Modellierung graphartiger Dokumente

Udo Kelter

22.11.2010

Zusammenfassung dieses Lehrmoduls

Dieses Lehrmodul behandelt die Analyse von Systemen, die graphartige Dokumente verwalten. Bei derartigen Dokumenten ist die konzeptuelle Grobstruktur ein abstrakter Graph. Dargestellt werden die Dokumente bevorzugt graphisch durch graphische Knoten und Linien dazwischen. Das graphische Layout wird i.d.R. manuell erstellt und muß daher neben den konzeptuellen Inhalten ebenfalls persistent verwaltet werden. Vorgestellt werden mehrere Muster bzw. Regeln zur Gestaltung von Analysedatenmodellen für graphartige Dokumenttypen.

Vorausgesetzte Lehrmodule:

obligatorisch: – Transformation von Analyse-Datenmodellen in Entwurfsdokumente

empfohlen: – Analysemuster (Stichworte)

Stoffumfang in Vorlesungsdoppelstunden: 1.0

Inhaltsverzeichnis

1 Einleitung

Wenn man Datenbestände modelliert, liegen typischerweise zunächst Darstellungen dieser Daten auf Papier vor. Bei betrieblichen Anwendungen erscheinen diese Daten häufig in Form von Listen, Tabellen, Karteikarten o.ä. und es ist einfach, entsprechende Entitätsmengen zu erkennen. Die Formatierung oder optische Gestaltung der Darstellungen auf Papier spielt keine Rolle und wird i.d.R. bei der Modellierung nicht berücksichtigt.

Die vorstehende Annahmen treffen nicht mehr zu, wenn man Daten modelliert, die als graphartige Dokumente dargestellt werden. Ein Dokument ist ein **graphartiges Dokument**, wenn

1. dessen wesentliche *konzeptuelle Struktur* als abstrakter Graph angesehen werden kann. Ein abstrakter Graph (im Sinne der Graphentheorie) ist eine Struktur aus Knoten und Kanten zwischen den Knoten. Wir nennen sie abstrakt, weil wir von allen Implementierungsdetails in konkreten Datenstrukturen abstrahieren.
2. das Dokument typischerweise als ein Netzwerk von graphischen Knotenpunkten und Verbindungslinien in einem zweidimensionalen Bild graphisch dargestellt wird[1]. Die Verbindungslinien bezeichnen wir auch als "graphische Kante".

Klassische, teilweise uralte Beispiele für solche Dokumente sind:

- Stammbäume
- Bus-, Zug-, Flugpläne
- Straßenkarten
- Moleküldarstellungen
- elektrische Schaltungen
- Verlegepläne für Heizungsrohre
- Brettspiele, z.B. "Mensch ärgere Dich nicht"
- Begriffsnetze (*mind maps*)

[1] Der Begriff graphisch ist sehr allgemein und umfaßt viele andere bildliche Darstellungsformen, z.B. Torten- oder Balkendiagramme, mit denen alle erdenklichen Arten von Daten dargestellt werden können, nicht nur graphartig strukturierte.

Darüber hinaus treten in der Informatik vielfach graphartige Dokumente auf, meist als Entwicklungsdokumente in Projekten:

- Kontrollflußgraphen von Programmen oder Geschäftsprozessen
- "benutzt"-Diagramme, z.B. die Import-Beziehungen zwischen Modulen
- Syntaxdiagramme von Programmiersprachen
- diverse UML-Modelltypen

Thema dieses Lehrmoduls ist die Systemanalyse von Systemen, die z.B. beliebige Stammbäume verarbeiten können. Allgemeiner ausgedrückt soll ein derartiges System beliebige graphartige Dokumente eines bestimmten Dokumenttyps verarbeiten können.

Wir behandeln hier fast ausschließlich die Datenmodellierung graphartiger Dokumente, die bei allen Dokumenttypen viele Gemeinsamkeiten aufweist. Auf die Funktionsmodellierung gehen wir nur am Rande ein.

2 Layout und Layoutdaten

Bei näherem Betrachten der vorstehenden Beispiele fällt auf, daß man die "wesentliche Information", die in den graphischen Darstellungen enthalten ist, auch in Tabellen darstellen könnte, also wie übliche betriebliche Daten. Wo ist also der Unterschied?

Ein Hauptunterschied liegt in der Wertigkeit des Layouts: Bei graphartigen Dokumenten ist die graphische Darstellung die bevorzugte Darstellung, weil ein gutes Layout die Verständlichkeit des Dokuments erheblich verbessert. Ein Layout ist "gut", wenn es für den Dokumenttyp sinnvolle Layoutregeln einhält; letztlich ist aber die Bewertung des Layouts oft Geschmackssache. Daher wird typischerweise verlangt, daß

- die Dokumente in der graphischen Darstellung editiert werden können,
- das Layout manuell festgelegt werden kann und
- das Layout daher persistent gespeichert wird.

Im Gegensatz dazu ist bei anderen Dokumentarten das Layout von Darstellungen auf einzelnen Ausgabemedien irrelevant, es wird bei jeder Ausgabe neu generiert und nicht persistent gespeichert.

Layout. Unter dem **Layout** der Darstellung eines graphartigen Dokuments verstehen wir folgende Merkmale:
- die Position von Knotenpunkten auf der Zeichenfläche
- Größe u.a. geometrische Eigenschaften der Knoten
- die Position von Stützpunkten von Verbindungslinien
- Farbe, Strichdicke, Strichelung und andere optische Eigenschaften der Verbindungslinien
- die relative Position von Beschriftungen an den Verbindungslinien

usw., wobei konkrete Details spezifisch für einzelne Ausgabemedien sein können.

Die Daten, die das Layout der graphischen Darstellung darstellen, bezeichnen wir i.f. als **Layoutdaten**. Die übrigen Daten werden in diesem Zusammenhang meist als **konzeptuelle Daten** bezeichnet.

2.1 Modellierung der Layoutdaten

Layoutdaten kann man im Kontext der Systemanalyse etwas zugespitzt als Zwitterwesen bezeichnen, weil oft unklar ist, ob und inwieweit sie im Rahmen der Systemanalyse behandelt werden sollen:

- Sie sind zwar sehr relevant für graphische externe Darstellungen, aber irrelevant für nichtgraphische externe Darstellungen, inhaltliche Konsistenzprüfungen und fast alle Weiterverarbeitungen der Dokumente. Sie werden daher aus einer Gesamtsicht als eher zweitklassig eingestuft.
- Sobald man versucht, das Layout konkreter zu spezifizieren, benötigt man die Begriffswelt eines Graphiksystems. In vielen Fällen kann oder will man sich bei der Systemanalyse aber noch nicht auf ein konkretes Graphiksystem bzw. -Produkt festlegen. Im Endeffekt benutzt man oft verbale Beschreibungen des gewünschten Layouts, also keine strukturierten Datenmodelle.

Eine direkte Konsequenz hieraus ist, daß die Layoutdaten und die konzeptuellen Daten sauber getrennt werden müssen.

Zusätzlich verkompliziert wird die Behandlung von Layout dadurch, daß ggf. mehrere Ausgabemedien parallel unterstützt werden müssen, also mehrere Layouts zum gleichen abstrakten Dokument verwaltet werden müssen, und daß selbst auf dem gleichen Ausgabemedium verschiedene Kontexte vorhanden sein können, in denen das ganze Dokument oder Teile davon unterschiedlich gelayoutet sein müssen. Im Extremfall sind Einzelheiten des Layouts sogar benutzerspezifisch und jederzeit dynamisch änderbar.

Wir werden daher in diesem Lehrmodul nicht auf die detaillierte Modellierung der Layoutdaten eingehen. Nichtsdestotrotz werden wir die graphischen Darstellungen der Dokumente als Ausgangsbasis einer Daten- bzw. Systemanalyse betrachten, denn anfänglich sind oft gar keine anderen Darstellungen verfügbar, und sie sind i.d.R. die am besten verständlichen Darstellungen.

2.2 Diagramme

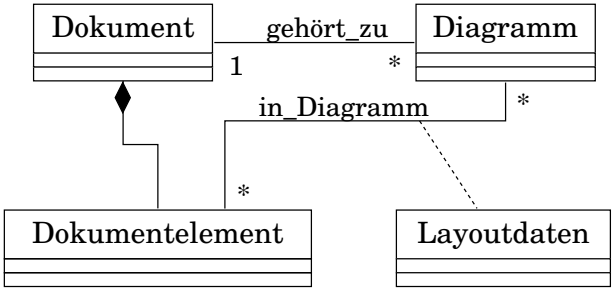
Graphische Darstellungen können nur relativ kleine Graphen übersichtlich darstellen. Abhängig von der geometrischen Größe der Knoten, der Zahl der Kanten pro Knoten und ggf. dem Umfang von Beschriftungen sind nur bis zu ca. 20 bis 50 Knoten übersichtlich darstellbar.

In vielen Fällen sind die abstrakten Graphen aber wesentlich größer. Ein häufig benutzter Ansatz zur Lösung dieses Problem sind Diagramme, die Teilgraphen darstellen. Unter einem **Teilgraphen TG** eines Graphen G verstehen wir hier eine Teilmenge der Knoten und Kanten von G, die

- aus Sicht der Anwendung sinnvoll ist, weil sie z.B. einer bestimmten Fragestellung entspricht,
- manuell zusammengestellt wird und
- ein eigenständiges Layout hat.

Ein Teilgraph wird zusammen mit seinem Layout als **Diagramm** be-

zeichnet. Zu einem Gesamtgraphen G kann es beliebig viele Diagramme geben. Die Diagramme sind i.d.R. nicht disjunkt, d.h. ein Knoten (bzw. eine Kante) kann in mehreren Diagrammen auftauchen. In den meisten Anwendungskontexten ist es zulässig, daß ein Knoten in keinem einzigen Diagramm vorkommt, weil zusätzlich nichtgraphische Darstellungen vorhanden sind, über die man alle Knoten erreichen kann. Hieraus ergibt sich das folgende Analyseklassendiagramm, das den Zusammenhang zwischen Dokumenten und Diagrammen zeigt; auf dessen Details gehen wir später genauer ein.



3 Abstrakte Graphen

In diesem Abschnitt definieren wir den schon oben benutzten Begriff eines abstrakten Graphen genauer.

3.1 Grundform

Ein **abstrakter Graph G** = (N,E) ist gegeben durch

- eine Knotenmenge **N** (*nodes*)
- eine Kantenmenge **E** (*edges*).

Im Normalfall verbindet eine Kante zwei Knoten. Es gibt sehr viele Erweiterungen dieser Grundform und Sonderformen. Diese Varianten und zugehörige Algorithmen werden in der Graphentheorie ausführlich behandelt. Wir gehen i.f. nur auf die Gerichtetheit von Kanten und auf die Attributierung von Graphen ein.

Gerichtete Graphen: Bei einem gerichteten Graphen sind die Kanten gerichtet. Eine Kante geht "von" einem Knoten n1 "nach" einem Knoten n2. Eine Kante kann daher als Tupel dargestellt werden, das als erste Komponente den Ausgangsknoten und als zweite den Zielknoten angibt. Formal ist daher die Kantenmenge **E** eine Teilmenge des Kreuzprodukts von **N** mit sich selbst, oder anders formuliert:

$$\mathbf{E} \subseteq \{\ (\text{n1,n2}) \mid n1, n2 \in \mathbf{N}\ \}$$

Erlaubt sind insb. folgende Fälle:

- $(n1, n1) \in \mathbf{E}$, d.h. die Kante ist optisch gesehen eine "Schlaufe" und führt zum Ausgangsknoten zurück.
- $(n1, n2) \in \mathbf{E}$ und $(n2, n1) \in \mathbf{E}$, d.h. die beiden Knoten n1 und n2 sind durch zwei gegenläufige Kanten in beiden Richtungen verbunden.

  Wenn eine Kante $(n1, n2) \in \mathbf{E}$ vorhanden ist, impliziert dies nicht, daß auch die gegenläufige Kante $(n2, n1) \in \mathbf{E}$ vorhanden ist. Beide Kanten existieren unabhängig voneinander.

Eine bekannte Sonderform von gerichteten Graphen sind Bäume, die man als gerichtete Graphen ansehen kann, in denen die Kanten die Pfade von der Wurzel zu den Blättern (oder umgekehrt) bilden.

Ungerichtete Graphen: Bei ungerichteten Graphen stellt eine Kante nur den Sachverhalt dar, daß zwei Knoten in Verbindung stehen, eine Unterscheidung von Rollen in dieser Verbindung ist nicht sinnvoll. Formal ist daher die Kantenmenge **E** ein Menge von Teilmengen von **N**:

$$\mathbf{E} \subseteq \{\ e \mid e \subseteq \mathbf{N} \text{ und } 1 \leq |e| \leq 2\ \}$$

Wenn $e = \{n1, n2\}$, dann sind die Knoten n1 und n2 verbunden (ohne Kenntnis einer Richtung). Wenn $|e| = 1$ ist, liegt eine Schlaufe vor; dieser Fall ist oft nicht sinnvoll und wird dann ausgeschlossen.

Übungsaufgabe: Bei welchen in der Einleitung genannten Beispielen treten ungerichtete Graphen auf?

3.2 Typisierte bzw. attributierte abstrakte Graphen

In den meisten Fällen stellen die abstrakten Graphen nur die Grobstruktur eines Dokuments dar, man findet zusätzlich in den graphischen Darstellungen, die ja Ausgangspunkt unserer Systemanalyse sind, diverse Beschriftungen an oder in den Knoten und Kanten.

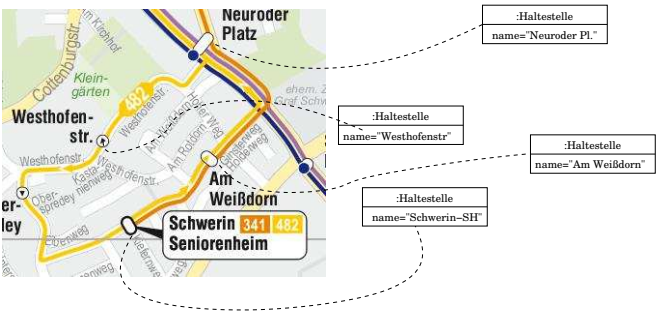
Häufig sind mehrere Typen von graphischen Knoten und Kanten mit unterschiedlichen Formen, Farben, Beschriftungsmustern usw. erkennbar, daher ist folgendes Vorgehen bei der Analyse eines Dokumenttyps sinnvoll:

- Es wird eine Menge von Typen abstrakter Knoten und abstrakter Kanten definiert, die bei der gegebenen Dokumentklasse auftreten können.
- Jedem abstrakten Knoten und jeder abstrakten Kante ist genau ein Kantentyp zugeordnet.
- Die Knoten- bzw. Kantentypen definieren (Analyse-) Attribute, die bei den Knoten bzw. Kanten dieses Typs vorhanden sind.

4 Beispiel: Busplan

Als Beispiel betrachten wir i.f. einen Busplan, s. untenstehenden Kartenauszug. Die Typisierung der abstrakten Graphen ist hier recht einfach: Wir benötigen nur einen abstrakten Knotentyp, den wir “Haltestelle” nennen, und nur einen abstrakten Kantentyp, den wir “Streckenabschnitt” nennen.

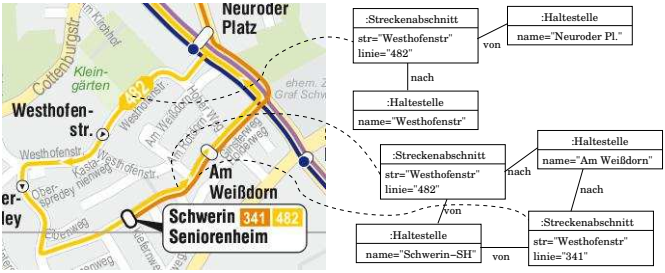
Modellierung von Haltestellen: Haltestellen haben offensichtlich einen Namen, der neben dem graphischen Symbol steht. Das folgende Bild zeigt beispielhaft für 4 graphische Knoten entsprechende Knoten eines abstrakten Graphen. Die abstrakten Knoten notieren wir in Form von Objekten gemäß der UML-Notation. Diese Knoten (bzw. die “Haltestelle”-Objekte) haben ein Attribut “name” für den Namen der Haltestelle. Jeder abstrakte Knoten ist mit einer gestrichelten Linie mit dem zugehörigen graphischen Knoten verbunden.



2. Repräsentation von Streckenabschnitten: Das nächste Bild zeigt beispielhaft, wie graphische Kanten im abstrakten Graphen zu repräsentieren sind. Wir haben nur einen Kantentyp "Streckenabschnitt", erkennbare Merkmale sind z.B. die Nummer der Buslinie und der Name der Straße.

Naheliegend ist hier, eine graphische Kante durch eine abstrakte Kante zu repräsentieren, der man passende Attribute zuordnet. Diese Attribute würde man in der UML-Notation der Kante in Form eines assoziativen Objekts zuordnen. Dieser Ansatz ist aus mehreren Gründen ungünstig, u.a. weil die Kanten Dokumentelemente sind, denen man Layoutdaten in den Diagrammen, in denen sie vorkommen, zuordnen können muß. Eine Beziehung kann aber bei üblichen Modellierungsregeln keine Rolle in einer weiteren Beziehung spielen.

Daher repräsentieren wir eine graphische Kante durch einen abstrakten Knoten. Im folgenden Bild sind diese abstrakten Knoten neu eingetragen und mit einer gestrichelten Linie mit der zugehörigen graphischen Kante verbunden. Von einem abstrakten Knoten, der eine graphische Kante repräsentiert, gehen zwei abstrakte Kanten aus. Diese führen zu den beiden abstrakten Knoten, die die graphischen Knoten repräsentieren, die durch die graphische Kante verbunden sind.



5 Sind abstrakte Graphen Modelle?

Die mathematischen Definitionen von Graphen sind insofern abstrakt, als sie keine Bezüge auf Datenstrukturen einer Programmiersprache oder eines Datenverwaltungssystems beinhalten, in diesem Sinne also *technologiefrei* sind. Ihrem Abstraktionsgrad nach sind sie somit *der Analysephase zuzuordnen.* Dies führt zu der Frage, ob man sie ähnlich wie Klassendiagramme auch als Analysemodelle ansehen kann. Die Antwort ist nein, die Begründung ist nicht ganz trivial.

Ein abstrakter Graph repräsentiert genau eine mögliche Grobstruktur, die bei einem Dokument in der modellierten Dokumentklasse auftreten kann. Ein anderes Dokument hat i.a. eine ganz andere Grobstruktur und demzufolge einen anderen abstrakten Graphen. Ein einzelner abstrakter Graph modelliert daher nicht die komplette Dokumentklasse, sondern allenfalls diejenigen Dokumente, die diese Grobstruktur aufweisen, die sich aber bei den Attributwerten unterscheiden können. Ein einzelner abstrakter Graph ist daher bestenfalls ein Modell im Sinne eines Beispiels, wie eine Dokumentstruktur aussehen kann, ähnlich wie ein Sequenzdiagramm einen denkbaren Ablauf von Operationsaufrufen zeigt.

Für die weitere Systementwicklung benötigen wir indes Modelle, die die Struktur *aller* Entitäten modellieren, die das geplante System verarbeiten soll, hier also aller Dokumente des zu unterstützenden Dokumenttyps. Hierzu eignen sich nur Analyseklassendiagramme oder äquivalente Modelltypen.

Abstrakte vs. implementierte Graphen. Aus den Analysemodellen können wir mit den bekannten Methoden [AMU] Entwicklungsdokumente der späteren Phasen ableiten, die die abstrakten Graphen in konkreten Technologien implementieren. Normalerweise werden sowohl persistente wie transiente Implementierungen benötigt:

1. *persistente* Implementierungen in Tabellen, XML-Dateien u.ä.
2. *transiente* Implementierungen in Form von Ecore-Objektnetzwerken, Adjazenzlisten (Nachbarschaftslisten) oder in Form einer Adjazenz- oder Inzidenzmatrix. Hierbei hängen viele Details der Datenstrukturen von der jeweiligen Programmiersprache ab. Da die Algorithmen immer auf transienten Implementierungen basieren, spielen bei der Wahl dieser Datenstrukturen auch Effizienzfragen, letztlich also Leistungsanforderungen der Applikation, eine große Rolle.

Es sei an dieser Stelle vor einer Verwechslung gewarnt: Adjazenzlisten u.ä. Datenstrukturen werden zwar oft auch als "Graphen" bezeichnet, sie sind aber technologiegebundene transiente Implementierungen von abstrakten Graphen und begrifflich der Entwurfsphase zuzuordnen.

6 Analyse-Datenmodelle von graphartigen Dokumenten

In diesem Abschnitt stellen wir die allgemeinen Regeln zusammen, wie analog zum Beispiel in Abschnitt 4 Analyseklassendiagramme für graphartige Dokumente entwickelt werden können.

6.1 Modellierung von Dokumenten und Diagrammen

Das Analyse-Klassendiagramm hierzu wurde schon in Abschnitt 2.2 vorgestellt. I.f. gehen wir kurz auf die enthaltenen Modellelemente ein.

Analyseklasse "Dokument": Normalerweise muß man mehrere Dokumente verwalten können - zumindest persistent -, selbst wenn ein Editor nur ein Dokument gleichzeitig verarbeiten kann. Typische Attribute sind: der Name des Dokuments, Autoren, Stand und diverse administrative Daten, z.B. der Dateiname.

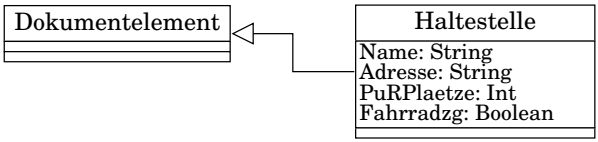
Analyseklasse "Dokumentelement": Diese Klasse modelliert alle selbständig erzeugbaren bzw. löschbaren Bestandteile der Dokumente, namentlich die Knoten und Kanten des abstrakten Graphen. Überlappende Dokumente betrachten wir hier nicht, d.h. jedes Dokumentelement ist Bestandteil von genau einem Dokument.

Analyseklasse "Diagramm" und Beziehungstyp "gehört_zu_dokument": Jedes Diagramm gehört zu genau einem Dokument. Typische Attribute sind: der Name des Diagramms, Autoren und administrative Daten.

Beziehungstyp "in_Diagramm": Eine Beziehung dieses Typs repräsentiert den Sachverhalt, daß das Dokumentelement in dem Diagramm vorkommt. Zugeordnete Attribute sind vor allem die Layoutdaten dieses Dokumentelements in diesem Diagramm.

6.2 Modellierung graphischer Knotentypen

Als Grundregel wird jedem graphischen Knotentyp eine Analyseklasse zugeordnet, die alle Knoten dieses Typs modelliert. Das folgende Bild zeigt diese Klasse am Beispiel der Haltestellen.



Analyseklasse "*graphischerKnotentyp*": Im allgemeinen ist diese Klasse wie folgt zu gestalten:

- Die Bezeichnung der Analyseklasse sollte passend aus der Begriffswelt der Anwender gewählt werden, z.B. “Person”, “Stadt” oder “Haltestelle”.
- Die Klasse ist Subtyp von Dokumentelement und erbt daher die Beziehungstypen, die zu den Klassen Dokument und Diagramm führen.
- Sofern die graphischen Knoten einen Namen bzw. eine Beschriftung (nicht notwendig eindeutig) haben, ist ein passendes Analyseattribut einzuplanen, z.B. “Familienname”, “Stadtname” usw.
- Weitere Analyseattribute können sich aus weiteren Beschriftungen ergeben, die in der graphischen Darstellung innerhalb oder direkt neben der Knotendarstellung stehen oder die aus sonstigen Quellen gewonnen werden. Beispiele: Name der Haltestelle, genaue Adresse, P+R-Plätze, Fahrradzugang, Behindertenzugang, Öffnungszeiten usw.

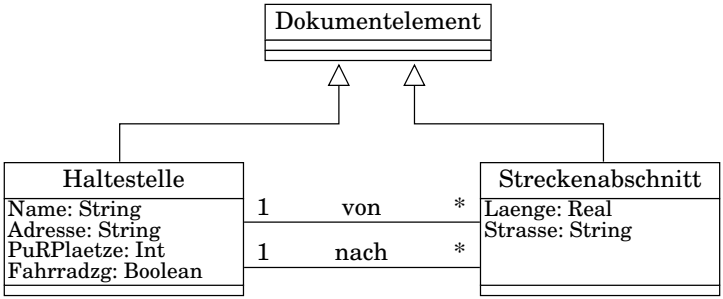
6.3 Modellierung graphischer Kantentypen

Die Modellierung graphischer Kantentypen (also Arten von Verbindungslinien) ist deutlich komplizierter als die Modellierung graphischer Knotentypen. Wie schon bei dem Busplan-Beispiel erwähnt ist es ungünstig, graphische Kantentypen als Beziehungstypen zu modellieren, selbst dann, wenn sie keine Attribute haben, weil sie u.a. eine Rolle in den “in_Diagramm”-Beziehungen spielen und hierüber die Layoutdaten zugeordnet bekommen. Ein graphischer Kantentyp muß daher grundsätzlich analog zum Analysemuster Koordinator modelliert werden durch

1. eine Analyseklasse “*graphischerKantentyp*” (s.u.)
2. ein bzw. zwei zugehörige Beziehungstypen, die die Kantenenden modellieren

Analyseklasse “*graphischerKantentyp*”: Der Name ist wieder passend zur Begriffswelt der Anwender zu wählen, bspw. “Streckenabschnitt”, “Mutter_von” usw. Sofern die graphischen Kantentypen

inhaltlich relevante Merkmale haben, z.B. Beschriftungen, sind geeignete Attribute einzuplanen[2]. Das folgende Klassendiagramm zeigt ein Beispiel.



6.3.1 Modellierung von Enden graphischer Kanten

Aufgrund der vorstehenden Regel wird eine abstrakte Kante durch ein Objekt repräsentiert. Ein solches Objekt muß mit den Objekten, die die beiden verbundenen abstrakten Knoten repräsentieren, in Beziehung gesetzt werden. Hierfür sind zwei Ansätze denkbar:

- pro Kantenende je ein **separater Beziehungstyp**
- ein **gemeinsamer Beziehungstyp** für alle Kantenenden

a) Separate Beziehungstypen pro Kantenende. Das vorstehende Klassendiagramm zeigt die Lösung gemäß dem ersten Ansatz: die beiden separaten Beziehungstypen “von” und “nach” geben die beiden Haltestellen an, die ein Streckenabschnitt verbindet. Die Multiplizitäten beider Beziehungstypen sind 1 und *. Allgemein gesagt führen Instanzen dieser Beziehungstypen zu den abstrakten Knoten, die am jeweiligen Ende der Kante erreicht werden. Die Bezeichnungen der beiden Enden einer Kante sind hier sozusagen im Namen der Beziehungstypen codiert.

[2]Beschriftungen, die sich auf die Kante als ganze beziehen, stehen meist nahe der Mitte der Kante. Diese Beschriftungen sind zu trennen von Beschriftungen, die an einem Ende der Kante stehen, auf diese gehen wir anschließend ein.

Die Namenswahl im obigen Beispiel suggeriert, daß eine Buslinie eine Route nur in einer Richtung befährt[3], also ein *gerichteter abstrakter Kantentyp* vorliegt. Der Ansatz kann aber auch bei ungerichteten abstrakten Kantentypen verwendet werden. In diesem Fall wird man die Namen der Beziehungstypen typischerweise identisch bis auf einen Zähler wählen, z.B. "stop1" und "stop2". Im Endeffekt ist nur durch Interpretation der Namen der Beziehungstypen bestimmbar, ob dieser Kantentyp gerichtet ist oder nicht und in welche Richtung die Kanten ggf. laufen.

b) Gemeinsamer Beziehungstyp für alle Kantenenden. Alternativ kann ein einziger Beziehungstyp für alle Kantenenden geplant werden. Sofern die Kantenenden identifizierbar sein müssen - bei gerichteten abstrakten Kanten ist das immer der Fall, auch sonst ziemlich häufig -, muß diese Identifizierung in einem Attribut der Beziehungen als Datenwert angegeben werden.

Im vorstehenden Beispiel würde man die Beziehungstypen "von" und "nach" ersetzen durch den Beziehungstyp "stoppt_an". Dieser hat:

- Multiplizitäten 2 und *
- eine assoziative Klasse, die ein Attribut hat, das die Kantenenden identifiziert. In unserem Beispiel wählen wir ein Text-Attribut namens "Rolle", das nur die Werte "Start" und "Ziel" enthalten darf.

6.3.2 Attributierte Enden graphischer Kanten

Als Beispiel erweitern wir unseren bisherigen Busplan um Informationen zu den Bus- bzw. Bahnsteigen der Haltestelle. In der graphischen Darstellung soll am Ende jeder graphischen Kante ein Kürzel für den Bussteig stehen, das angibt, an welchem Bussteig diese Buslinie anhält, wenn sie über diesen Streckenabschnitt in die Haltestelle einfährt.

Daß man diese Angabe am Ende der graphischen Kante notiert, ist kein Zufall: diese Angabe ist kein Merkmal der Kante als solcher, son-

[3] Für die Rückfahrt, sofern vorhanden, sollte hier eine andere Nummer gewählt werden.

dern ein Merkmal des Sachverhalts, daß diese Kante in diesem Knoten endet[4].

Unsere bisherigen Modellierungsregeln erweitern wir daher wie folgt: alle Beziehungstypen, die attributierte Kantenenden modellieren, sind um eine passende assoziative Klasse zu ergänzen, die diese Attribute aufnimmt. In unserem obigen Analyseklassendiagramm würde man daher den Beziehungstyp "nach" um eine assoziative Klasse ergänzen; diese würde das Attribut "Bussteig" haben.

Analog können wir im Fall eines gemeinsamen Beziehungstyps vorgehen, z.B. den Beziehungstyp "stoppt_an" ebenfalls um eine assoziative Klasse ergänzen bzw. die dort meist ohnehin vorhandene assoziative Klasse um diese Attribute ergänzen.

6.3.3 Separate vs. gemeinsame Beziehungstypen für Kantenenden

Beide Modellierungsmethoden führen zu korrekten Modellen, die insofern auch äquivalent sind.

Wenn man allerdings die Standard-Methoden zur Umsetzung von Analysemodellen in Entwurfsmodelle anwendet, kommt man zu recht unterschiedlichen Ergebnissen:

- Die separaten Beziehungstypen haben Multiplizität 1:n und können, sofern sie nicht attributiert sind, sehr effizient implementiert werden, ggf. auch dann, wenn sie wenige Attribute haben.
- Ein gemeinsamer Beziehungstyp hat Multiplizität 2:n und muß zunächst wie ein m:n-Beziehungstyp behandelt werden, führt also zu relativ aufwendigen Implementierungen (z.B. als Beziehungstabelle). Wenn die Kantenenden allerdings attributiert sind, sind die aufwendigen Implementierungen ohnehin naheliegend.

Im Endeffekt ist vor allem relevant, ob die Kantenenden attributiert

[4] Dies wird noch deutlicher, wenn man die Streckenabschnitte als ungerichtet ansieht, also als Darstellung, daß die Busse in beiden Richtungen fahren. Dann muß an *beiden Enden* jeweils eine Angabe stehen, an welchem Bussteig eine Buslinie in der Haltestelle anhält.

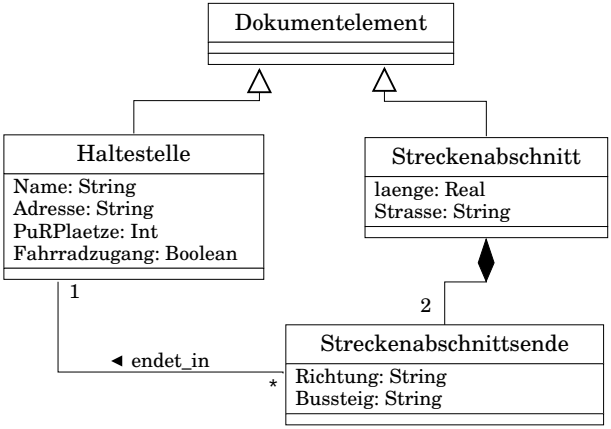
sind: falls ja, sollte ein gemeinsamer attributierter Beziehungstyp benutzt werden, falls nein, separate 1:n-Beziehungstypen pro Kantenende.

Softwaretechnische Beurteilung. Bei den vorstehenden Überlegungen haben wir genau genommen Problemstellungen der Entwurfsphase herangezogen, um über die Gestaltung von Analyse-Dokumenten zu entscheiden. Vordergründig ist das ein Verstoß gegen das Prinzip, bei der Analyse keine Entwurfsentscheidungen vorwegzunehmen. Allerdings ist dieses Prinzip in der Praxis ohnehin nicht strikt durchzuhalten, bei der Analyse werden immer erste als wesentlich erachtete Entwurfsentscheidungen getroffen. Wenn man sich dessen sehr präzise bewußt ist, ist auch nichts dagegen einzuwenden, Analysemodelle zur Dokumentation von Entwurfsentscheidungen zu nutzen.

Sofern man keine Entwurfsentscheidungen vorwegnehmen möchte, sei daran erinnert, daß beide Modellierungen äquivalent sind und daß man - zumindest manuell - aus beiden Varianten der Analysemodelle beide Varianten (und noch weitere) der Entwurfsdokumente "ableiten" kann. Diese Art der Ableitung ist aber eher ein kreativer Akt und nicht automatisierbar im Sinne der modellgetriebenen Softwareentwicklung.

6.3.4 Kantenendenobjekte

Die Beschriftungen an den Kantenenden führen in manchen Fällen zu einem weiteren Problem, wenn deren Position ebenfalls manuell einstellbar ist, z.B. unter oder über der Kante und mit einem gewünschten Abstand zum Knotensymbol. Dann müssen diesen Beschriftungen bzw. inhaltlich gesehen den Merkmalen der Kantenenden ebenfalls Layout-Daten zugeordnet werden. In diesem Fall greift erneut das Argument, das wir schon bei der Modellierung der graphischen Kanten benutzt haben: statt attributierter Beziehungen sollten besser Verbindungsobjekte benutzt werden. Diese Objekte sollten als Komponente des Objekts, das die graphische Kante repräsentiert, behandelt werden, denn sie können nur in diesem Kontext existieren.



Das vorstehende Klassendiagramm zeigt das Ergebnis dieser Modellierungsregel in unserem Beispiel. Hierbei haben wir einen ursprünglich vorhandenen gemeinsamen Beziehungstyp für alle Enden unterstellt, daher ist ein Attribut "Richtung" vorhanden. Es müssen genau 2 Streckenabschnittsenden pro Streckenabschnitt vorhanden sein, die in diesem Attribut unterschiedliche Werte haben.

6.3.5 Mehrfachkanten

In dem Beispiel-Busplan in Abschnitt 4 trat der Fall auf, daß mehrere Buslinien, die über den gleichen Streckenabschnitt zwischen zwei Haltestellen fahren, durch parallel laufende, verschieden gefärbte Linien angezeigt werden.

Wir haben diesen Sachverhalt bisher so interpretiert, daß in unserem abstrakten Graphen die beiden involvierten abstrakten Knoten durch je eine abstrakte Kante pro Linie verbunden werden.

Stattdessen kann man den Sachverhalt aber auch so interpretieren, unser abstrakter Graph nur genau eine Kante enthält, die man eher als die befahrene Straße interpretieren kann. Dieser Kante wird die *Liste* aller dort fahrenden Buslinien als Merkmal zugeordnet. Dieses Vorgehen ist besonders dann günstig, wenn alle graphischen Kanten ein gemeinsames Layout haben bzw. mit einem einzigen Layoutdatensatz auskommen. Problematisch an diesem Vorgehen ist das entstandene

komplexe Analyseattribut (das "BuslinienListe" heißen könnte) an der Klasse, die diese Gruppe von Linien modelliert.

6.4 Modellierung komplexer Merkmale

Die im vorigen Abschnitt diskutierten Mehrfachkanten sind nur ein Grund, warum komplexe Analyseattribute entstehen können. Komplexe Analyseattribute treten auch als Merkmale von Knoten auf, z.B.:

- die Liste der Bussteige in einer Haltestelle
- eine Liste der Umsteigemöglichkeiten in einer Bushaltestelle, die zugleich S-Bahnhof ist
- die Liste der Attribute einer Klasse in einem Klassendiagramm

Komplexe Attribute sind zwar im Prinzip in Analysemodellen zulässig, sind aber dennoch problematisch, wenn sie inhaltlich Referenzen auf andere Entitäten oder sogar Modellelemente enthalten. Dies ist z.B. bei dem oben erwähnten Analyseattribut "BuslinienListe" der Fall: die Buslinien sollten ebenfalls als eigene Entitäten modelliert sein, demnach wären hier Beziehungen angebracht und keine Werte, die als Referenzen auf Schlüsselattribute interpretiert werden müssen.

Komplexe Attribute sind auch dann unerwünscht, wenn ihre Struktur als wichtig für das Verständnis der gesamten Dokumentstrukturen angesehen wird; ihr Typ kann bestenfalls als (langer) Text oder in einem anderen Dokument spezifiziert sein, ist jedenfalls in einem Gesamtanalysemodell des Dokumenttyps schlecht erkennbar.

Häufig wird daher ein komplexes Attribut ersetzt durch eine Klasse, die als Container oder Anker einer Kollektion fungiert, und weitere Klassen, die die Komponenten modellieren. Hierbei können oft die Analysemuster Liste oder Gruppe angewandt werden.

Diese Komponenten werden vielfach als Dokumentelemente bezeichnet, speziell wenn sie einigermaßen groß sind und innerhalb der Kollektion angelegt oder gelöscht werden können. Dies aber im Sinne unserer grundlegenden Dokumentmodellierung nicht korrekt: da sie nicht graphisch dargestellt werden, sind ihnen keine Layoutdaten

zugeordnet. Sie können auch nicht unabhängig von dem echten Dokumentelement, das sie beschreiben, in ein Diagramm aufgenommen werden.

Relativ viel Konfusion entsteht nun dadurch, daß die Instanzen der Analyseklassen, die keinerlei graphische Repräsentation haben, ebenfalls als Repräsentanten von Knoten oder Kanten des abstrakten Graphen bezeichnet werden. Problematisch ist hier:

- Da diese Teile des abstrakten Graphen nicht graphisch dargestellt werden, fehlt das intuitive Verständnis. Die graphische Struktur wird immer unähnlicher zur Struktur des abstrakten Graphen, dies läuft der ursprünglichen Motivation, die sichtbare Dokumentstruktur als abstrakten Graphen darzustellen, völlig zuwider.
- Die Abbildung komplexer Attribute auf Netzwerke von Analyse-Objekten erfordert oft willkürliche Entscheidungen. So kann eine Liste von Entitäten wie eine echte Liste von verketteten Objekten modelliert werden oder alternativ als Verzeichnis, in dem die Einträge durchnumeriert sind. Anders gesagt werden hier abstrakte Graphen als "Technologie" zur Implementierung von Kollektionen ge- bzw. mißbraucht. Im Endeffekt herrscht nicht automatisch Konsens darüber, wie die Feinstrukturen zu modellieren sind, was also der "wirkliche" abstrakte Graph eines gegebenen Dokuments ist.

Literatur

[AMU] Kelter, U.: Lehrmodul "Analysemuster (Stichworte)"; 2010

[TAE] Kelter, U.: Lehrmodul "Transformation von Analyse-Datenmodellen in Entwurfsdokumente"; 2003

