\forall c_1, c_2, c_3 \in C \forall t_1, t_2 \in T : (c_1 \neq c_2) \wedge (c_2 \neq c_3) \wedge (c_1, c_2, t_1) \in R \wedge (c_3, c_2, t_2) \in R$
$\Rightarrow (c_1, c_3, t_1 \otimes t_2^{-1}) \in R^2$

2a) $\forall c_1, c_{n-1}, c_n \in C \forall t_1, t_2 \in T^* : (c_1 \neq c_{n-1}) \wedge (c_{n-1} \neq c_n) \wedge (c_1, c_{n-1}, t_1) \in R^{n-1} \wedge (c_{n-1}, c_n, t_2) \in R$
$\Rightarrow (c_1, c_n, t_1 \otimes t_2) \in R^n$

2b) $\forall c_1, c_{n-1}, c_n \in C \forall t_1, t_2 \in T^* : (c_1 \neq c_{n-1}) \wedge (c_{n-1} \neq c_n) \wedge (c_1, c_{n-1}, t_1) \in R^{n-1} \wedge (c_n, c_{n-1}, t_2) \in R$
$\Rightarrow (c_1, c_n, t_1 \otimes t_2^{-1}) \in R^n$

3) $R^* = \bigcup_{n=1}^{\infty} R^n$

Damit die Definitionen von R^2, R^n und R^* eindeutig sind, muss der Kompositionsoperator $\otimes$ assoziativ sein, d. h. $\forall t_1, t_2, t_3 \in T^* : (t_1 \otimes t_2) \times t_3 = t_1 \otimes (t_2 \times t_3)$ (B1). Abbildung 2 veranschaulicht die Ableitung von 2-Schritt Beziehungsklassen und die Notwendigkeit der Formel (1b)[3]: Da die Beziehungsklassen gerichtet sind, wird in dem dargestellten Beispiel durch Formel (1a) keine transitive Beziehungsklasse zwischen c_1 und c_3 abgeleitet, obwohl eine 2dA zwischen diesen Klassen möglich ist. Formel (2b) begründet sich mit dem gleichen Argument. Für den Kompositionsoperator ergibt sich als weitere Bedingung: $\forall t_1, t_2 \in T^* : (t_2 \otimes t_1^{-1}) = (t_1 \otimes t_2^{-1})^{-1}$ (B2). Die Bedingungen (B1) und (B2) können für das in den Tabellen 1 und 2 spezifizierte Typsystem einfach durch Wertetabellen nachgewiesen werden.

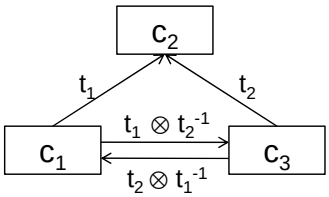


Abbildung 2: Ableitung von 2-Schritt Beziehungsklassen

Auf Basis des Kompositionsoperators für Beziehungstypen kann nun der Kompositionsoperator für Beziehungsklassen: $\otimes_R : R^* \times R^* \rightarrow R^*$ definiert werden. Der Projektionsoperator π_i ergibt dabei das i-te Element jedes Tupels in R^*:

4) $r_1 \otimes_R r_2 = \begin{cases} (\pi_1(r_1), \pi_2(r_2), \pi_3(r_1) \otimes \pi_3(r_2)), \text{falls } \pi_1(r_2) = \pi_2(r_1) \wedge \pi_1(r_1) \neq \pi_2(r_1) \wedge \pi_1(r_2) \neq \pi_2(r_2) \\ (\pi_1(r_1), \pi_1(r_2), \pi_3(r_1) \otimes \pi_3(r_2)^{-1}), \text{falls } \pi_2(r_2) = \pi_2(r_1) \wedge \pi_1(r_1) \neq \pi_2(r_1) \wedge \pi_1(r_2) \neq \pi_2(r_2) \\ \text{undefiniert sonst} \end{cases}$

---

[2] Als Ergänzung zu [5] schließen wir reflexive Kanten der Form $(c, c, t) \in R$ für ein $c \in C, t \in T^*$ (d. h. Beziehungsklassen einer Gestaltungsklasse zu sich selbst) aufgrund ihrer speziellen Semantik aus den Definitionen von R^2, R^n und R^* aus. Sie werden in Abschnitt X gesondert behandelt.

[3] Die Formeln (1b) und (2b) sind eine weitere Ergänzung zu [5]. Durch diese Formeln begründet sich auch das Konzept der *Umkehrtypen* in der Definition des Metamodells.

Die bisherige Betrachtung fand auf der Ebene des Metamodells statt. Zur Ableitung einer Abhängigkeitsanalyse muss jedoch die Modellebene betrachtet werden. Wiederum in Anlehnung an [5] definieren wir: Ein *Modell (A)* ist definiert als ein 4-Tupel $A = (E, T^*, Q, F_c)$ mit:

- E ist eine Menge von *Gestaltungsobjekten* (Instanzen von Gestaltungsklassen, kurz: *Objekte*).
- T^* ist eine Menge von *Beziehungstypen*. (Dabei handelt es sich um die Beziehungstypen aus dem Metamodell.)
- $Q \subseteq E \times E \times T^*$ Ist eine Menge von *Beziehungen* zwischen den Gestaltungsobjekten.
- $F_c : E \to C$ ist eine Funktion, die jedes Gestaltungsobjekt einer Gestaltungsklasse zuordnet.

$E_c = \{e \in E \mid F_c(e) = c\}$ ist die Menge aller Objekte der Klasse $c \in C$. Ein Modell $A = (E, T^*, Q, F_c)$ *entspricht* einem Metamodell $M = (C, T, R)$ genau dann, wenn $\forall t \in T^* \forall e_1, e_2 \in E : (e_1, e_2, t) \in Q \Rightarrow (F_c(e_1), F_c(e_2), t) \in R$. Analog zum Metamodell lässt sich die transitive Hülle Q^* der Menge der Beziehungen rekursiv über 2-Schritt Beziehungen Q^2 und n-Schritt Beziehungen Q^n definieren. Aus Platzgründen wird nur die Formel für Q^2 angegeben, Q^n ergibt sich analog zu den Formel 2a und b.

5a) $\forall e_1, e_2, e_3 \in E \forall t_1, t_2 \in T^* : (e_1 \neq e_2) \wedge (e_2 \neq e_3) \wedge (e_1, e_2, t_1) \in Q \wedge (e_2, e_3, t_2) \in Q$
$\Rightarrow (e_1, e_3, t_1 \otimes t_2) \in Q^2$

5b) $\forall e_1, e_2, e_3 \in E \forall t_1, t_2 \in T^* : (e_1 \neq e_2) \wedge (e_2 \neq e_3) \wedge (e_1, e_2, t_1) \in Q \wedge (e_3, e_2, t_2) \in Q$
$\Rightarrow (e_1, e_3, t_1 \otimes t_2^{-1}) \in Q^2$

6) $Q^* = \bigcup_{n=1}^{\infty} Q^n$

Zwei Objekte $e_1, e_2 \in E$ stehen dann miteinander über eine Beziehung des Typs $t \in T$ in Beziehung, wenn gilt: $(e_1, e_2, t) \in Q^*$. Analog zu dem Kompositionsoperator für Beziehungsklassen (Formel 6) lässt sich der Kompositionsoperator für Beziehungen $\otimes_Q : Q^* \times Q^* \to Q^*$ definieren. Im Folgenden bezeichne $Q_{c_1,c_2} = \{(e_1, e_2, t) \in Q^* \mid e_1 \in E_{c_1} \wedge e_1 \in E_{c_2}\}$ die Menge aller Beziehungen zwischen Objekten der Klasse $c_1 \in C$ und $c_2 \in C$.

Soll eine 2dA abgeleitet werden, um den Zusammenhang zwischen den Objekten zweier Klassen $c_1, c_2 \in C$ darzustellen, kann hierzu jede (direkte oder abgeleitete) Beziehungsklasse aus der Menge $R_{c_1,c_2} \subseteq R^*$ herangezogen werden:

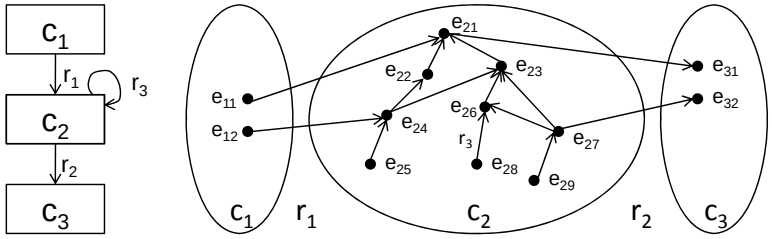
$$R_{c_1,c_2} = \{r \in R^* \mid (\pi_1(r) = c_1 \wedge \pi_2(r) = c_2) \vee (\pi_1(r) = c_2 \wedge \pi_2(r) = c_1))\}$$

Die einzelnen Beziehungsklassen in R_{c_1,c_2} unterscheiden sich im Typ und/oder der Richtung. Im Fall $R_{c_1,c_2} = \varnothing$ kann keine 2dA zwischen den Klassen c_1 und c_2 abgeleitet werden. Im Fall $\left|R_{c_1,c_2}\right| > 1$ stehen mehrere Beziehungsklassen unterschiedlichen Typs und/oder Richtung zur Verfügung, welche alternativ zur Ableitung einer 2dA herangezogen werden können. Auch lässt sich für jede Klasse c_1 die Menge $C_{c_1} \subseteq C$ aller Klassen angeben, die überhaupt zur Ableitung einer 2dA in Frage kommen: $R_{c_1c_2} \neq \varnothing \Rightarrow c_2 \in C_{c_1}$. Beachte, dass die Granularität der Beziehungstypen

und die Definition des Kompositionsoperators $\otimes$ potentiellen Einfluss auf die Anzahl der Typen und auf die Mächtigkeit der Menge R_{c_1,c_2} haben.

4. Reflexive Beziehungsklassen

In der bisherigen Betrachtung wurden reflexive Kanten im Metamodell, d. h. Beziehungsklassen einer Gestaltungsklasse zu sich selbst mit $(c,c,t) \in R$ für $c \in C, t \in T^*$, ausgeklammert. Beispiele für reflexive Beziehungsklassen sind die Assoziationen *kommuniziert mit* zwischen Objekten der Kasse *Organisationseinheit* oder *ist vernetzt mit* zwischen Objekten der Klasse *Server*.



(a) Metamodell (b) Modell

Abbildung 3: Reflexive Beziehungsklasse im Kontext

Abbildung 3 zeigt Beziehungen von und zu Objekten einer Klasse c_2 mit einer reflexiven Beziehungsklasse r_3. Für eine Klasse $c \in C$ bezeichne $Q_c = \{(e_1,e_2,t) \in Q^* \mid e_1 \in E_c \wedge e_2 \in E_c\}$ die Menge aller Beziehungen zwischen Objekten der Klasse c, die definitionsgemäß Beziehungen reflexiver Beziehungsklassen sind. Bei der Ableitung der komponierten Beziehungen zwischen Objekten der Klassen c_1 und c_3 gibt es, in Abhängigkeit von der Semantik der reflexiven Beziehungsklasse r_3, verschiedene Möglichkeiten. Beispielsweise ist denkbar, dass zwei Objekte $e_1 \in E_{c_1}$ und $e_3 \in E_{c_3}$ dann über eine komponierte Beziehung $(e_1,e_3,t_1 \otimes t_2) \in Q_{c_1c_3}$ verbunden sind, wenn es ein Objekt $e_2 \in E_{c_2}$ gibt mit $(e_1,e_2,t_1) \in Q_{c_1c_2}$ und $(e_2,e_3,t_2) \in Q_{c_2c_3}$ für $t_1,t_2 \in T^*$ (direkte Verbindung). In Abbildung 3 ist dies beispielsweise für die Objekte e_{11} und e_{31} der Fall, welche über das Objekt e_{21} in Beziehung stehen. Dies entspricht der Semantik des in Abschnitt 3 eingeführten Kompositionsoperators $\otimes_Q$. Eine weitere Möglichkeit ist, dass e_1 und e_3 über die transitive Hülle der Beziehungen aus Q_{c_2} verbunden sind (d. h. es gilt $(e_1,e_3,t_1 \otimes t_3 \otimes t_2) \in Q_{13}$, wenn es Objekte $e_{21},e_{22} \in E_{c_2}$ gibt mit $(e_1,e_{21},t_1) \in Q_{c_1c_2}$ und $(e_{22},e_3,t_2) \in Q_{c_2c_3}$ und $(e_{21},e_{22},t_3) \in Q_{c_2}^+$ für $t_1,t_2,t_3 \in T^*$) (transitive Verbindung). In Abbildung 3 ist dies beispielsweise für die Objekte e_{12} und e_{31} der Fall, welche über die Objekte e_{24}, e_{22} und e_{21} in Beziehung stehen. Dies kann auch so interpretiert werden, dass die Beziehung zwischen e_{12} und e_{24} implizit auch zwischen e_{12} und e_{22} sowie zwischen e_{12} und e_{21} gilt.

Neben der direkten Verbindung und der transitiven Verbindung sind weitere kontextspezifische Semantiken für reflexive Beziehungsklassen und deren Zusammenwirken denkbar, insbesondere wenn eine Klasse mehrere reflexive Beziehungsklassen (z. B. vom Typ Generalisierung und Aggregation) besitzt. Als allgemeiner Ansatz zur Spezifikation verschiedener Semantiken reflexiver Beziehungsklassen lassen sich Funktionen der Form $ext_{c_1c_2} : E_{c_1} \times E_{c_2} \times T^* \rightarrow Pot(E_{c_1} \times E_{c_2} \times T^*)$ angeben, die eine Beziehung zwischen Objekten der Klassen $c_1,c_2 \in C$ expandieren. *Pot(X)* bezeichne dabei die Potenzmenge, also die Menge aller Teilmengen, der Menge X. Die Expansion

einer Beziehung bedeutet, dass weitere Beziehungen erzeugt werden, die aufgrund der expliziten Beziehung implizit gelten. Nun kann die Gesamtmenge aller expandierten Beziehungen $Q^{\exp}$ wie folgt berechnet werden:

$$Q_{c_1c_2}^{\exp} = \bigcup_{q \in Q_{c_1c_2}} \exp_{c_1c_2}(q) \text{ und } Q^{\exp} = \bigcup_{c_1,c_2 \in C} Q_{c_1c_2}^{\exp}$$

Wobei die Funktionen $ext_{c_1c_2}$ für je zwei Klassen $c_1, c_2 \in C$ die gewünschte Semantik der reflexiven Beziehungsklassen repräsentieren muss. Die Komposition der Beziehungen und die Berechnung der transitiven Hülle $Q^{\exp^*}$ der expandierten Beziehungen kann zu Q^* mit Hilfe des in Abschnitt 3 eingeführten Kompositionsoperators $\otimes_Q$ durchgeführt werden.

5. Hierarchische Verfeinerung

Eine häufige Verwendung reflexiver Beziehungsklassen bei der Modellierung von Unternehmensarchitekturen ist die hierarchische Verfeinerung von Gestaltungsobjekten. Beispielsweise haben sich in den Praxisfallstudien reflexive Beziehungsklassen der Beziehungstypen „Aggregation“ (z. B. zur Zerlegung eines Geschäftsprozesses in Teilprozesse oder von Softwarekomponenten in Subkomponenten) und „Generalisierung“ (z. B. zur Zusammenfassung mehrere Produkte zu einer Produktlinie) als wichtiges Modellierungsvokabular herausgestellt. In diesem Abschnitt wird daher untersucht, wie die Expansionsfunktion *exp* für hierarchische Verfeinerungsbeziehungen zu definieren ist. Wir nehmen hierzu an, dass die Gestaltungsobjekte E_{c_1} einer Klasse c_1 in einer Verfeinerungsbeziehung stehen, die durch eine Relation auf $P \subseteq E_{c1} \times E_{c1}$ beschrieben ist: $(e_1, e_2) \in P$ bedeutet, dass e_1 eine Verfeinerung (z. B. eine Spezialisierung oder Teil ein Zerlegung) von e_2 ist. Die Verfeinerungsrelation kann aus den Beziehungen aus Q_{c_1} abgeleitet werden, im einfachsten Fall durch $P = \{(\pi_1(q), \pi_2(q)) \mid q \in Q_{c_1}\}$. Durch die Verfeinerungsbeziehung soll ausgedrückt werden, dass ein Objekt e durch ein oder mehrere *untergeordnete* Objekte verfeinert wird und/oder ein oder mehrere *übergeordnete* Objekte verfeinert. Nicht erlaubt ist, dass ein Objekt eine Verfeinerung von sich selbst ist, weshalb der durch die Verfeinerungsrelation aufgespannte Graph kreisfrei (nicht zu verwechseln mit zykelfrei) sein muss. Der in Abbildung 3 dargestellte Verfeinerungsbaum der Beziehungen aus Q_{c_2} erfüllt diese Bedingung.

In Anlehnung an [4] definieren wir für ein Objekt $e \in E$ die Verfeinerungsrelation $P_{e\downarrow}$, welche alle e untergeordneten Objekte auf e abbildet: $P_{e\downarrow} = P^+\big|_{ran}\{e\}$. P^+ ist dabei die transitive Hülle von P, die im Wertebereich auf die Menge $\{e\}$ eingeschränkt wird ($\big|_{ran}$). Der Definitionsbereich $E_{e\downarrow} = dom(P_{e\downarrow})$ beinhaltet alle e untergeordneten Objekte. Analog definieren wir die Verfeinerungsrelation $P_{e\uparrow}$, welche e auf alle e übergeordneten Objekte abbildet: $P_{e\uparrow} = P^+\big|_{dom}\{e\}$. Der Wertebereich $E_{e\uparrow} = ran(P_{e\uparrow})$ beinhaltet alle e übergeordneten Objekte. In Abbildung 3 gilt beispielsweise $P_{e_{23}\downarrow} = \{e_{26}, e_{28}, e_{27}, e_{29}\}$ und $P_{e_{27}\uparrow} = \{e_{26}, e_{23}, e_{21}\}$. Gegeben seien zwei Gestaltungsklassen c_1 und c_2 und zwei Gestaltungsobjekte $e_1 \in c_1$ und $e_2 \in c_2$, für die jeweils sowohl übergeordnete als auch untergeordnete Objekte existieren. e_1 und e_2 stehen weiterhin miteinander in Beziehung, d. h. es existiert ein $q \in Q_{c_1c_2}$ mit $q = (e_1, e_2, t)$ für ein $t \in T^*$. Die Fallstudienbeispiele zeigen, dass eine Be-

ziehung von bzw. zu einem Gestaltungsobjekt e, das sich in einer hierarchischen Verfeinerungsbeziehung befindet, mit unterschiedlicher Semantik belegt sein kann (Gültigkeitstypen):

(1) Die Beziehung gilt ausschließlich für das Objekt e. Es wird keine Aussage getroffen bezüglich übergeordneter und untergeordneter Objekte. (Beispiel: Beziehung *hat Entwicklungsverantwortung für* eines *IT-Zulieferers* zu *Softwarekomponenten*.)
(2) Die Beziehung gilt implizit auch für alle untergeordneten Objekte. (Beispiel: Beziehung *ist verantwortlich für* zwischen *Organisationseinheiten* und *Geschäftsprozesse*n, wenn die Organisationseinheit auch stets verantwortlich für alle Teilprozesse ist.)
(3) Die Beziehung gilt implizit auch für alle übergeordneten Objekte. (Beispiel: Beziehung *wird genutzt in* zwischen *Applikationen* und *Geschäftsprozessen*. Wird eine Applikation in einem Teilprozess genutzt, wird diese definitionsgemäß auch in dem übergeordneten Prozess genutzt.)
(4) Die Beziehung gilt implizit auch für alle übergeordneten und untergeordneten Objekte.

Bei der Definition der Expansionsfunktion $ext_{c_1c_2} : E_{c_1} \times E_{c_2} \times T^* \rightarrow Pot(E_{c_1} \times E_{c_2} \times T^*)$ müssen diese unterschiedlichen Semantiken berücksichtigt werden. Für eine Beziehung $q \in Q_{c_1c_2}$ gilt sowohl für das Ausgangsobjekt e_1 als auch für das Zielobjekt e_2 je einer der vier Gültigkeitstypen, wodurch sich 16 Erweiterungsfunktionen ergeben: $ext_{c_1c_2--}$, $ext_{c_1c_2\uparrow-}$, $ext_{c_1c_2\downarrow-}$, $ext_{c_1c_2\updownarrow-}$, $ext_{c_1c_2-\uparrow}$, $ext_{c_1c_2\uparrow\uparrow}$, $ext_{c_1c_2\downarrow\uparrow}$, $ext_{c_1c_2\updownarrow\uparrow}$, $ext_{c_1c_2-\downarrow}$, $ext_{c_1c_2\uparrow\downarrow}$, $ext_{c_1c_2\downarrow\downarrow}$, $ext_{c_1c_2\updownarrow\downarrow}$, $ext_{c_1c_2-\updownarrow}$, $ext_{c_1c_2\uparrow\updownarrow}$, $ext_{c_1c_2\downarrow\updownarrow}$, $ext_{c_1c_2\updownarrow\updownarrow}$. Der erste Pfeil gibt für die Verfeinerungsbeziehung des Ausgangsobjekts an, ob die Beziehung nur für das Ausgangsobjekt ($-$), implizit für alle übergeordneten Objekte ($\uparrow$), alle untergeordneten Objekte ($\downarrow$) oder beides gilt ($\updownarrow$). Der zweite Pfeil gibt dies entsprechend für die Verfeinerungsbeziehung des Zielobjekts an. Zwei der Funktionen sind exemplarisch wie folgt definiert; alle übrigen Funktionen lassen sich analog definieren: $ext_{c_1c_2\uparrow-}((e_1,e_2,t)) = E_{e1\uparrow} \times \{e_2\} \times \{t\}$ und $ext_{c_1c_2\updownarrow\updownarrow}((e_1,e_2,t)) = E_{e1\uparrow} \cup E_{e1\downarrow} \times E_{e2\uparrow} \cup E_{e2\downarrow} \times \{t\}$

6. Aggregierte Betrachtung von Beziehungen

Oft sind die Strukturen komplex, die sich durch Beziehungen zwischen Objekten von hierarchisch verfeinerbaren Gestaltungsklassen ergeben. Zur Komplexitätsreduktion kann es in Abhängigkeitsanalysen sinnvoll sein, die unteren Verfeinerungsebenen auszublenden. Die Beziehungen der ausgeblendeten Gestaltungsobjekte werden dann zwischen den übergeordneten Objekten aggregiert betrachtet. Abbildung 4 zeigt hierfür ein Beispiel: Zwei Gestaltungsobjekte $e_1 \in c_1$ und $e_2 \in c_2$ sind hierarchisch verfeinert. Die Verfeinerungen der Objekte stehen auf dritter Verfeinerungsebene miteinander in Beziehung („X“). Jedes „X“ entspricht einen Element $q \in Q_{c_1c_2}$. In Abbildung 4 sind dies z. B. die Beziehungen $\{(e_{112},e_{23}),(e_{113},e_{222}),(e_{111},e_{221}),(e_{122},e_{221}),(e_{121},e_{211})\}$. Diese Beziehungen können auch aggregiert auf den Ebenen 2 („XX“) und 1 („XXX“) betrachtet werden. Abbildung 4 veranschaulicht weiterhin, wie das Ausblenden der unteren Verfeinerungsebenen Abhängigkeitsanalysen kompakter und übersichtlicher werden lässt. Die Diskussion mit Stakeholdern in den Praxisfallstudien zeigte, dass das beliebige Ein- und Ausblenden von Verfeinerungsebenen und Teilbäumen in Matrixdarstellungen als wichtige Navigationsoperation zur Erschließung der strukturellen Zusammenhänge der Unternehmensarchitektur eingeschätzt wird.

Diese aggregierten Beziehungen werden durch eine sogenannte „Lifting“-Transformation ($\uparrow$) erzeugt, die im Folgenden in Anlehnung an [4] und [7] eingeführt wird. Die Verfeinerungsrelation $P_{c_1} \subseteq E_{c_1} \times E_{c_1}$ beschreibe wieder die Verfeinerungsbeziehung zwischen den Gestaltungsobjekten

der Klasse c_1. Dafür stellen wir die zusätzlichen Bedingungen, dass die Relation funktional ist (d. h. für jedes e_a existiert höchstens ein e_b mit $e_a P_{c_1} e_b$) und azyklisch, so dass die Verfeinerungsrelation einen Baum ergibt wie in Abbildung 4. O. B. d. A. sollen in der Hierarchie (z. B. in der Matrixdarstellung) alle dem Objekt $e \in E_{e_1\downarrow}$ untergeordneten Objekte ausgeblendet werden (z. B. Objekt e_{11} in Abbildung 4).

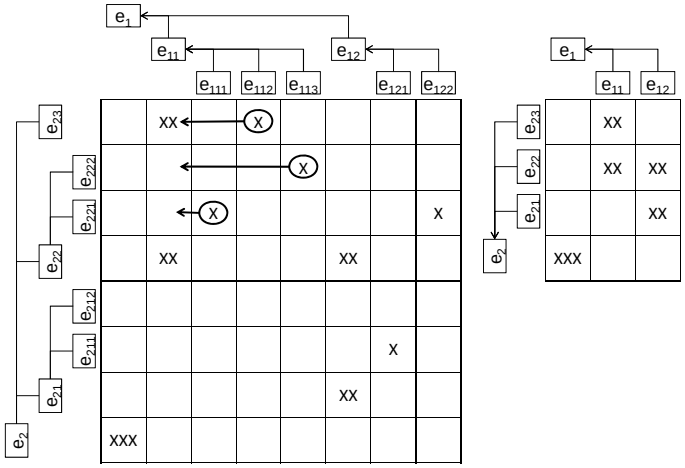


Abbildung 4: Aggregierte Betrachtung von Beziehungen (Verfeinerungsebenen 3 und 2) in einer Matrix-Darstellung

D. h. nur die Objekte $E_{\bar{e}} = E_{e_1\downarrow} - E_e\downarrow$ sollen berücksichtigt werden[4] (z. B. $\{e_1, e_{11}, e_{12}, e_{121}, e_{122}\}$). Der darzustellende Verfeinerungsbaum wird durch die Verfeinerungsrelation $P_{\bar{e}} = P_{c_1}\big|E_{\bar{e}}$ beschrieben. Dies ist die Teilmenge der Verfeinerungsrelation P_{c_1}, in der jedoch alle Tupel nur Elemente von $E_{\bar{e}}$ enthalten (z. B. $\{(e_{11}, e_1), (e_{12}, e_1), (e_{121}, e_{12}), (e_{121}, e_{12})\}$). In der Relation $Q_{\bar{e}c_2} \subseteq E_{\bar{e}} \times E_{c_2} \times T^*$ werden nun alle e untergeordneten Objekte durch e ersetzt. Hierzu definieren wir zunächst $I_{\bar{e}}$ als identische Abbildung auf $E_{\bar{e}}$. Die Relation $P_e = P_{c_1}^+\big|_{ran}\{e\}$, d. h. die transitive Hülle der Verfeinerungsrelation P_{c_1} eingeschränkt auf alle Tupel mit dem Wertebereich (range) e, bildet weiterhin alle e untergeordneten Objekte auf e ab (z. B. $\{(e_{111}, e_{11}), (e_{112}, e_{11}), (e_{113}, e_{11})\}$). Die Relation $M_{\bar{e}} = I_{\bar{e}} \cup P_e$ bildet dann alle e untergeordneten Objekte auf e und alle anderen Objekte auf sich selbst ab
$(\{(e_{111}, e_{11}), (e_{112}, e_{11}), (e_{113}, e_{11}), (e_1, e_1), (e_{11}, e_{11}), (e_{12}, e_{12}), (e_{121}, e_{121}), (e_{122}, e_{122})\})$. Die Relation $q_{\bar{e}}$ ist dann definiert durch $q_{\bar{e}} = M_{\bar{e}}^{-1} \circ q$, wofür die Kurzschreibweise mit dem Lifting-Operator verwendet werden kann: $q_{\bar{e}} = q \uparrow_{dom} M_{\bar{e}}$. Für die Relationen aus dem Beispiel ergibt sich: $\{(e_{11}, e_{23}), (e_{11}, e_{222}), (e_{11}, e_{221}), (e_{122}, e_{221}), (e_{121}, e_{211})\}$ (vgl. Pfeile in Abbildung 4). Diese Ausführungen betrachteten die Seite des Deinitionsbereichs (domain) der Beziehung q. Analog kann die Lifting-Operation auf dem Wertebereich der Relation durchgeführt werden. In diesem Fall gilt: $q \uparrow_{ran} M_{\bar{e}} = q \circ M_{\bar{e}}$.

[4] Der Operator $-$ steht an dieser Stelle für die (asymmetrische) Differenz zwischen Mengen.

7. Zusammenfassung und Ausblick

In diesem Beitrag wurde ausgehend von dem Bedarf, für beliebige Unternehmensarchitekturmodelle flexible Analysen ableiten zu können, ein allgemeiner formaler Lösungsansatz für zweidimensionale Abhängigkeitsanalysen auf Unternehmensarchitekturmodellen vorgestellt. In einem ersten Schritt wird hier das mathematische Fundament für Abhängigkeitsanalysen auf Unternehmensarchitekturmodellen beschrieben, das in nachfolgenden Arbeiten wie folgt erweitert und überprüft werden kann:

1) Betrachtung mehrdimensionaler Abhängigkeiten: Auch mehrdimensionale Abhängigkeiten können betrachtet werden, beispielsweise indem in den Zellen einer Matrixdarstellung weitere Informationen dargestellt werden. Dies entspricht einer verschränkten Ausführung zweidimensionaler Abhängigkeitsanalysen.
2) Betrachtung weiterer Darstellungsformen und Manipulationsmechanismen: Neben den Matrixanalysen können weitere Visualisierung wie z. B. Dependency-Diagramme, Landscape-Maps oder Wiring-Diagramme abgeleitet werden. Darüber hinaus können verschiedene Manipulationsoperationen für die verschiedenen Darstellungsformen spezifiziert werden, um beispielsweise Bestandteile der Ergebnisse ein- und auszublenden oder Darstellungsformen ineinander zu überführen.
3) Prototypische Werkzeugumsetzung: Das Ziel der hier beschriebenen Arbeit ist es, die Grundlage zu legen für flexiblere Softwarewerkzeuge. Im Rahmen eines Proof-of-Concept sollte der Ansatz daher prototypisch umgesetzt werden.

8. Referenzen

[1] AIER, S., KURPJUWEIT, S., RIEGE, C., SAAT, J.: Stakeholderorientierte Dokumentation und Analyse der Unternehmensarchitektur, in: Hegering, H.-G., Lehmann, A., Ohlbach, H.J., Scheideler, C. (eds.): Proceedings of INFORMATIK 2008: Beherrschbare Systeme – dank Informatik, Vol. LNI P-134, Band 2. GI/Köllen, Bonn 2008, S. 559–565

[2] AIER, S., KURPJUWEIT, S., SCHMITZ, O., SCHULZ, J., THOMAS, A., WINTER, R.: An Engineering Approach to Enterprise Architecture Design and its Application at a Financial Service Provider: Proceedings Modellierung betrieblicher Informationssysteme (MobIS 2008). GI/Köllen, Bonn 2008, noch nicht erschienen.

[3] AIER, S., RIEGE, C., WINTER, R.: Unternehmensarchitektur - Literaturüberblick und Stand der Praxis, in: Wirtschaftsinformatik, Vol. 50, 2008, S. 292-304

[4] BRIL, R., FEIJS, L., GLAS, A., KRIKHAAR, R., WINTER, T.: Hiding Expressed Using Relation Algebra with Multi-Relations Oblique Lifting and Lowering for Unbalanced Systems: Proceedings of the Conference on Software Maintenance and Reengineering. IEEE Computer Society 2000

[5] VAN BUUREN, R., JONKERS, H., IACOB, M.-E., STRATING, P.: Composition of Relations in Enterprise Architecture Models: ICGT 2004, S. 39–53

[6] FAVRE, J.-M.: Foundations of Meta-Pyramids: Languages vs. Metamodels – Episode II: Story of Thotus the Baboon1, in: Bezivin, J., Heckel, R. (eds.), Dagstuhl 2005

[7] FEIJS, L.M.G., VAN OMMERING, R.C.: Relation partition algebra \— mathematical aspects of uses and part-of relations, in: Sci. Comput. Program., Vol. 33, 1999, S. 163-212

[8] IEEE: IEEE Recommended Practice for Architectural Description of Software Intensive Systems (IEEE Std 1471-2000). IEEE Computer Society, New York, NY 2000

[9] KÜHNE, T.: Matters of (meta-) modeling, in: Software and Systems Modeling, Vol. 5, 2006, S. 369-385

[10] NIEMANN, K.D.: Von der Unternehmensarchitektur zur IT-Governance: Leitfaden für effizientes und effektives IT-Management. Vieweg 2005

[11] SINZ, E.J.: Architektur von Informationssystemen, in: Rechenberg, P., Pomberger, G. (eds.): Informatik-Handbuch. Hanser, München, Wien 1999 S. 1035-1046

[12] STRAHRINGER, S.: Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips, in: Pohl, K., Schürr, A., Vossen, G. (eds.): 1998, S. 15-20

[13] WINTER, R., FISCHER, R.: Essential Layers, Artifacts, and Dependencies of Enterprise Architecture, in: Journal of Enterprise Architecture, Vol. 3, 2007, S. 7-18