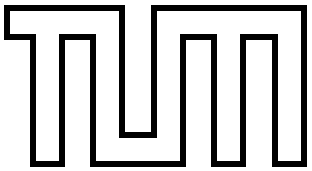


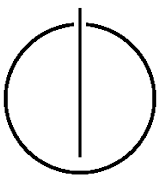
FAKULTÄT FÜR INFORMATIK

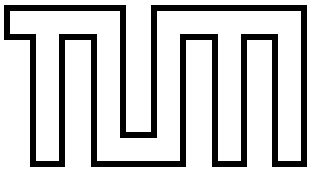
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Wirtschaftsinformatik

Erweiterte UML-Klassendiagramme zur Modellierung emergenter Informationsstrukturen - Analyse und prototypische Web-basierte Implementierung

Alexej Utz





FAKULTÄT FÜR INFORMATIK

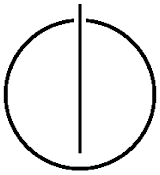
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Wirtschaftsinformatik

Erweiterte UML-Klassendiagramme zur Modellierung emergenter Informationsstrukturen - Analyse und prototypische Web-basierte Implementierung

Extended UML Class Diagrams for Modeling Emergent Information Structures - Analysis and Prototypical Web-based Implementation

Bearbeiter:	Alexej Utz
Aufgabensteller:	Prof. Dr. Florian Matthes
Betreuer:	Christian Neubert
Datum:	Dezember 14, 2011



Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 21. Februar 2012 Alexej Utz

Danksagung

Ich möchte mich bei Christian Neubert für eine sehr gute Betreuung und viele hilfreiche Ratschläge bedanken. Er trug durch konstruktive Hilfestellungen zu einer positiven Motivation bei der Erstellung dieser Bachelorarbeit bei.

Kurzbeschreibung

Die Bedeutung von Informationen in der modernen Welt ist schwer zu überschätzen. Sie werden als vollwertige Produktionsfaktoren in Unternehmen betrachtet und spielen vor allem in der unternehmerischen Praxis eine entscheidende Rolle für den Erfolg. Der Informationsfluss steigt dabei kontinuierlich an und wird durch die Verwendung von unterschiedlichen Medien und Kommunikationsmitteln immer heterogener.

Die Beherrschung von Informationen ist also keine einfache Aufgabe. Zur Bewältigung des wachsenden Informationsvolumens existieren viele Ansätze und Werkzeuge. Als vielversprechend für das Informations- und Wissensmanagement erwies sich der Enterprise-2.0 Ansatz. Er basiert auf den im Internet bereits allgegenwärtigen Web-2.0 Technologien. Ein Merkmal von Enterprise-2.0 liegt darin, dass die Benutzer ihre Informationen frei strukturieren können. Auf diese Weise entstehen emergente Strukturen – sie bilden sich aus den Benutzeraktionen heraus.

In dieser Bachelorarbeit wird die Modellierung von emergenten Informationsstrukturen am Beispiel der Hybrid Wikis analysiert und prototypisch umgesetzt. Eine Besonderheit von Hybrid Wikis liegt in der Vereinigung von zwei unterschiedlichen Modellierungskonzepten. Neben der expliziten Modelldefinition kann die Struktur implizit durch die Bearbeitung der Wiki-Seiten modelliert werden. Es wird eine Modellableitungsstrategie entwickelt, die die unterschiedlichen Konzepte berücksichtigt. Zur Darstellung des resultierenden Modells werden neue Visualisierungskonzepte entworfen und umgesetzt.

Abstract

It is hard to overestimate the importance of information in today's world. Information is seen as an important production factor and plays a crucial role for the business success. The volume and flow of the information rises continuously and is becoming more and more heterogeneous through the use of different media and communications.

Therefore, the management of information is a difficult and complex task. Many approaches and tools have been developed to cope with the huge amount of information. Enterprise 2.0 is known as highly promising approach for information and knowledge management. It is based on the ubiquitous Web 2.0 technologies. A special feature of the Enterprise 2.0 is that it doesn't impose on its users how the information should be structured or categorized. Instead Enterprise 2.0 provides tools that let the structure emerge from the user's actions.

Taking Hybrid Wikis as an example of a emergent structure, its modeling aspects are analyzed and a protype modelling tool is implemented in this bachelor thesis. A special feature of hybrid wiki is the combination of two different modeling approaches. Besides the explicit model definition, the structure can be modeled implicitly by the user's actions – by editing the wiki pages. A model derivation strategy that takes into account the different modeling concepts of Hybrid Wikis is developed. For the presentation of the resulting model new visualization concepts are designed and implemented.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die moderne Welt ist durch einen rapide steigenden Kommunikations- und Informationsfluss gekennzeichnet. Neben der Menge gewinnt auch die Rolle der Informationen kontinuierlich an Bedeutung. Um das riesige Informationsvolumen zu bewältigen und effektiv zu nutzen, werden laufend neue Werkzeuge entwickelt. Die klassischen Kommunikationsmitteln wie E-Mail, Instant Messaging oder Papier-Dokumente werden zwar eingesetzt, können jedoch das Wissensmanagement in einem modernen Unternehmen nicht oder nur unzureichend unterstützen. Die meisten zum Wissensmanagement verwendeten Kommunikationsmitteln fallen grundsätzlich in zwei Kategorien: Informationskanäle und Plattformen. Mit Kanälen wie z.B. E-Mail lassen sich Information erstellen und austauschen. Die Zugänglichkeit von solchen Informationen ist jedoch stark beschränkt. Dagegen zielen Plattformen wie Intranet oder Informationsportale auf eine breite Zugänglichkeit der Informationen ab. Die sogenannten Wissensmanagementsysteme versuchten beide Konzepte zu vereinen: das Expertenwissen aus den Informationskanälen zu extrahieren und an einer zentralen Stelle zu veröffentlichen. Der Erfolg dieser Systeme war allerdings begrenzt. Ein anderer Ansatz war erforderlich. Eine Lösung zur Erstellung, Austausch und Modifikation von Informationen bieten die im Internet bereits weit verbreiteten und anerkannten Web 2.0-Technologien - Blogs, Wikis und soziale Netze. Sie fanden den Einzug in die unternehmerische Praxis unter dem Titel Enterprise 2.0 [15].

1.1 Motivation

„Ein Sachverhalt ist denkbar, heißt: Wir können uns ein Bild von ihm machen."

Ludwig Wittgenstein

Ein grundlegendes Prinzip und gleichzeitig einer der Hauptunterschiede des Enterprise 2.0 von anderen Ansätzen ist die Möglichkeit für den Benutzer Informationen nach freiem Ermessen zu strukturieren. Es entstehen emergente Informationsstrukturen. Sie bilden sich aus den Benutzeraktionen heraus und sind deswegen an die Bedürfnisse und Gegebenheiten der Nutzer angepasst. Die meisten bestehenden Informationssysteme geben hingegen eine feste initiale Struktur vor, die von Benutzern eingehalten werden soll. Solche Strukturen sind in der Regel klar und übersichtlich. Ein Nachteil liegt jedoch in einer geringeren Akzeptanz unter den Benutzern [15].

Am Lehrstuhl für Software Engineering for Business Information Systems (sebis) an der Technischen Universität München wird eine Enterprise Collaboration-Plattform (Tricia) entwickelt, die neben den Standardfunktionen und Eigenschaften eines Enterprise 2.0-Systems über eine zusätzliche Funktionalität zur Strukturierung und Modellierung von Informationen verfügt - Hybrid Wikis. Diese ermöglichen es Inhalte nach bestimmten Regeln zu strukturieren, die vom Benutzer selbst definiert werden können[11][18].

Die so entstandenen Informationsstrukturen beinhalten eine hohe Dynamik und werden aufgrund des hohen Grads der Vernetzung schnell komplex und unübersichtlich. Tricia bietet derzeit keine Funktionalität zur Gesamtdarstellung der emergenten Struktur, die sich aus einem von Benutzern definiertem Schema sowie einem aus Daten abgeleiteten Modell zusammensetzt. Eine mögliche Lösung wäre ein aus beiden Konstrukten kombiniertes Modell, das die vorhandenen Strukturierungsregeln und innere Zusammenhänge beinhaltet. Zusätzlich kann eine passende Visualisierung die Kommunikation und Verständlichkeit des Modells verbessern.

Es existieren bereits Projekte, die sich mit der Visualisierung von Daten befassen (wie z.B. das Forschungsprojekt Pivot von Microsoft Labs). Auch gibt es Werkzeuge zur Konstruktion und Darstellung von standardisierten Modellen (UML), die Informationssysteme beschreiben und als ein Skelett für darauf aufbauende Datenstrukturen dienen. Im Falle von Hybrid Wikis muss aus einer bestehenden Datenbasis ein Modell abgeleitet werden, das zusammen mit dem fest definiertem Schema die Gesamtstruktur repräsentiert. Die Ableitung und kombinierte Visualisierung dieser Struktur stellt einen Spezialfall dar.

1.2 Ziele der Arbeit

Das erste Ziel dieser Arbeit ist es eine Strategie zur Ableitung eines Modells, das die emergente Datenstruktur in Hybrid Wikis repräsentiert, zu entwerfen und zu implementieren. Die Ableitung des Modells soll basierend auf den in Hybrid Wikis vorhandenen Strukturierungs- und Modellierungskonzepten erfolgen.

Das zweite Ziel der Arbeit ist die Konzeption und eine prototypische web-basierte Implementierung der Visualisierung für das abgeleitete Modell. Als Grundlage für die Visualisierung soll das UML-Klassendiagramm dienen, das um notwendige Hybrid Wikis-spezifische Elemente erweitert werden soll.

Das dritte Ziel der Arbeit ist die Integration der Modellableitung sowie der Visualisierung in die Collaboration-Plattform Tricia.

1.3 Aufbau der Arbeit

Kapitel 2 gibt eine detaillierte Einführung in die grundlegenden Technologien, die im Rahmen dieser Arbeit verwenden werden. Neben Hybrid Wikis selbst, auf dem diese Arbeit aufbaut, spielen Ecore aus dem Eclipse Modeling Framework und Unified Modeling Language (UML) wichtige Rollen und werden in diesem Kapitel ebenso erläutert. Ecore ist das Zielformat zur Repräsentation des Modells, das bei Modellableitung zum Einsatz kommt. Auf UML basiert die Visualisierung des abgeleiteten Modells.

Im Kapitel 3 wird der Zusammenhang zwischen den Strukturierungs- und Modellierungskonzepten von Hybrid Wikis und den Konstrukten eines UML-Klassendiagramms erläutert. Außerdem wird in diesem Kapitel der Entwurf der UML-Erweiterung beschrieben.

Das Kapitel 4 geht auf die Einzelheiten der Umsetzung der Ableitungsstrategie und der Visualisierung ein. Es wird u.a. deren Funktionsweise sowie Integration in Tricia beschrieben.

Die Ergebnisse der Arbeit werden im Kapitel 5 zusammengefasst. Neben den im Laufe der Arbeit aufgetretenen Problemen und Schwierigkeiten werden dort Ideen und Empfehlungen für die weitere Entwicklung der Modellierung und visuellen Darstellung emergenter Informationsstrukturen vorgestellt.

2 Grundlagen

Dieses Kapitel beschreibt die im Rahmen der Arbeit eingesetzten Basistechnologien. Obwohl eine Vielzahl unterschiedlicher Technologien (Software, Programmiersprachen usw.) verwendet wurde, beschränkt sich die Auswahl auf die wichtigsten drei, die die Arbeit maßgeblich beeinflussten. Es werden Hybrid Wikis, Unified Modeling Language und Eclipse Modeling Framework beschrieben.

2.1 Hybrid Wikis

Wikis werden vermehrt als Wissensspeicher im unternehmerischen Umfeld eingesetzt. Neben der Konsolidierung der Wissensbasis an einer zentralen Stelle bieten sie eine Suchfunktion und Verlinkung zusammenhängender Informationen, was erhebliche Vorteile gegenüber einer dezentralen heterogenen Infrastruktur für das Wissensmanagement aufweist. Außerdem steigt durch den Einsatz von Wikis die Motivation der Mitarbeiter ihr eigenes Wissen einzubringen. Der klassische Wiki-Ansatz weist jedoch eine Schwachstelle auf: bei steigender Informationsmenge reichen die klassischen Wiki-Konzepte wie Suchfunktion und Links für eine effiziente Such und Navigation nicht mehr aus. Eine Lösung für dieses Problem verspricht die Funktionalität von Semantic Wikis: die eingetragenen Daten werden in strukturierter Form gespeichert und können über Abfragen abgerufen werden. Die Benutzung von Semantic Wikis ist jedoch nicht intuitiv, da die Strukturierung von Inhalten eine spezielle Syntax verwendet [11].

2.1.1 Allgemein

Hybrid Wikis sind eine Erweiterung des Enterprise Collaboration-Plattforms Tricia. Sie vereint die Idee des klassischen Wikis mit den Strukturierungskonzepten von Semantic Wikis und zielt auf eine intuitive Benutzung und hohe Akzeptanz unter den Anwendern. Anstatt Inhalte mit semantischen Annotationen zu versehen, die separat definiert werden, wird dem Anwender vorgeschlagen die Semantik implizit durch Eingabe von Daten in tabellarischer Form zu bestimmen (Abbildung 2.1). Der Anwender kann nach Bedarf neue Eingabefelder hinzufügen. Ein Vorschlagssystem, das die bisherige Semantik berücksichtigt, erleichtert die Dateneingabe. Die Struktur entsteht also nach dem Pull-Prinzip - Anwender erkennen den eigenen Nutzen und die Vorteile der Strukturierung von Informationen und tragen so einem konsistentem Datenmodell bei [11].

Klassische Wikis bieten die Möglichkeit einer Wiki-Seite einen Name zuzuweisen, Verlinkungen auf andere Wiki-Seiten zu definieren sowie den Inhalt mithilfe eines WYSIWYG -Editors (”What You See Is What You Get“) zu gestalten. Hybrid Wikis enthalten neben der klassischen Wiki-Funktionalität einige Strukturierungs- und Modellierungskonzepte, die eine hohe Benutzerfreundlichkeit versprechen. Diese werden im Folgenden erklärt.

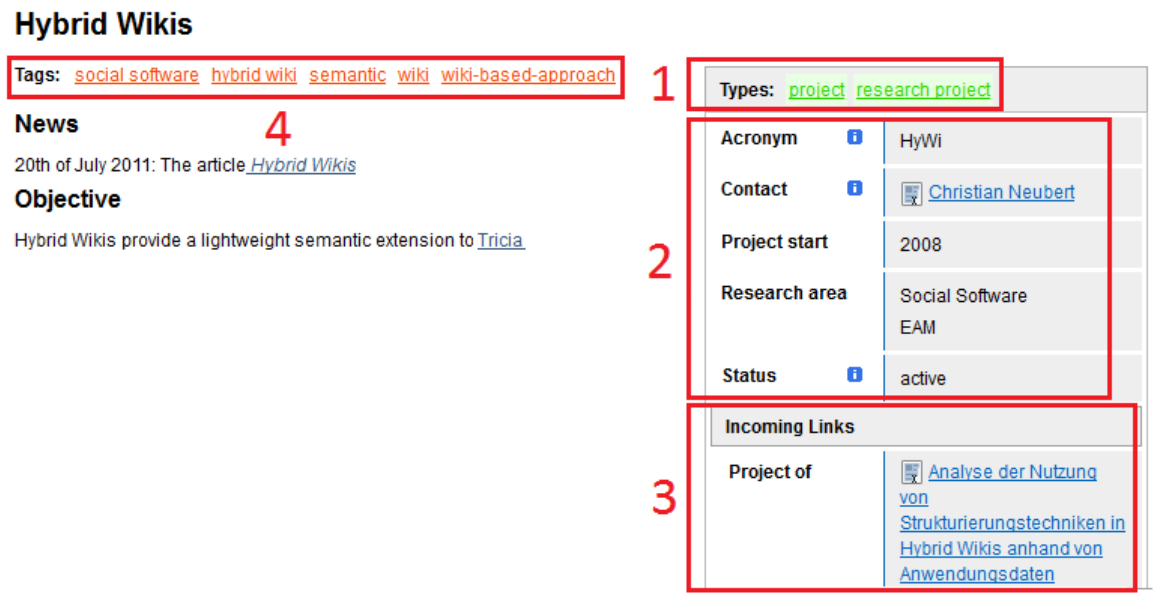


Abbildung 2.1: Modellierungskonzepte in Hybrid Wikis

2.1.2 Attribute

Attribute repräsentieren eine strukturierte Komponente der Wiki-Seite. Sie bestehen aus einem Attributnamen und einem Attributwert. Als Attributwerte eignen sich in Hybrid Wikis Zeichenketten, Dates, Verlinkungen auf andere Wiki-Seiten oder Dateien usw. Es sind mehrere Attributwerte (auch von unterschiedlichen Typen) zulässig. Attribute werden rechts oben auf der Wiki-Seite in alphabetischer Reihenfolge dargestellt (Punkt 2 in Abbildung 2.1).

Zur Vermeidung redundanter Eingaben werden die Link-Attribute auch auf der Zielseite im Abschnitt *incoming links* angezeigt (Punkt 3 in Abbildung 2.1). Der Attributname besteht aus dem ursprünglichen Namen mit dem Zusatz *of*. Als Attributwert ist der Link auf die Wiki-Seite mit dem ursprünglichen Attribut angegeben. „Incoming links“ stellt ein effektives Mittel zur Verbesserung der Benutzerfreundlichkeit und Navigation dar. Man sieht auf einen Blick alle Referenzen auf die betrachtete Wiki-Seite – die aufwendige Suche bleibt somit erspart.

Beim Betrachten einer Wiki-Seite werden dem Benutzer zusammen mit bereits vorhandenen Attributen Vorschläge angezeigt, die auf ähnlichen Seiten angegeben wurden (Punkt 1 in Abbildung 2.2). Ähnlichkeit der Wiki-Seiten ist über die Zuweisung von Tags (Punkt 4 in Abbildung 2.1) geregelt. Sie signalisieren einen Zusammenhang zwischen der Wiki-Seite und dem in Tag enthaltenem Thema. Eine besondere Art von Tags – Type Tags – beschreibt eine höhere Ähnlichkeit der Wiki-Seiten. Auf der Ebene der Type Tags können Attribute definiert werden, sodass Wiki-Seiten vom gleichem Typ (also mit gleichem Type Tag) die gleichen Attribute besitzen [11]. Type Tags werden im Folgenden vorgestellt.

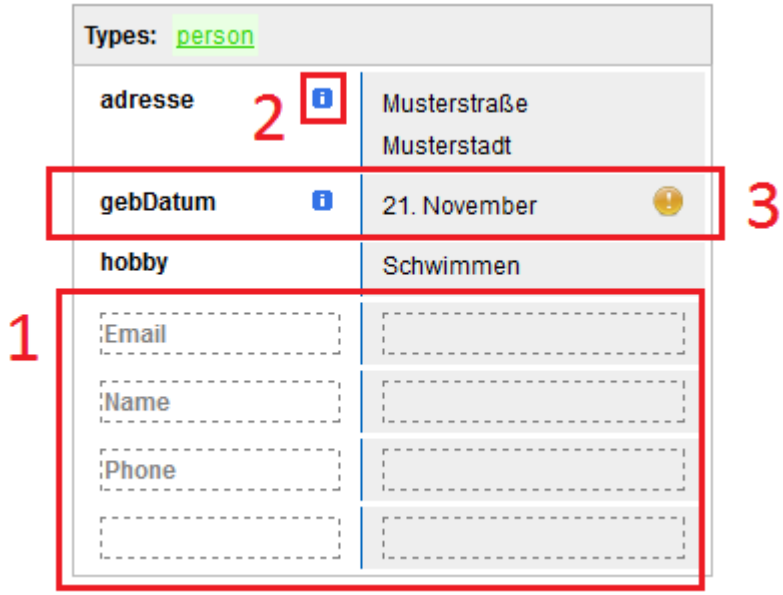


Abbildung 2.2: Attributvorschläge auf einer Wiki-Seite

2.1.3 Type Tags

Mit Type Tags weist der Anwender einer Wiki-Seite bzw. einem auf dieser Seite beschriebenen Objekt einen Typ zu. Somit besitzen alle Objekte von gleichen Typ idealerweise auch den gleichen Type Tag. Type Tags werden rechts oben auf den Wiki-Seiten über den Attributen dargestellt (Punkt 1 in Abbildung 2.1).

Beim Bearbeiten einer Wiki-Seite kann man dieser Seite einen bestehenden Type Tag zuweisen oder auch einen neuen anlegen. Nach der Auswahl eines existierenden Type Tags schlägt das System Attribute vor, die auf ähnlichen Seiten (also Seiten mit dem gleichen Type Tag) angegeben wurden. Enthält die bearbeitete Seite mehrere Typ Tags, so werden bei Vorschlägen die Attribute präferiert, die auf Wiki-Seiten mit möglichst vielen dieser Type Tags angegeben sind. Somit unterstützen die Type Tags das Erstellen und Editieren von Wiki-Seiten. Für erfahrene Benutzer besteht zusätzlich die Möglichkeit, auf der Ebene der Type Tags Attributdefinitionen mit Integritätsbedingungen festzulegen. Solche Attribute, die eine Attributdefinition besitzen, sind durch ein Zeichen rechts von dem Attributnamen gekennzeichnet (Punkt 2 in Abbildung 2.2). Die angegebenen Attributwerte werden mit Hilfe der definierten Restriktionen 2.1.4 überprüft [11].

Type Tags werden für Abfragen verwendet: möchte man z.B. eine Liste mit allem Wiki-Seiten von Typ *person*, so wird die Abfrage als einzige Bedingung den Type Tag *person* haben. In den Abfragen sind mehrere Type Tags und Attribute kombinierbar. Hybrid Wikis bieten keine Generalisierung für Type Tags an, sodass bei einem spezifischem Type Tag immer der allgemeinere anzugeben ist.

2.1.4 Integritätsbedingungen

Wie in 2.1.3 bereits erwähnt, können in Type Tags Attributdefinition deklariert werden. Sie legen und beschreiben Attribute fest, die Wiki-Seiten vom jeweiligen Typ besitzen sollen. Eine Attributdefinition (Abbildung 2.3) beinhaltet neben einem Attributnamen, dessen Eingabe obligatorisch ist, auch optionale Eingabefelder. Man kann den Typ sowie die Multiplizität für die Attributwerte bestimmen. Wählt man Link als Typ aus, so können zusätzlich die Type Tags für die Zielseite bestimmt werden. Soweit mehrere Type Tags angegeben sind, sollte die referenzierte Wiki-Seite nur einen davon besitzen - die angegebenen Type Tags sind disjunktiv verknüpft.

Wenn eine Attributdefinition als *Strict* gekennzeichnet ist, können nur korrekte also dem definierten Typ und der Multiplizität entsprechende Eingaben gespeichert werden. Ist die Attributdefinition nicht *Strict*, so wird lediglich eine Warnmeldung angezeigt: Punkt 3 in Abbildung 2.2 zeigt ein Attribut, das mit dem Text „21. November“ instanziiert wurde. Die dazugehörige Attributdefinition *gebDatum* besitzt jedoch Date als Typ. Somit ist die Eingabe fehlerhaft, was ein Ausrufezeichen rechts von dem Attributwert signalisiert. Ändert man eine Attributdefinition, die vorher nicht *Strict* war, auf *Strict*, so werden die bestehenden inkorrekten Eingaben beibehalten. Somit ist eine Validation der Eingabe mit Hilfe der sich mit der Zeit ändernden Regeln möglich.

2.1.5 Datenmodellierung

Mit Type Tags und Attributdefinitionen wurden bereits zwei explizite Modellierungsinstrumente in Hybrid Wikis betrachtet. Doch Hybrid Wikis verfügen auch über Werkzeuge, die implizit zur Erstellung einer Struktur beitragen. In diesem Abschnitt werden die beiden Ansätze voneinander abgegrenzt und im Zusammenhang mit der Datenmodellierung kurz beschrieben.

Die Datenmodellierung beschäftigt sich mit der Beschreibung der Struktur der Datenbasis von Informationssystemen. Sie betrachtet im Allgemeinen 3 Konzepte, die zusammen ein Datenmodell bilden [16]:

1. Objekttypen um existierende Objekte im System zu abstrahieren
2. Eigenschaften der Objekttypen – Attribute
3. Beziehungen zwischen den Objekttypen

Hybrid Wikis besitzen Repräsentationen für diese Elemente:

- Type Tags repräsentieren die Objekttypen und beschreiben dabei Wiki-Seiten, die als Objekte im System agieren.
- Attribute eines Datenmodells, die die Objekttypen charakterisieren, werden in Hybrid Wikis durch Attributdefinition in Type Tags und instanziierte Attribute auf den Wiki-Seiten symbolisiert.
- Die Rolle der Beziehungen unter den Objekttypen spielen in Hybrid Wikis die Link-Attribute.

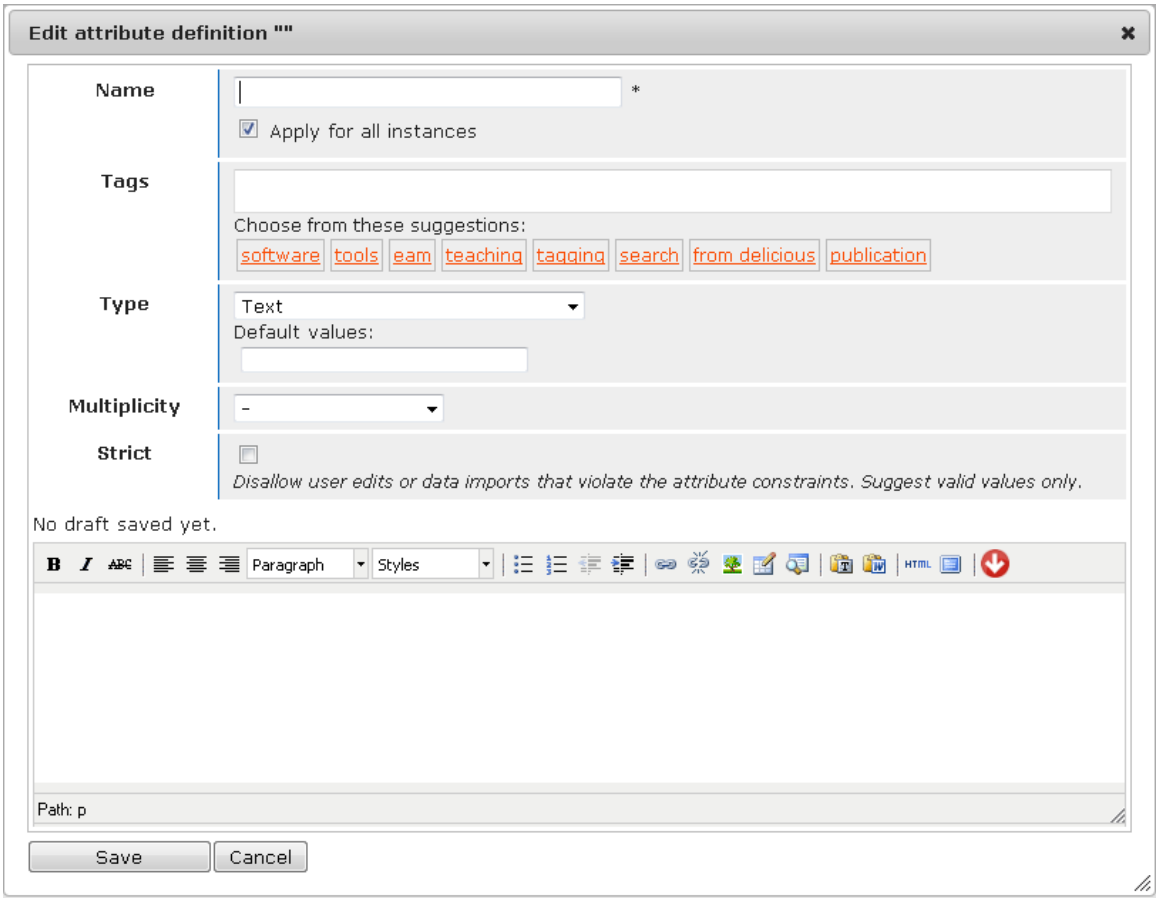


Abbildung 2.3: Bearbeitung einer Attributdefinition

Mit den o.g. Modellierungselementen bieten Hybird Wikis zwei unterschiedliche Ansätze zur Datenmodellierung [14].:

- Bottom-up: die Struktur entsteht implizit durch die Bearbeitung von Wiki-Seiten, Zuweisung von Type Tags, Instanziierung von Attributen auf den Wiki-Seiten. Das Datenmodell wird aus der entstandenen Datenbasis abgeleitet.
- Top-down: mit Hilfe von Type Tags und Attributdefinitionen wird ein Datenmodell spezifiziert. Die Modellierung kann auch ohne bestehende Wiki-Seiten erfolgen. Das Datenmodell wird explizit definiert.

2.1.6 Zusammenfassung

Hybrid Wikis unterstützen die Anwender mit Vorschlägen beim Erstellen und Editieren von Wiki-Seiten. Die Vorschläge werden aus expliziten Attributdefinitionen und aus bestehenden Benutzereingaben automatisch generiert. Für eine Attributdefinition können neben dem Namen u.a. der Typ und die Multiplizität angegeben werden. Außerdem kann ein Attributwert falsch oder korrekt sein, solange es eine dazugehörige Attributdefinition existiert. Die Type Tags mit Attributdefinitionen bilden zusammen mit den Attributen aus den Benutzereingaben eine Struktur, dem die bestehende Datenbasis zugrunde liegt.

2.2 UML-Klassendiagramme

Die Unified Modeling Language (UML) ist eine Modellierungssprache mit einem sehr breitem Anwendungsspektrum. Sie bildet eine Familie grafischer Notationen, die für Dokumentation, Visualisierung, Spezifikation und Entwurf von (hauptsächlich) Informationssystemen geschaffen wurde. UML entstand 1997 durch die lang erwartete Vereinheitlichung der vielen bis dahin existierenden grafischen Modellierungssprachen. Die Grundidee von UML ist, dass ein komplexes System aus verschiedenen Sichten am besten beschrieben werden kann. Eine einzige Sicht kann nicht alle bedeutende Aspekte erfassen. Aus diesem Grund bietet die Sprache in der letzten Version (2.4.1) 14 Diagrammtypen. Die mit Abstand am häufigsten verwendete Modellierungstechnik bilden die Klassendiagramme. Klassendiagramme werden oft mit UML-Diagrammen gleichgesetzt. Neben ihrer Allgegenwärtigkeit beinhaltet sie die breiteste Palette der Modellierungskonzepte aus UML [12].[3].

Klassendiagramme sind Strukturdiagramme. Sie beschreiben die Objekttypen im System und die statischen Beziehungen unter ihnen. Neben den strukturellen Zusammenhängen eines Systems zeigen Klassendiagramme die Eigenschaften und Operationen der Klassen [12]. Klassendiagramme beinhalten eine große Anzahl von Modellierungselementen. Viele davon werden selten und lediglich in Spezialfällen benutzt. Im Folgenden werden nur die wichtigsten Elemente beschrieben, die im Rahmen dieser Arbeit relevant sind.

2.2.1 Klassen

Die primäre Aufgabe der Klassen ist die Gruppierung und Kapselung von Attributen und Methoden zur einer Einheit. Klassen werden als Rechtecke mit drei Abschnitten – für den

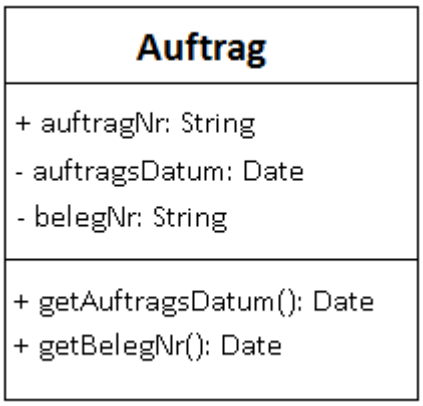


Abbildung 2.4: eine Klasse aus dem UML-Klassendiagramm

Klassennamen, für die Attribute und Methoden – dargestellt (Abbildung 2.4). Der mittlere Abschnitt enthält eine Liste mit den für die Klasse definierten Attributen.

Attribute sind die Zustandskomponenten einer Klasse. Instanzen einer Klasse unterscheiden sich durch die verschiedenen Ausprägungen ihrer Attribute. Attributdefinitionen bestehen (ähnlich wie bei Type Tags) aus einem Namen und Typ, der einschränkt welche Art von Objekten als Attributwert in der konkreten Ausprägung zulässig ist. Des Weiteren werden Attribute durch eine Multiplizität spezifiziert. Sie beschreibt den quantitativen Zusammenhang zwischen der Ausprägung eines Attributes und der entsprechenden Instanz einer Klasse. Die Multiplizität sagt aus, wie viele Attributwerte ein Objekt enthalten darf. Da das Konzept Klasse aus der objektorientierten Programmierparadigma übernommen wurde, enthalten Klassendiagramme auch rein implementierungstechnische Elemente (obwohl alle o.g. Konstrukte ebenso zur Implementierungsbeschreibung verwendet werden können). So besitzen Attribute und Methoden z.B. eine Sichtbarkeit. Sie zeigt an, ob das entsprechende Merkmal privat oder öffentlich ist , ob man also darauf von Außerhalb zugreifen kann [12][17].

Die Methoden (oder Operationen) werden im unteren Abschnitt dargestellt. Sie beschreiben die Funktionalität einer Klasse, welche Aktivitäten diese ausführen kann. Zusätzlich zum Namen und Sichtbarkeit beinhaltet die Beschreibung der Methode einen Rückgabetyp sowie eine Parameterliste. Die Beschreibung der Methode im Klassendiagramm gibt im Wesentlichen ihre Signatur wieder. Methoden werden im Rahmen dieser Arbeit nicht weiter betrachtet und wurden nur wegen der Vollständigkeit erwähnt.

2.2.2 Assoziationen

Eine spezielle Form der Attribute sind die Assoziationen. Sie beschreiben binäre Beziehungen zwischen den Klassen in einem System. In Klassendiagrammen werden Assoziationen als Linien zwischen zwei Klassen dargestellt. Sie beinhalten fast die gleichen Eigenschaften wie die Attribute. ung 5 zeigt dieselbe Information in zwei unterschiedlichen Notationen: als Attribute und Assoziationen [12]. Zusätzlich zu den von Attributen bekannten Merkmalen besitzen Assoziationen Rollennamen und eine Beschreibung der möglichen

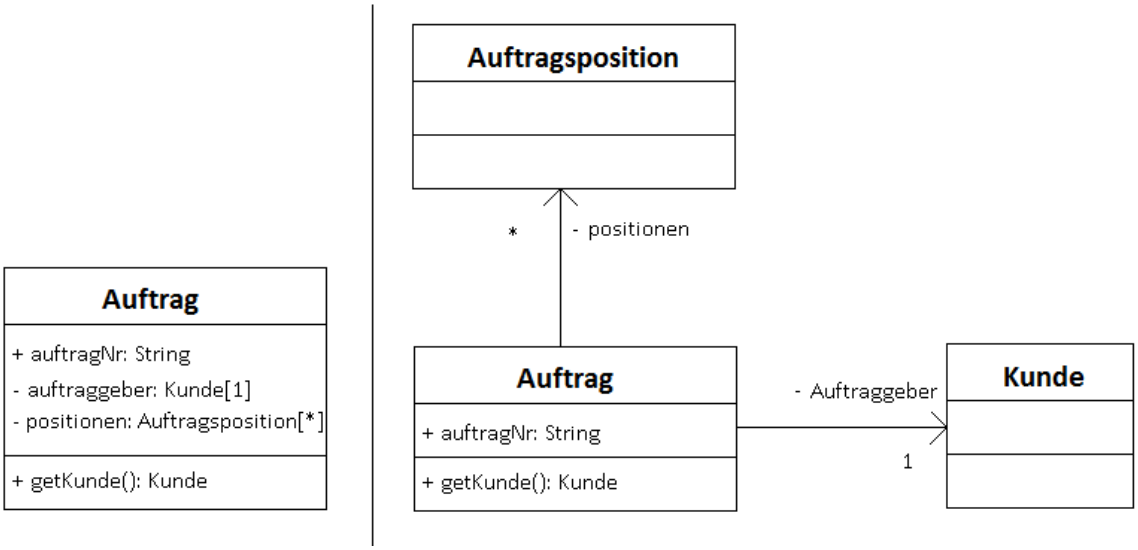


Abbildung 2.5: Darstellung von gleichen Informationen mit Attributen und Assoziationen

Navigationsrichtungen. Als Rollenname kann auch der Assoziationsname angenommen werden, wenn dieser die Beziehung zwischen den Klassen eindeutig beschreibt. Es existieren gerichtete und ungerichtete Assoziationen. Gerichtete Beziehung bedeuten, dass die Navigation nur in eine Richtung möglich ist. Sie werden mit einem Navigationspfeil zur Zielklasse gekennzeichnet. Ungerichtete oder bidirektionale Assoziationen erlauben eine Navigation in beide Richtungen. Diese sollen mit jeweils zwei Pfeilen verdeutlicht werden [17][12].

Zwei besondere Formen der Assoziation bilden die Komposition und Aggregation. Eine Aggregation beschreibt eine “Ist-ein-Teil-von”-Beziehung und wird mit einer nicht ausgefüllten Raute dargestellt. Komposition erweitert das Konzept der Aggregation um die Existenzabhängigkeit – einzelne Teile können nicht ohne das Ganze existieren. Sie wird mit einer ausgefüllten Raute dargestellt [12].

Zur hierarchischen Strukturierung von Klassen wird die Generalisierung verwendet. Existieren mehrere Klassen mit teilweise gleichen Eigenschaften (gleiche Attribute, Methoden), so können die Übereinstimmungen der verschiedenen Klassen in einer allgemeinen Klasse erfasst werden. [17]. In Abbildung 2.6 besitzen die Klassen Kundenauftrag und Fertigungsauftrag zwei gemeinsame Attribute: auftraNr:String und autragsDatum:Date. Diese sind in der Superklasse Auftrag definiert. Kundenauftrag und Fertigungsauftrag sind die Unterklassen von Auftrag. Sie erben die Eigenschaften von der Superklasse. Für die Vererbung gilt das Prinzip der Ersetzbarkeit: an jeder Stelle, wo die Verwendung von einer Superklasse vorausgesetzt ist, müssen auch alle ihre Unterklassen verwendbar sein. Konkret: besteht z.B. eine Assoziation zur Klasse Auftrag, so können anstelle der Klasse Auftrag die beiden Unterklassen Kunden- und Fertigungsauftrag verwendet werden.

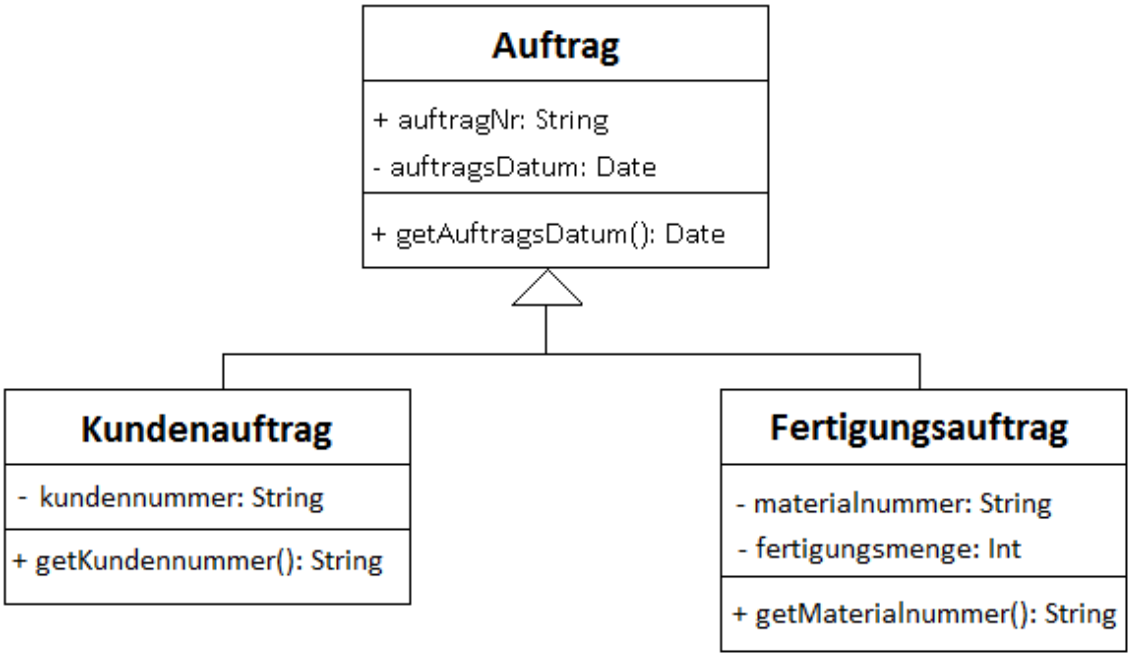


Abbildung 2.6: Generalisierung in UML

2.2.3 Zusammenfassung

UML-Klassendiagramme bilden ein funktionsmächtiges Modellierungsinstrument. Es enthält ein breites Spektrum an Modellierungskonzepten, die zur Beschreibung von strukturellen Zusammenhängen benutzt werden. Eine gleichartige Aufgabe verfolgen in Hybrid Wikis die Type Tags und Attribute. Da Klassen mit Attributen in UML und Type Tags mit Attributdefinitionen in Hybrid Wikis ähnliche Konstrukte mit äquivalenten Funktionen repräsentieren, wurde das UML-Klassendiagramm als Basis für die Visualisierung des aus Hybrid Wikis abgeleiteten Modells ausgewählt.

2.3 Ecore

Eclipse Modeling Framework (EMF) ist ein Open-Source-Framework zur Codegenerierung und Modellierung auf Basis von Eclipse. EMF bietet die Möglichkeit ein Modell mit XML (Extensible Markup Language), UML oder mit annotiertem Java-Code zu erstellen und daraus die beiden verbliebenen Varianten gemäß dem Modell automatisch zu generieren. Um Modelle in EMF erstellen zu können, wird eine allgemeine Terminologie benötigt, die diese beschreiben kann. Eine Sprache oder ein Modell, in der die Modellierungselemente definiert sind, ist ein Metamodell. EMF verwendet für diesen Zweck Ecore [7].

Ecore ist also ein Metamodell, das die EMF-Modelle repräsentiert. Da EMF sich mit UML-Klassendiagrammen befasst, gibt Ecore im Grunde das Metamodell vom UML-Klassendiagramm wieder. Abbildung 2.7 zeigt die wichtigsten Elemente – den Kern – von

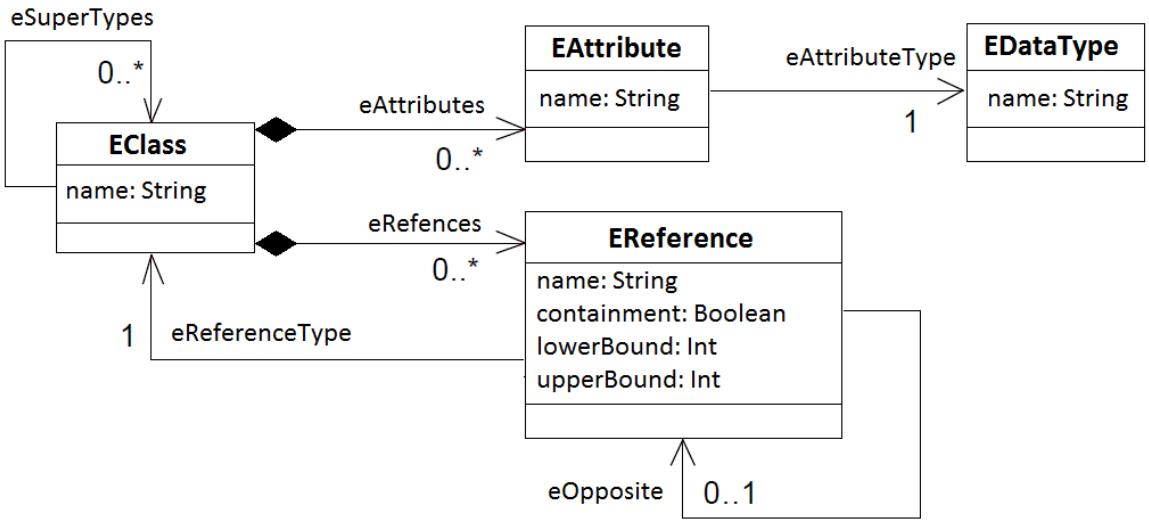


Abbildung 2.7: Die Datenstruktur von Ecore

Ecore. Dieses Modell definiert 4 Typen von Objekten, die in einem Klassendiagramm verwendet werden [7].:

1. EClass modelliert Klassen. Klassen besitzen einen Namen und bestehen (Komposition-Beziehung) aus Attributen und Assoziationen (EReference). Generalisierung (oder Vererbung) wird im Form einer gerichteten Assoziation zu Superklassen (Rollenname eSuperTypes) definiert.

2. EAttribute modelliert Attribute. Diese werden mit einem Namen identifiziert und besitzen einen Typ (EdataType).

3. EDataType repräsentiert einfache Datentypen, die nicht als Klassen im aktuellen Klassendiagramm modelliert werden. Damit werden primitive oder in Java bereits vorhandene Typen modelliert.

4. EReference modelliert Assoziationen zwischen den Klassen. Sie beschreibt eine Seite der Assoziation. Wenn die Assoziation ungerichtet ist, dann wird sie mit zwei Objekten von Typ EReference, die miteinander in Beziehung stehen (eOpposite), instanziiert.

Abbildung 2.7 zeigt eine stark vereinfachte Sicht auf Ecore. So sind z.B. EAttribute und EReference Unterklassen von EStructuralFeature, EClass und EDataType – von EClassifier. Außerdem fehlt im Diagramm eine Repräsentation für Methoden einer Klasse. Diese werden aber hier nicht weiter besprochen.

Wichtig zu erwähnen sind noch die Annotationen (Klasse EAnnotation in Ecore). Mit deren Hilfe kann man an alle Objekte eines Klassendiagramms zusätzliche Informationen anhängen. Alle Elemente können keine oder mehrere Annotationen besitzen. Eine besondere Form der Annotationen bilden die Extended Metadata-Annotationen. Sie werden in

der Regel in Modellen verwendet, die aus XML-Schema generiert wurden. Extended Metadata beinhaltet die Informationen aus dem XML-Schema wie z.B. XML-Namensräume. Diese Informationen haben aber keine Repräsentationen in Ecore und werden im Klassendiagramm nicht dargestellt [7].

Ecore ist eine Sprache, die Modellierungskonzepte im EMF beschreibt und sich in Modellierungsfunktionalität an die UML-Klassendiagramme stark orientiert. Da allen Modellierungstechniken in EMF das Ecore-Metamodell zugrunde liegt, kann ein Klassendiagramm direkt mit Ecore (mit Java-Code) modelliert werden. Deswegen spielt Ecore als Zielformat für die Modellrepräsentation bei der Umsetzung der Modellableitung aus Hybrid Wikis eine wichtige Rolle.

3 Konzeption des Modells und der visuellen Darstellung

Als Grundlage für die Modellableitung und -visualisierung stellen Hybrid Wikis folgende Informationen zur Verfügung: explizite Modellierungskonzepte wie Type Tags mit Attributdefinitionen sowie die von Benutzern eingegebenen Daten, die die Gesamtstruktur implizit beeinflussen. Außerdem lassen sich aus Hybrid Wikis einige Kennzahlen ausrechnen, die für das Modell interessant sind. Es steht also eine Menge an Informationen zur Verfügung. Welche davon im Modell in den Einsatz kommen und wie sie aus Hybrid Wikis abgeleitet werden, beschreibt das Kapitel 3.1.

Wie bereits erwähnt, bilden die Type Tags in Hybrid Wikis und Klassen im UML-Klassendiagramm ähnliche konzeptuelle Einheiten. Außerdem werden in beiden Konstrukten gleichbedeutende Eigenschaften definiert – Attribute. Wie Type Tags und Attribute aus Hybrid Wikis auf die Modellierungskonzepte des Klassendiagramms abgebildet werden, beschreibt das Kapitel 3.2. Im Abschnitt 3.3 wird anhand eines Beispiels die Modellableitung demonstriert.

Hybrid Wikis beinhalten eine Menge an Informationen, die für das abgeleitete Modell zwar eine Rolle spielen, jedoch keine Repräsentation in UML-Klassendiagrammen besitzen. Um diese Informationen darstellen zu können, bedarf es der Erweiterung von existierenden Modellierungselementen. Kapitel 3.4 beinhaltet den Entwurfsprozess neuer Elemente. Viele Informationen können außerdem durch die Positionierung der Modellelemente kommuniziert werden. Das Layout wurde für die Visualisierung des Modells neu entworfen. Die Konzeption ist im Kapitel 3.5 enthalten.

3.1 Inhalt des Modells

Ein System kann aus mehreren Sichten beschrieben werden, die unterschiedliche Aspekte berücksichtigen. Für die Modellierung von Daten ist die Datensicht zuständig. Sie betrachtet die Struktur der Datenbasis eines Systems, die sich aus der Zusammenwirkung ihrer Elemente herausbildet. Im Falle von Hybrid Wikis handelt es sich um eine emergente Struktur – sie entwickelt sich durch neue Definitionen und Benutzereingaben ständig weiter. Das zu entwickelnde Modell repräsentiert also den aktuellen Zustand der Datenstruktur. Da Hybrid Wikis ein neues Konzept darstellen, sind neben den bekannten Modellierungselementen (Siehe 2.1.5) zusätzliche Informationen erforderlich, um die resultierende Datenstruktur zu beschreiben [16].

3.1.1 Strukturelle Information

Wie bereits erwähnt, besteht ein Datenmodell aus Objekttypen, Attributen und Beziehungen. Diese Konzepte besitzen Repräsentationen in Hybrid Wikis (Siehe 2.1.5).

Die Modellbildung in Hybrid Wikis basiert grundsätzlich auf zwei Ansätzen: Bottom-up und Top-down (Siehe 2.1.5). Somit stammen die strukturellen Informationen für die Modellableitung entweder aus den Typ- und Attributdefinition (Top-down) oder aus der Datenbasis (Datenbasis = Benutzereingaben = Attribute auf den Wiki-Seiten) – Bottom-up-Ansatz. Um das Gesamtmodell zu bilden, werden die nach den unterschiedlichen Ansätzen modellierten Informationen kombiniert. Im ersten Schritt der Modellableitung wird ein Modell auf Basis von Typ- und Attributdefinitionen extrahiert. Als nächstes wird ein Modell aus der Datenbasis abgeleitet. Anschließend werden die zwei Modelle gegenübergestellt um Zusammenhänge zu identifizieren und Informationen für das Gesamtmodell aufzubereiten.

Typen

Das abgeleitete Gesamtmodell beinhaltet Typen, um existierende Objekte mit Attributen zu beschreiben. Als Typen im Modell dienen die Type Tags aus Hybrid Wikis. Sie können Attributdefinitionen besitzen, die aus einem Attributtyp, einer Multiplizität sowie einer *Strict*-Eigenschaft bestehen (Siehe 2.1.4). Die Attributdefinitionen eines Type Tags werden als Attribute (weiter als Modellattribute bezeichnet) des entsprechenden Typs übernommen. Abgeleitete Modellattribute bestehen in der Regel aus einem Namen, Attributtyp und einer Multiplizität.

Modellattribute

Typen besitzen neben den Modellattributen aus Attributdefinitionen auch Attribute, die aus den Benutzereingaben abgeleitet sind. Um diese zu ermitteln, werden alle Wiki-Seiten des aktuellen Typs (alle Wiki-Seiten mit dem entsprechenden Type Tag) betrachtet – das Bottom-up-Modell wird abgeleitet und analysiert. Im Gesamtmodell wird die Herkunft der Modellattribute verdeutlicht. Soweit das Modellattribut aus einer Attributdefinition stammt, wird die *Strict*-Eigenschaft (wenn vorhanden) dargestellt.

Es besteht die Möglichkeit, dass es zwar eine Attributdefinition vorliegt, jedoch die Multiplizität und/oder der Attributtyp nicht definiert ist[1]. Sobald es auf den Wiki-Seiten Attribute existieren, die aus der entsprechenden Attributdefinition instanziiert wurden, wird die fehlende Information für das abgeleitete Modellattribut aus dem Bottom-up-Modell übernommen. Besitzen die instanziierten Attribute Werte von unterschiedlichen Typen, so wird in das Modell Text als Typ übernommen. Denn Text stellt unter den möglichen Attributtypen wie Date, Number, Currency, Boolean usw. den allgemeinsten Typ dar. Sobald es aber unter den Attributwerten ein Link existiert, wird der Typ der Zielseite als Attributtyp übernommen. Dann wird das Modellattribut als eine Assoziation modelliert, weil Beziehungen für die Modellierung der Datenstruktur einen höheren Stellenwert als Attribute haben.

Assoziationen

Beziehungen entsprechen in Hybrid Wikis den Link-Attributen. Sie werden aus den Attributdefinitionen und aus der Datenbasis nach gleichen Regeln wie Modellattribute abgeleitet. Wenn es für eine Assoziation mehrere definierte oder abgeleitete Typen vorliegen, modelliert man die Assoziation zu einem gemeinsamen Supertyp (mehr dazu im Abschnitt 3.2).

Der Abschnitt 2.1.2 beschreibt u.a. die *incoming links*. Zusammen mit den ursprünglichen

[1] In der aktuellen Version beinhalten die Attributdefinitionen Text als Default-Attributtyp. Da Text jedoch alle anderen Datentypen als valide Eingabe zulässt, wird er als undefiniertes Attributtyp behandelt.

Assoziationen stellen sie ungerichtete (bidirektionale) Beziehungen dar, denn eine Navigation in beide Richtungen möglich ist. Die Rückrichtung wird im Modell jedoch ignoriert. Die *Incoming links*-Attribute werden weder in Attributdefinitionen modelliert, noch kann der Benutzer sie auf den Wiki-Seiten eingeben. Sie werden automatisch von Tricia für eine bessere Navigation gebildet und gehören somit nicht in das Modell. Es wäre denkbar eine bidirektionale Assoziation zu modellieren, wenn es Link-Definitionen (Attributdefinition mit dem Typ Link) oder instanziierte Link-Attribute in beide Richtungen existieren würden. Aber eine Zuordnung von zwei gerichteten Assoziationen ist nicht eindeutig, denn sie können mit Absicht zwei unäbhängige Beziehungen darstellen. In solchen Fällen werden also zwei gerichtete Assoziationen modelliert.

3.1.2 Zusätzliche Informationen

Neben den o.g. Informationen, die die Struktur des Datenbestands beschreiben, sind bestimmte analytische/statistische Informationen denkbar, um die Gesamtsituation besser zu kommunizieren. Da das Modell aus den Daten abgeleitet wird und den aktuellen Zustand wiedergibt, können diese Informationen z.B. zum Vergleich mit älteren Versionen der Datenstruktur verwendet werden, um Entwicklungen und Tendenzen zu erkennen. Die zusätzlichen Informationen sind:

- **Anzahl der Instanzen**: beschreibt sowohl Modellattribute (wie oft wurde das Attribut mit diesem Attributnamen angegeben), als auch Modelltypen (wie viele Wiki-Seiten mit dem entsprechenden Type Tag existieren). Anzahl der Instanzen verdeutlicht die Wertigkeit eines Attributs: wird auf 100 Wiki-Seiten von Typ *person* nur einmal das Attribut *hobby* angegeben, so hat es eine geringere Auswirkung auf die Gesamtstruktur als z.B. das 99 mal instanziierte Attribut *nachname*, das somit ein Kandidat für eine Attributdefinition ist. Handelt es sich um ein aus den Daten abgeleitetes Attribut, so gibt die Anzahl der Instanzen dieses Attributs – geteilt durch die Anzahl der Wiki-Seiten des zugehörigen Typs – wie hoch die Auswirkung des Modellattributs auf die Gesamtstruktur ist.

- **Anzahl der fehlerhaften Instanzen**: existiert eine Attributdefinition, so werden alle instanziierten Attribute auf Korrektheit kontrolliert. Es kann entweder die Multiplizität oder der Attributtyp falsch sein. Sind es viele Attribute, die fehlerhaft sind, so soll z.B. eine Änderung der Attributdefinition in Betracht gezogen werden.

- **Abgeleitete Multiplizität** gibt an, wie viele Werte mindestens und wie viele maximal bei einem bestimmten Attribut auf allen Wiki-Seiten angegeben wurden. Wenn die untere Grenze 0 ist, dann bedeutet das, dass auf mindestens einer Wiki-Seite des dazugehörigen Typs das entsprechende Attribut nicht vorhanden ist. Ist die Anzahl der Instanzen eines Modellattributs kleiner als die Anzahl der Instanzen des zugehörigen Typs, so ist die untere Grenze automatisch 0. Abgeleitete Multiplizität dient in erster Linie zur Angabe der Multiplizität für Attribute, die keine Attributdefinition besitzen. Für diejenigen, die eine definierte Multiplizität haben, ist die Abweichung zur tatsächlichen Multiplizität interessant, die von Benutzern bestimmt wird.

3.1.3 Zusammenfassung

Das Modell beinhaltet strukturelle Informationen, die Objekte und Zusammenhänge beschreiben. Typen bilden die höchste Ebene im Modell. Sie werden aus den Type Tags abgeleitet. Die Typen besitzen Attribute, die aus Attributdefinitionen sowie aus den Daten abgeleitet werden. Es existieren folgende Konstellationen:

- Es liegt eine Attributdefinition vor, die einen Attributtyp sowie eine Multiplizität besitzt. Information wird in das Modell übernommen. Es muss auch die Herkunft aus Attributdefinition kommuniziert werden.
- Es liegt eine Attributdefinition vor, die aber keinen Attributtyp und/oder keine Multiplizität besitzt. In diesem Fall wird die fehlende Information aus den Benutzereingaben (instanziierten Attributen) ermittelt. Es wird im Modell kenntlich gemacht, dass die Informationen unterschiedlicher Herkunft sind – aus Attributdefinition und Benutzereingaben.
- Attributtyp wird aus instanziierten Attributen ermittelt, deren Werte von unterschiedlichen Typen sind. Es wird der allgemeinste Typ – Text (String) – in das Modell übernommen. Liegt unter den Attributwerten ein Link vor, so wird aus dem Attribut eine Assoziation gebildet.
- Eine Assoziation besitzt mehrere mögliche Typen. In diesem Fall wird ein Supertyp modelliert.
- Ist die Attributdefinition *Strict*, so wird das im Modell dargestellt.
- Es liegt keine Attributdefinition vor. Die Information für das Modell wird komplett aus den Benutzereingaben ermittelt. Die Herkunft der Daten wird im Modell dargestellt.

Neben den strukturellen beinhaltet das Modell auch analytische Informationen:

- Jeder Typ im Modell wird mit der Anzahl der Wiki-Seiten versehen, die den entsprechenden Type Tag besitzen.
- Modellattribute (egal ob aus Attributdefinition oder Benutzereingaben) werden ebenfalls mit der Anzahl der Instanzen versehen.
- Liegt eine Attributdefinition vor, so wird zusätzlich die Anzahl der fehlerhaften Instanzen der Attribute angezeigt.

Typen besitzen also einen Namen, Anzahl der Instanzen und Modellattribute. Modellattribute besitzen folgende Informationen:

1. Attributname
2. Attributtyp
3. definierte Multiplizität (wenn vorhanden)
4. Strict-Eigenschaft

5. Attributdefinition vorhanden (Herkunft)
6. Typdefinition vorhanden (Herkunft)
7. Abgeleitete Multiplizität
8. Anzahl der Instanzen
9. Anzahl der fehlerhaften Instanzen

Beziehung (oder Assoziation) verfügen über die gleichen Informationen wie Modellattribute. Der referenzierte Typ wird als Attributtyp modelliert.

Die Visualisierung von diesen Informationen wird im Folgenden beschrieben.

3.2 Mapping zwischen Hybrid Wikis und UML-Klassendiagramm

UML-Klassendiagramme eigenen sich zur Modellierung von Daten, denn sie besitzen die notwendigen Modellierungselemente (Siehe 2.1.5). Klassen repräsentieren die Objekttypen in einem System. Attribute eines Klassendiagramms entsprechen den in 3.1 abgeleiteten Attributen. Sie besitzen zum Teil gleiche Informationen: Attributname, Attributtyp und Multiplizität. Deswegen sind die aus Type Tags abgeleiteten Typen mit Modellattributen analog zu Klassen und Attributen des UML-Klassendiagramms und können als Klassen visualisiert werden.

Die aus den Link-Attributen in Hybrid Wikis abgeleiteten Assoziationen sind mit den Assoziation in einem Klassendiagramm vergleichbar. Sie beschreiben Beziehungen zwischen den Typen (analog zu Klassen).

Abbildung 3.1 zeigt zuerst die Ableitung der für die Modellierung relevanten Informationen. Im nächstem Schritt wird die Abbildung auf die Konzepte des Klassendiagramms dargestellt.

Das Konzept der Generalisierung in UML ermöglicht die Darstellung von Link-Attributen, die in der Attributdefinition mehrere Type Tags für die Zielseite besitzen. Da die Type Tags disjunktiv verknüpft sind, werden Wiki-Seiten von unterschiedlichen Typen als Attributwert der Assoziation zugelassen. Zur Darstellung wird eine Superklasse (oder Supertyp) gebildet, die als Unterklassen die angegebenen Typen (Type Tags) beinhaltet. Diese Superklasse wird als Zielklasse der Assoziation gesetzt. Abbildung 3.2 zeigt die Attributdefinition *bearbeiter* mit Link als Typ und zwei Type Tags *person* und *professor*. Die Attributdefinition ist im Type Tag *projekt* modelliert.

Abbildung 3.3 zeigt das dazugehörige Diagramm: die Klasse *projekt*, die den aus dem entsprechenden Type Tag abgeleiteten Typ symbolisiert, besitzt eine gerichtete Assoziation zur Klasse *professor_student*. Dieses Diagramm drückt aus, dass es für den Typ Projekt ein Bearbeiter definiert ist, der entweder ein Student oder ein Professor sein kann. Das gleiche Prinzip gilt auch für die Beziehungen, die keine Attributdefinition besitzen und aus der Datenbasis abgeleitet sind. Sie werden auf die gleiche Art und Weise dargestellt. Die Herkunft der abgeleiteten Assoziation wird im Modell dargestellt.

UML-Klassendiagramme ermöglichen die Visualisierung von strukturellen Informationen aus dem abgeleiteten Modell. Es können die Typen, deren Attribute inklusive der

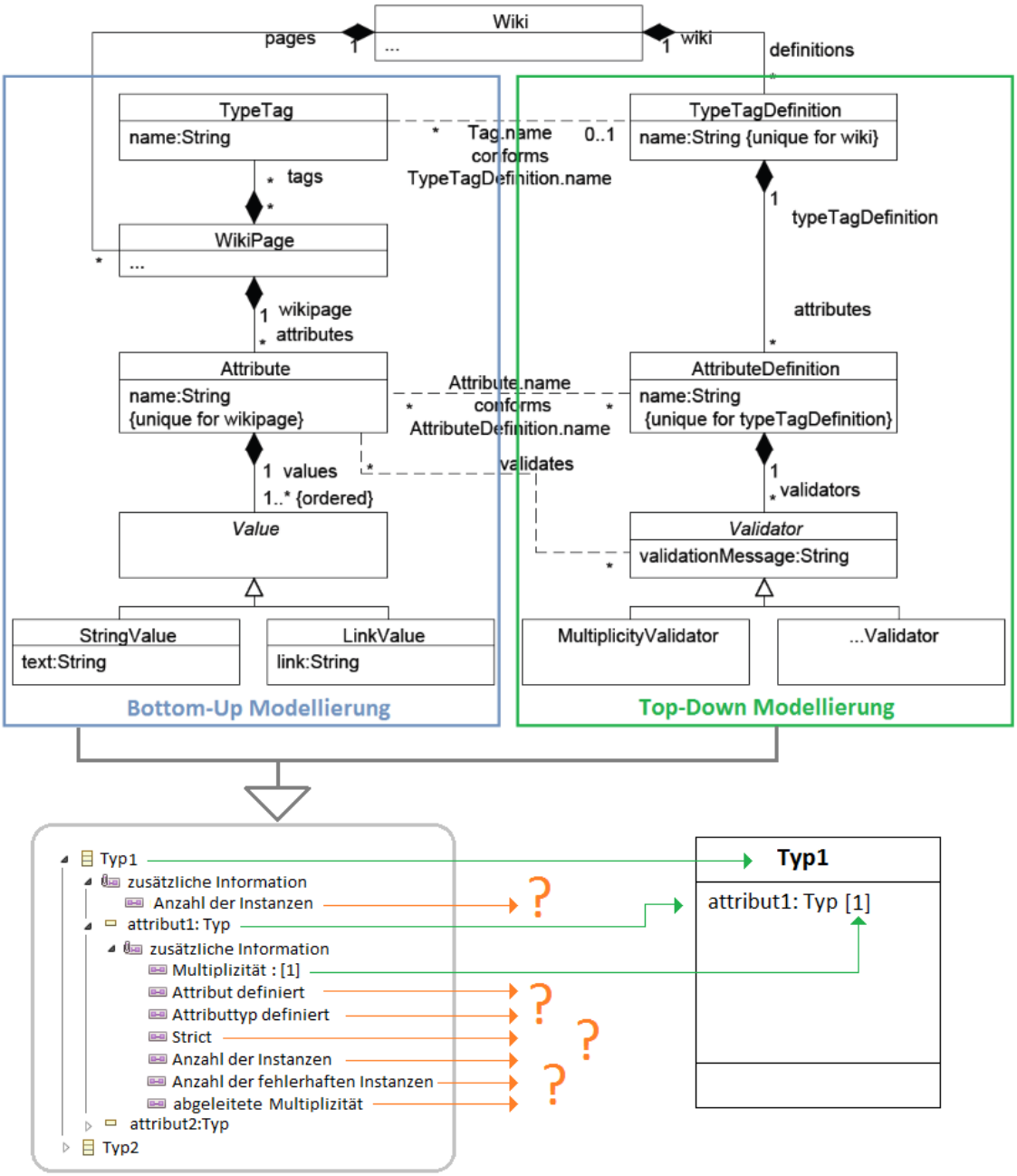


Abbildung 3.1: Schematische Darstellung der Modellableitung und Abbildung auf UML-Klassendiagramm

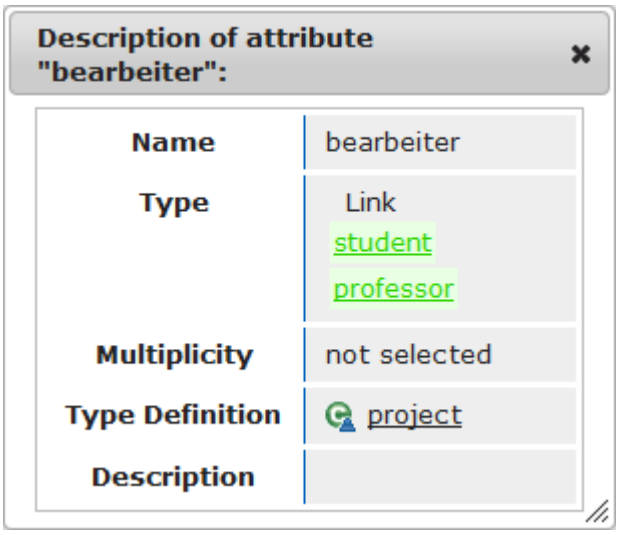


Abbildung 3.2: Attributdefinition mit zwei möglichen Type Tags für die Zielseite

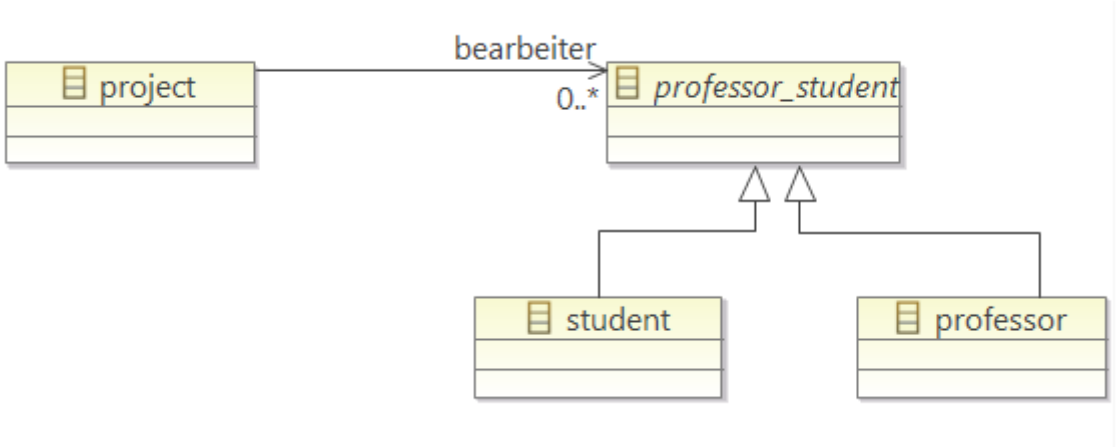


Abbildung 3.3: Darstellung einer Assoziation mit zwei möglichen Typen mit Hilfe eines Supertyps

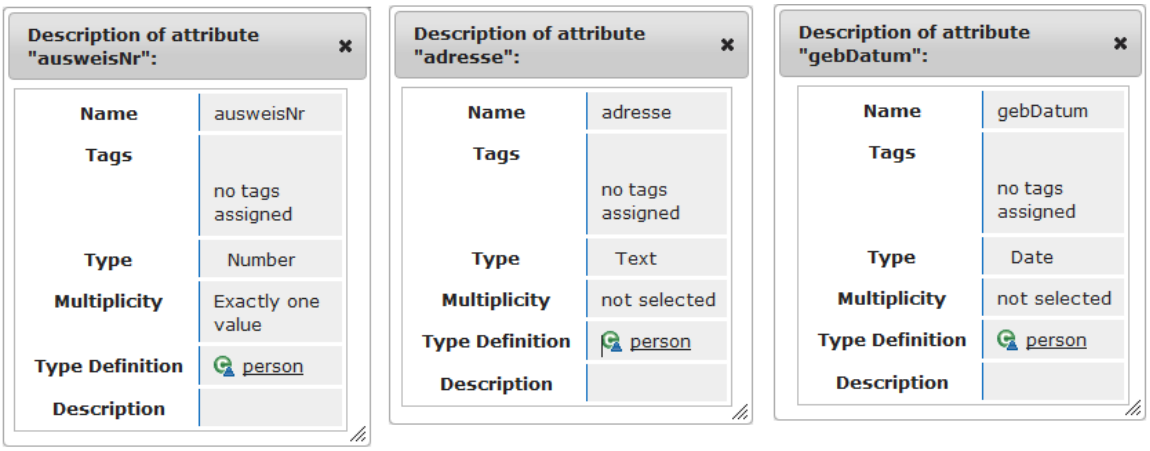


Abbildung 3.4: Beispiele für Attributdefinitionen

Multiplizität sowie die Assoziationen dargestellt werden. Jedoch existiert für viele der in Abschnitt 3.1.1 identifizierten Informationen keine Darstellungsmöglichkeit (Siehe Abbildung 3.1). So lässt sich z.B. die Herkunft der Modellattribute (die für die Modellierung im Falle von Hybrid Wikis von großer Bedeutung ist) mit Standardelementen eines UML-Klassendiagramms nicht darstellen.

3.3 Beispiel: Modellableitung

Um die Modellableitungsstrategie zu veranschaulichen, wird im Folgenden die Ableitung von modellrelevanten Informationen aus Hybrid Wikis an einem Beispiel demonstriert. Außerdem wird die Notwendigkeit der Einführung von zusätzlichen Visualisierungselementen verdeutlicht.

In Hybrid Wikis wurde eine Datenstruktur erstellt. Die Top-down-Modellierung beinhaltet folgende Informationen:

- Es liegt eine Typdefinition (ein Type Tag) mit dem Titel *person* vor.
- In diesem Type Tag sind drei Attributdefinitionen vorhanden: *ausweisNr*, *adresse* und *gebDatum* (Abb. 3.4).
- Die Attributdefinition *ausweisNr* besteht aus einer Multiplizität (1) und einem Attributtyp (Number). Außerdem ist die Attributdefinition *Strict*.
- Die Attributdefinition *addresse* besitzt keine Informationen.
- Die Attributdefinition *gebDatum* besitzt einen Attributtyp (Date) aber keine Multiplizität.

Die Benutzereingaben (Bottom-up-Modellierung) liefern folgendes Bild (Abb. 3.5) :

- Es existiert eine Wiki-Seite mit dem Type Tag *person*.

Abbildung 3.5: Instanziierte Attribute auf einer Wiki-Seite

- Auf dieser Wiki-Seite ist das Attribut *ausweisNr* mit dem Wert 12345 instanziiert.
- Das Attribut *adresse* besitzt zwei Werte von Typ Text: Musterstraße und Musterstadt.
- *gebDatum* ist mit dem Wert 21. November von Typ Text instanziiert.
- Attribut hobby ist mit dem Wert Schwimmen von Typ Text angegeben.

Nun wird aus den vorliegenden Definitionen und Benutzereingaben ein Modell abgeleitet. Im ersten Schritt betrachtet man die Definitionen:

- Aus dem Type Tag *person* wird der Typ *person* gebildet.
- Aus den dazugehörigen Attributdefinitionen werden die Modellattribute *ausweisNr*, *adresse* und *gebDatum* abgeleitet.

Nach der Modellableitung aus den Definitionen besitzt das Modell folgende Informationen:
Person:

- ausweisNr: Number
  - Attribut definiert: true
  - Attributtyp definiert: true
  - definierte Multiplizität: 1
  - Strict: true
- adresse:
  - Attribut definiert: true
  - Attributtyp definiert: false
  - definierte Multiplizität: -
  - Strict: false

- gebDatum: Date
  - Attribut definiert: true
  - Attributtyp definiert: true
  - definierte Multiplizität: -
  - Strict: false

Im nächsten Schritt wird die Datenbasis (Benutzereingaben) betrachtet. Dabei werden nicht die Definitionen, sondern Objekte (Wiki-Seiten) analysiert.

- Die einzige vorhandene Wiki-Seite hat den Type Tag „person“. Anzahl der Wiki-Seiten mit dem Type Tag „person“ beträgt also 1.
- Attribut „ausweisNr“ ist korrekt. Daraus werden für das Modellattribut „name“ im Typ „person“ folgende Informationen ermittelt: Anzahl der fehlerhaften Instanzen = 0, Anzahl der Instanzen = 1 und abgeleitete Multiplizität = 1.
- Attribut „adresse“ liefert für das entsprechende Modellattribut folgende Informationen: Attributtyp = Text, abgeleitete Multiplizität = 2, Anzahl der Instanzen = 1, Anzahl der fehlerhaften Instanzen = 0.
- Attribut „gebDatum“ ist fehlerhaft, da sein Wert einen anderen Typ besitzt als definiert (Text statt Date). Anzahl der Instanzen = 1, Anzahl der fehlerhaften Instanzen = 1, instanziiere Multiplizität = 1.
- Attribut „hobby“ besitzt keine Attributdefinition. Es werden folgende Informationen abgeleitet: Attributtyp = Text, Attribut definiert = false, Anzahl der Instanzen = 1, instanziiere Multiplizität = 1.

Das abgeleitete Modell beinhaltet nun folgende Informationen:
Person:(Anzahl der Instanzen: 1)

- ausweisNr: Number
  - Attribut definiert: true
  - Attributtyp definiert: true
  - definierte Multiplizität: 1
  - Strict: true
  - Anzahl der Instanzen: 1
  - Anzahl der fehlerhaften Attribute: 0
  - instanziiere Multiplizität: 1
- adresse: Text
  - Attribut definiert: true
  - Attributtyp definiert: false
  - definierte Multiplizität: -

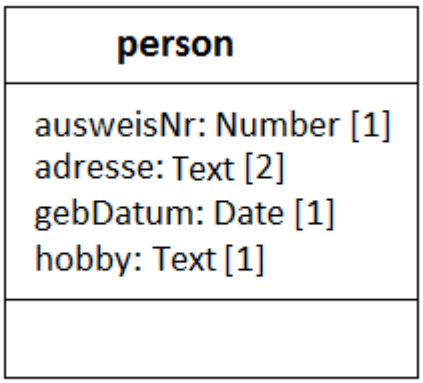


Abbildung 3.6: Darstellung des Modells aus dem Beispiel mit Standardelementen des UML-Klassendiagramms

  - Strict: false
  - Anzahl der Instanzen: 1
  - Anzahl der fehlerhaften Attribute: 0
  - instanziiere Multiplizität: 2

- gebDatum: Date
  - Attribut definiert: true
  - Attributtyp definiert: true
  - definierte Multiplizität: -
  - Strict: false
  - Anzahl der Instanzen: 1
  - Anzahl der fehlerhaften Attribute: 1
  - instanziiere Multiplizität: 1

- hobby: Text
  - Attribut definiert: false
  - Anzahl der Instanzen: 1
  - instanziiere Multiplizität: 1

Die Visualisierung mit vorhandenen Modellierungselementen berücksichtigt nur strukturelle Informationen. Viele der abgeleiteten Informationen gehen dabei verloren. Abbildung 3.6 zeigt die Visualisierung des abgeleiteten Modells mit Hilfe eines UML-Klassendiagramms

3.4 UML-Erweiterung

Für die vollständige Visualisierung des aus Hybrid Wikis abgeleiteten Modells müssen zusätzlich zu den Elementen eines UML-Klassendiagramms neue Modellierungskonzepte entwickelt werden. Mit einem UML-Klassendiagramm lässt sich die Herkunft der abgeleiteten Informationen sowie quantitative Daten wie Anzahl der Instanzen oder die abgeleitete Multiplizität nicht darstellen. Im Folgenden werden unterschiedliche Varianten der Visualisierung für diese Informationen betrachtet und diskutiert.

3.4.1 Herkunft der Informationen

Die Herkunft der Informationen ist von zentraler Bedeutung für die Modellableitung in Hybrid Wikis. Die aus den Typ- und Attributdefinitionen stammenden Informationen entsprechen in ihrem Ursprung den klassischen Attributen eines Klassendiagramms. Sie werden ebenso explizit modelliert und dienen als ein festes Gerüst für die darauf abstützenden Daten. Sie bilden den Kern eines Typs und können auf die gleiche Weise wie die Attribute einer Klasse visualisiert werden. Es soll nun herausgearbeitet werden, wie die aus den Daten abgeleiteten Modellattribute zu visualisieren sind.

In Zusammenhang mit der Herkunft der Informationen ist für die Modellattribute, die aus den Daten abgeleitet sind, der Festigungsgrad wichtig. Dieser beschreibt die Wertigkeit des Modellattributs für die Bestimmung der Gesamtstruktur. Der Festigungsgrad ist definiert als die Anzahl der instanziierten Attribute geteilt durch die Anzahl der Wiki-Seiten des zugehörigen Typs. Es sind also Werte zwischen 1 und 0 möglich, wobei 1 einen sehr hohen Festigungsgrad darstellt. Dieser Indikator verfeinert den Unterschied zwischen dem fest definierten und aus den Daten abgeleiteten Modell: ist ein Attribut bei den meisten Wiki-Seiten des zugehörigen Typs angegeben, so besitzt es einen hohen Festigungsgrad und bestimmt die Struktur in ähnlichem Maße wie die Attributdefinitionen. Natürlich sind auch andere Definitionen für den Festigungsgrad denkbar, die z.B. noch die Zusammensetzung der aus den Daten abgeleiteten Attributtypen berücksichtigen. Die o.g. Formel wurde aus Gründen der Einfachheit gewählt.

3.4.1.1 Variante 1

Die erste Variante ist visuell teilweise an die Schlagwortwolken (engl.: tag clouds) angelehnt. Die Überlegung dahinter ist, dass die aus den Daten abgeleiteten Modellattribute eine *diffuse* Struktur darstellen, die keinen festen Rahmen besitzt. Diese Struktur wird wie eine Schlagwortwolke um den Kern des Typs (fest definiertes Schema aus den Typ- und Attributdefinitionen) visualisiert. Abbildung 3.7 zeigt in der Mitte einen Typ mit zwei Modellattributen, die Attributdefinitionen besitzen. name und adresse sind fest definiert. Die Modellattribute gehalt, sprache, email, hobby und gebDatum besitzen keine Definitionen und wurden aus den Daten abgeleitet. Ihre Positionierung - Nähe zum Kern – repräsentiert die Auswirkung auf das Modell.

Die Entfernung wird mit Hilfe des Festigungsgrads bestimmt. Modellattribute mit dem Festigungsgrad nahe 1 sind am Kern platziert. Die Positionierung kann z.B. mit Hilfe einer Spirale durchgeführt werden (Abbildung 3.8). Die Modellattribute werden entlang der Spirale von der Mitte aus platziert.

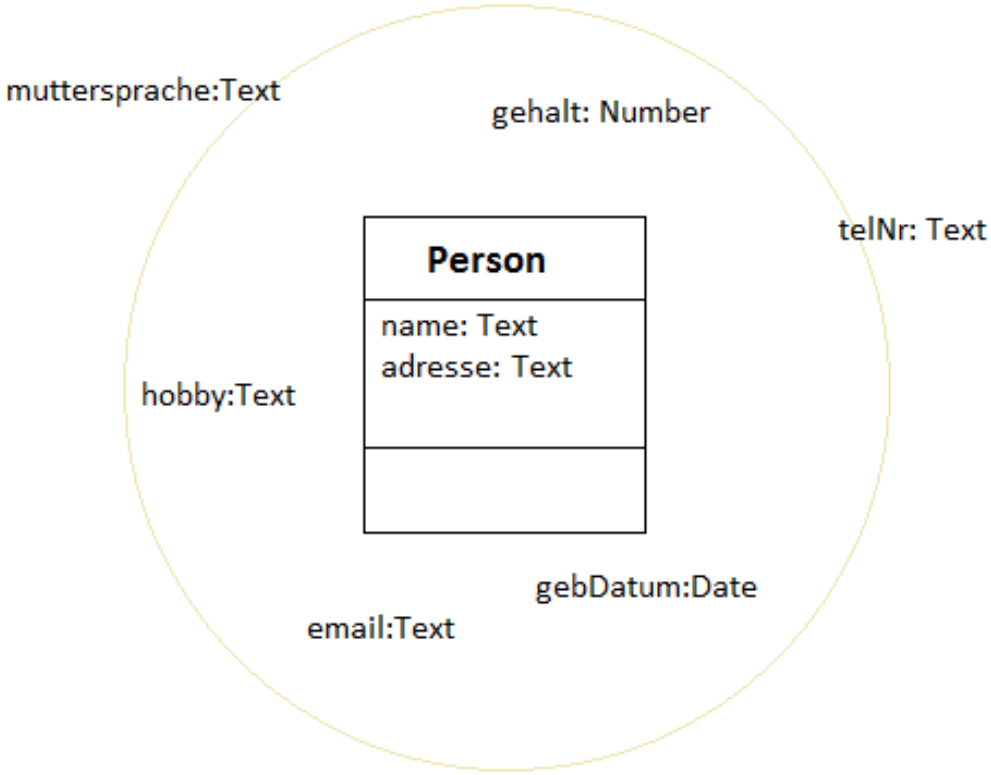


Abbildung 3.7: Variante 1 der kombinierten Visualisierung

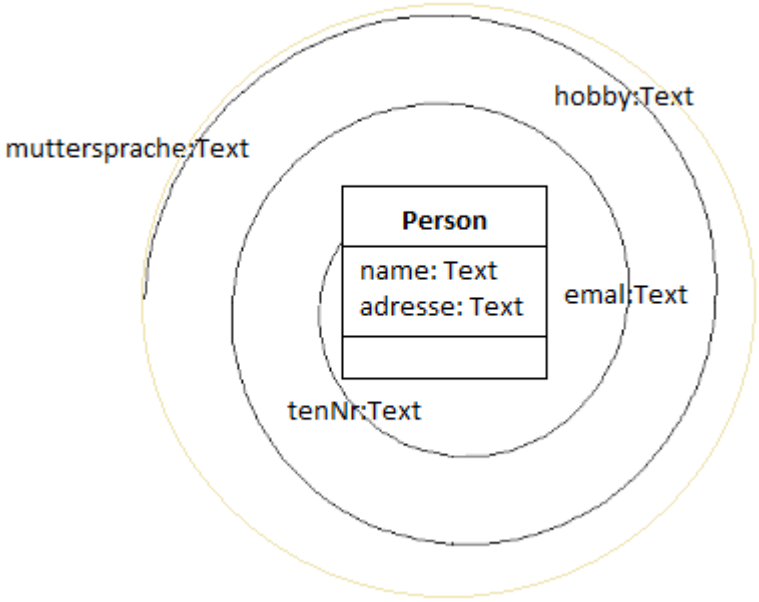


Abbildung 3.8: Positionierung der Elemente für die Variante 1

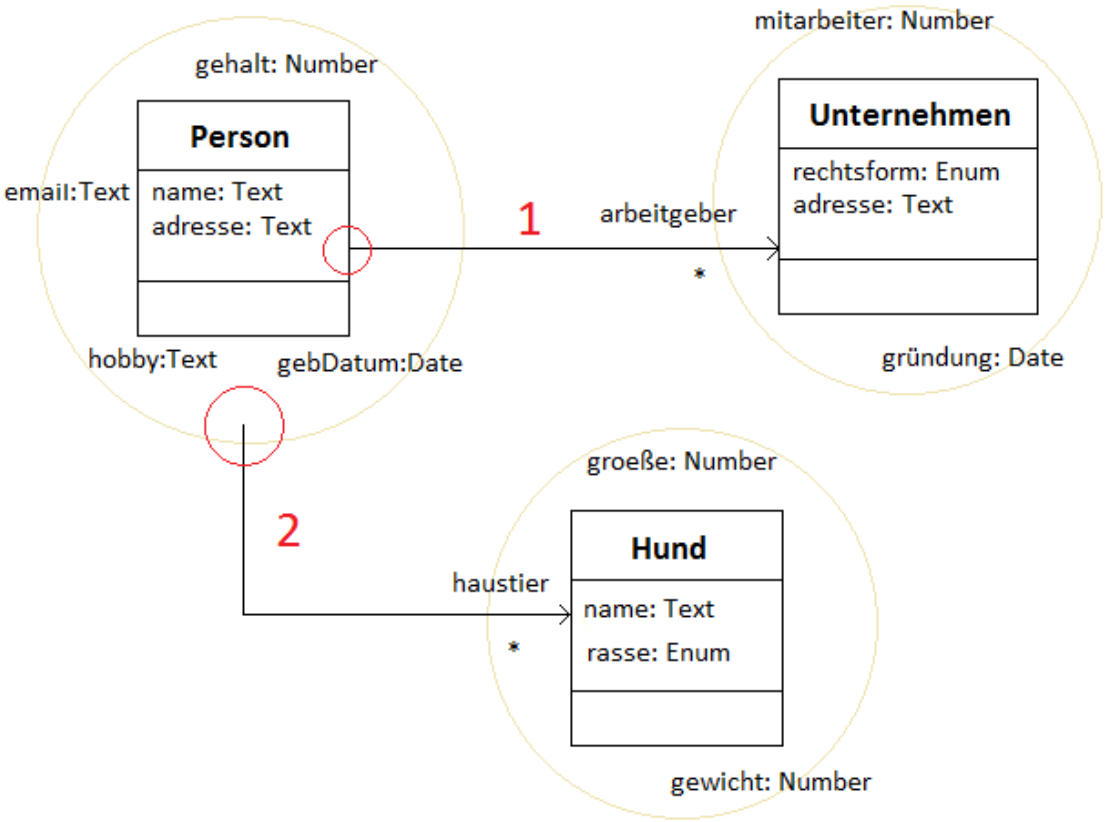


Abbildung 3.9: Assoziationen in der Variante 1

Das gleiche Prinzip gilt für die Assoziationen (Abbildung 3.9). Besitzt eine Assoziation eine zugrunde liegende Attributdefinition, so wird diese analog zu Assoziation eines Klassendiagramms dargestellt (Punkt 1 in der Abbildung 3.9). Wenn eine Assoziation jedoch aus der Datenbasis abgeleitet wurde, so ist ihr Ursprung entsprechend von dem Kern des Typs entfernt - Punkt 2 in Abbildung 3.9. Die Assoziation *haustier* (Punkt 2) besitzt keine Attributdefinition und kommt selten auf den Wiki-Seiten vom Typ *person* vor.

Vorteile:

- durch die räumliche Anordnung ist die Herkunft der Modellattribute und gleichzeitig der Festigungsgrad gut erkennbar

Nachteile:

- Visualisierung eines Typs benötigt eine große Darstellungsfläche
- Darstellung der Assoziationen ist problematisch, weil die Assoziationen durch die aus den Daten abgeleiteten Modellattribute behindert werden
- relativ schlechte Übersichtlichkeit
- Zusammengesetzte Modellattribute (Attributdefinition vorhanden, Typ aus den Daten abgeleitet) sind nicht darstellbar

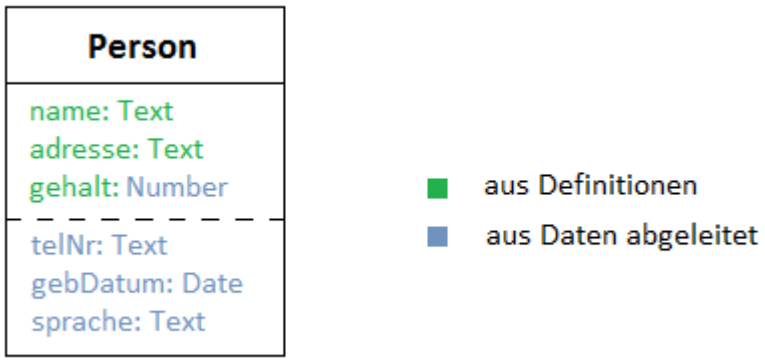


Abbildung 3.10: Kombinierte Darstellung eines Typs in der Variante 2

3.4.1.2 Variante 2

Die Herkunft der Daten wird in Variante 2 (Abbildung 3.9) durch die räumliche Anordnung und Farben visualisiert. Die Modellattribute aus den Definitionen werden analog zu den Attributen einer Klasse dargestellt. Die aus den Daten abgeleiteten Modellattribute werden im unteren, für Methoden der Klasse vorgesehenen, Abschnitt visualisiert. Die Trennlinie zwischen den Abschnitten ist allerdings gestrichelt, da es sich um gleiche konzeptuelle Einheiten handelt. Unterschiedliche Farben werden zur Verdeutlichung sowie zur Darstellung von kombinierten Modellattributen verwendet. So besitzt das Modellattribut *gehalt* aus der Abbildung 3.9 zwar eine Attributdefinition, der Attributtyp wurde jedoch aus den Daten abgeleitet.

Die gleichen Darstellungsregeln gelten für die Assoziationen. Besitzt eine Assoziation eine Attributdefinition, so wird der Rollenname (Attributname) in Grün dargestellt. Ist in der Attributdefinition auch der Typ der Zielseite definiert, so wird die Linie, die die Assoziation repräsentiert, auch grün dargestellt. Wird der Typ jedoch aus den Daten abgeleitet, so ist die Linie in Blau. Abbildung 3.14 beinhaltet eine kombinierte Assoziation *arbeitgeber*: es liegt eine Attributdefinition vor, aber der Typ wurde aus den Daten abgeleitet. Abbildung 3.11 zeigt den gleichen Sachverhalt wie die Abbildung 3.9 in der zweiten Variante der Visualisierung.

Der Festigungsgrad kann in Variante 2 auf unterschiedliche Weisen visualisiert werden. Eine Möglichkeit ist die Unterscheidung durch den Schriftgrad: je größer ein Modellattribut dargestellt wird, desto höher ist seine Auswirkung auf die Gesamtstruktur (Punkt 1 in Abbildung 3.12). Eine andere Möglichkeit ist die Verwendung der Farbsättigung (Punkt 2 in Abbildung 3.12): wird ein Modellattribut beinahe transparent dargestellt, so kommt es selten auf den Wiki-Seiten vor und hat deswegen eine geringe Auswirkung auf das Modell.

Darstellung des Festigungsgrads mit Hilfe des Schriftgrads und der Farbsättigung beeinträchtigt die Lesbarkeit der Visualisierung. Kleingeschriebene oder fast durchsichtige Modellattribute besitzen zwar eine niedrige Wertigkeit für das Modell, sollen aber trotzdem lesbar und leicht zu finden sein. Außerdem sind im Vergleich zur Variante 3.4.1.1 die Unterschiede schlecht erkennbar. So kann z.B. die Differenz zwischen zwei Modellattri-

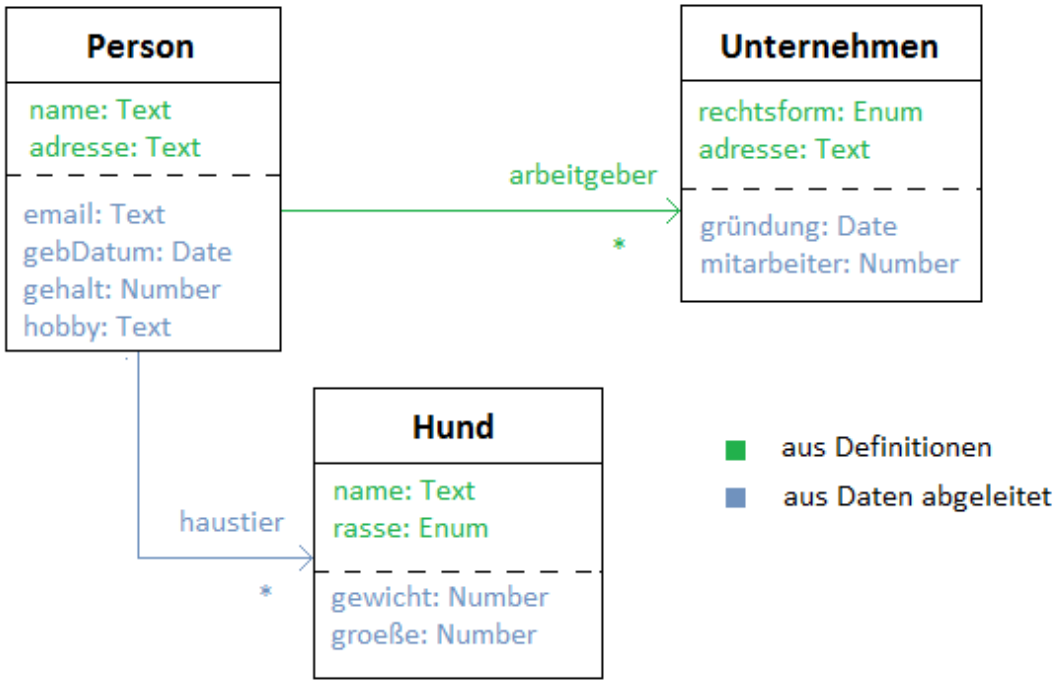


Abbildung 3.11: Assoziationen in der Variante 2

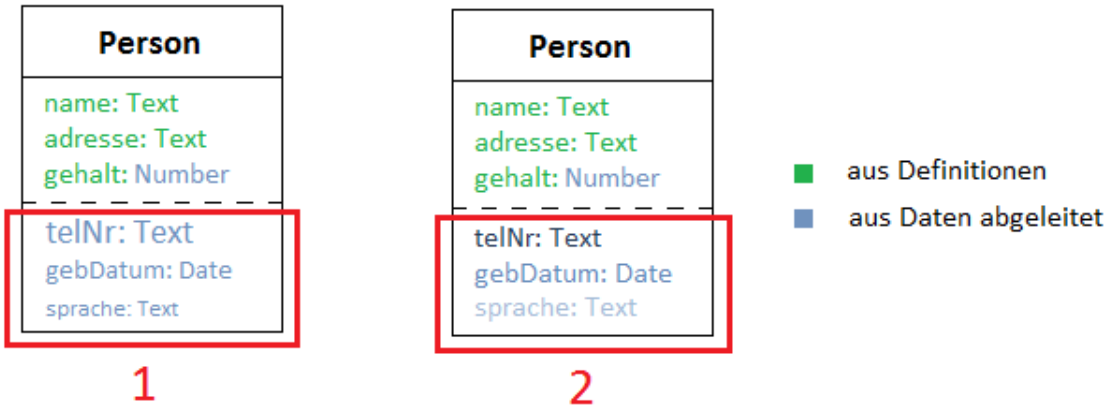


Abbildung 3.12: Möglichkeiten zur Darstellung des Festigungsgrads

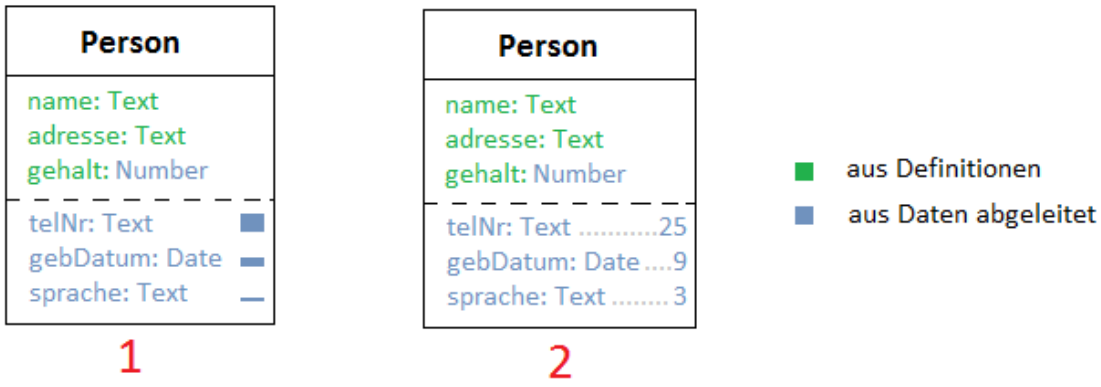


Abbildung 3.13: Möglichkeiten zur Darstellung des Festigungsgrads 2

buten mit 1 und 100 Instanzen durch den Schriftgrad schlecht dargestellt werden. Eine Kombination von Schriftgrad und Sättigung führt zu etwas besseren Ergebnissen.

Eine Lösung für das Problem bieten die in Abbildung 3.13 vorgestellten Formen der Visualisierung. Punkt 1 zeigt die Verwendung von Balken, die abhängig von dem Festigungsgrad des jeweiligen Modellattributs unterschiedliche Höhen aufweisen. Im Punkt 2 wird neben den Modellattributen die Anzahl der Instanzen angezeigt. Zwar gibt es hier keinen Höchstpunkt wie bei den anderen Vorschlägen zur Darstellung des Grads der Auswirkung (der Festigungsgrad kann maximal 1 sein), jedoch können die Unterschiede besser kommuniziert werden. Die beiden Ansätze aus der Abbildung 3.13 lassen sich auf die Assoziationen übertragen. Die Rolle der Balken übernimmt bei den Assoziationen die Linienstärke (Abbildung 3.14).

Vorteile:

- platzsparend
- kombinierte Modellattribute darstellbar
- übersichtlich

Nachteile:

- Herkunft der aus der Datenbasis abgeleiteten Modellattribute nicht intuitiv verständlich

3.4.1.3 Auswahlentscheidung

Es sind weitere Möglichkeiten zur kombinierten Darstellung des Modells denkbar. Sie stellen aber in gewisser Weise Abwandlungen der vorgestellten Varianten mit ähnlichen Vor- und Nachteilen dar. Deswegen wurde die Auswahl auf die o.g. Varianten beschränkt.

Variante 3.4.1.1 und die möglichen Abwandlungen davon stellen intuitiv die unterschiedliche Herkunft der Informationen und die Wertigkeit der einzelnen aus den Daten abgeleiteten Modellattribute dar. Dies wird jedoch auf Kosten der Übersichtlichkeit und

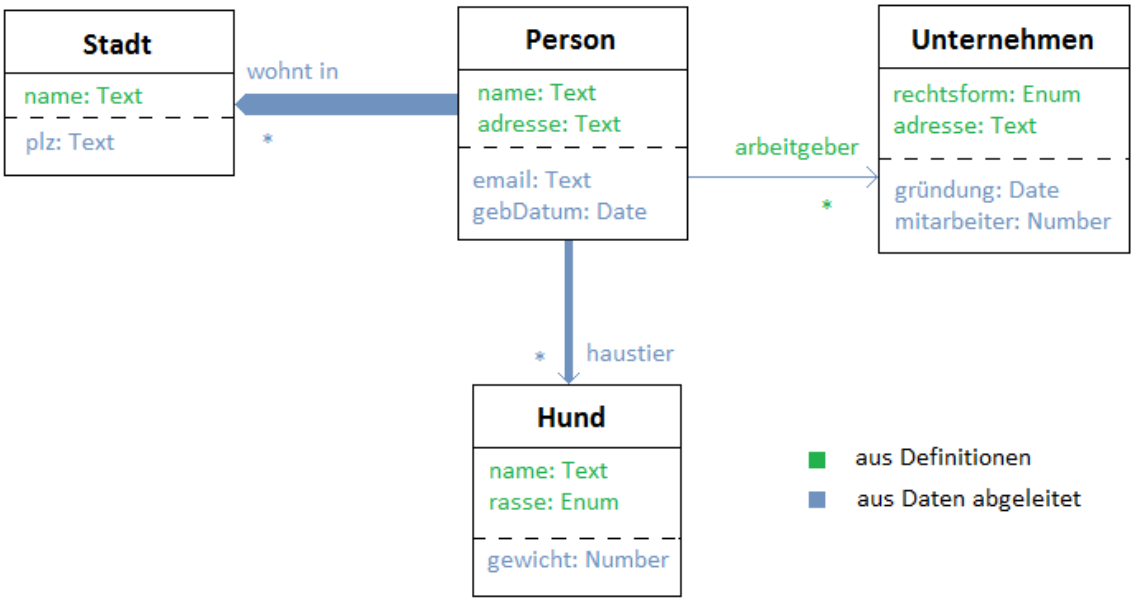


Abbildung 3.14: Darstellung des Festigungsgrads für Assoziationen

der großen Darstellungsfläche erreicht. Außerdem sind zusammengesetzte Modellattribute und -assoziationen nicht darstellbar.

Variante 3.4.1.2 bietet eine klare Struktur, ist übersichtlich und benötigt zur Darstellung relativ wenig Platz. Die Herkunft der Informationen wird durch räumliche Positionierung sowie durch Farben signalisiert. Der Nachteil dieser Variante liegt darin, dass sie einen Erklärungsbedarf aufwirft, denn die Bedeutung von Farben und der räumlichen Anordnung sind nicht sofort ersichtlich. Zu Visualisierung des Festigungsgrads liegen mehrere Möglichkeiten vor. Darstellung mit Hilfe der Farbsättigung und des Schriftgrades beeinträchtigt die Lesbarkeit und zeigt die Unterschiede der möglichen Werte ungenau an. Darstellung mit Balken und Zahlen liefert bessere Ergebnisse in Bezug auf die Lesbarkeit und Genauigkeit der angezeigten Informationen.

Aufgrund der überwiegenden Vorteile wurde die zweite Variante für die weitere Entwicklung ausgewählt.

3.4.2 Zusätzliche Informationen

Mit den bisher eingeführten Modellierungselementen können folgende der in 3.1 identifizierten Informationen visualisiert werden:

- Attributname
- Attributtyp
- definierte Multiplizität (wenn vorhanden)
- Ob Attributdefinition vorhanden (Herkunft)

Abbildung 3.15: Darstellungsmöglichkeit der Anzahl der Instanzen in der Variante 2

- Ob Typdefinition vorhanden (Herkunft)
- Anzahl der Instanzen

Anzahl der Instanzen wurde bisher nur für die aus den Daten abgeleiteten Modellattribute angegeben (Punkt 2 in Abbildung 3.13). Auf die gleiche Weise kann die Anzahl der Instanzen für die Modellattribute aus den Attributdefinitionen (Punkt 1 in Abbildung 3.15), die Anzahl der Wiki-Seiten des Typs (Punkt 2) sowie Anzahl der Instanzen einer Assoziation (Punkt 3) visualisiert werden.

Es bleiben also noch die Strict-Eigenschaft, die abgeleitete Multiplizität sowie die Anzahl der fehlerhaften Instanzen, die keine Darstellungsmöglichkeiten haben.

Die Strict-Eigenschaft kommt nur bei Modellattributen vor, die aus den Attributdefinitionen stammen. Sie kann durch Hinzufügen eines Ausrufezeichens zum Modellattribut dargestellt werden. Bei Assoziationen wird das Ausrufezeichen nach der Multiplizität hinzugefügt (Punkt 1 in Abbildung 3.17). Ein Ausrufezeichen steht allgemein für *wichtig* und eignet sich deswegen zur Darstellung der *Strict*-Eigenschaft. Eine weitere Möglichkeit zu kommunizieren, dass ein Modellattribut Strict ist, besteht darin, die Anzahl der fehlerhaften Instanzen eines Modellattributs in Rot darzustellen. Ist das Modellattribut nicht Strict, so wird die Anzahl der Instanzen gelb visualisiert. Rote Farbe signalisiert, dass die Attributwerte nicht zulässig sind. Gelb gibt an, dass sie zwar fehlerhaft sind, jedoch bei erneuter Eingabe vom System angenommen und gespeichert werden können. In Abbildung 3.16 besitzt das Modellattribut *name* die *Strict*-Eigenschaft. Diese ist durch ein Ausrufezeichen nach der Multiplizität gekennzeichnet. Außerdem ist die Anzahl der fehlerhaften Instanzen (in Klammern nach der Anzahl der instanziierten Attribute) in Rot dargestellt. Die Modellattribute *adresse* und *gehalt* besitzen keine *Strict*-Eigenschaft. Aus diesem Grund wird die Anzahl der fehlerhaften Instanzen gelb dargestellt.

Die abgeleitete Multiplizität gibt im Gegensatz zur definierten Multiplizität die tatsächliche Anzahl der angegebenen Attribute an. Sie wird bei allen aus der Datenbasis abgeleiteten Modellattributen dargestellt. Wie man in der Abbildung 3.16 sieht, beinhalten die Modellattribute aus den Attributdefinitionen eine Menge an Daten. Um eine Informationsüberladung zu vermeiden, wird die abgeleitete Multiplizität nur bei fehlender

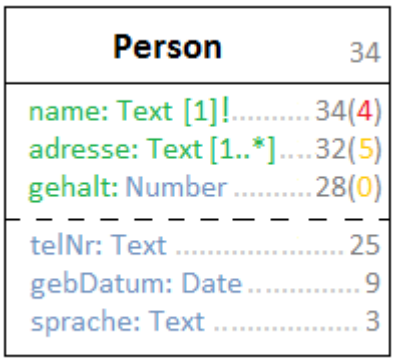


Abbildung 3.16: Darstellung der Strict-Eigenschaft

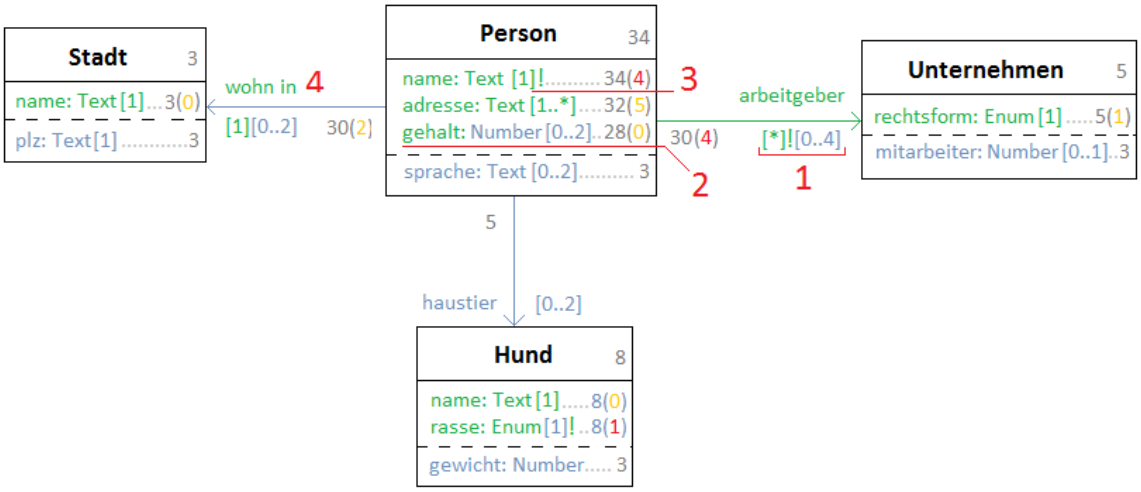


Abbildung 3.17: All Visualisierungskonzepte der Variante 2

definierter Multiplizität dargestellt (Punkt 2 in Abbildung 3.17). Im Falle von Assoziationen ist die Informationsdichte relativ gering, was die Darstellung von beiden Multiplizitäten ermöglicht ohne dabei die Übersichtlichkeit zu beeinträchtigen (Punkt 1 in Abbildung 3.17). Die abgeleitete Multiplizität wird in Blau dargestellt.

3.4.3 Zusammenfassung

Mit Hilfe der Modellierungselemente des UML-Klassendiagramms sowie der neu eingeführten Visualisierungstechniken können nun alle aus den Definitionen und Daten abgeleiteten Modellinformationen visualisiert werden. Die strukturellen Informationen wie Typen, Modellattribute und Modellassoziationen werden analog zur Klassen und Assoziationen des Klassendiagramms dargestellt. Die unterschiedliche Herkunft der Modellattribute sowie ihrer Bestandteile (Attributtyp oder Multiplizität) wird mit Farben sowie

räumlicher Anordnung signalisiert und ist somit gut erkennbar. Der Festigungsgrad wird durch die Anzahl der instanziierten Attribute angegeben. Diese wird rechts von dem entsprechenden Modellattribut dargestellt. Bei den Modellattributen aus Definitionen wird daneben die Anzahl der fehlerhaften Attribute angezeigt. Wenn sie gelb ist, dann liegt für das Modellattribut keine *Strict*-Eigenschaft vor (Punkt 2 in Abbildung 3.17). Ihre Präsenz wird mit roter Farbe sowie mit einem Ausrufezeichen dargestellt (Punkt 3 in Abbildung 3.17). Die Visualisierung von zusammengesetzten Attributen (Punkt 2 in Abbildung 3.17) und Assoziationen ist möglich. Wird der Attributtyp bei Modellattributen bzw. Typ der Zielseite bei Assoziationen aus den Daten abgeleitet, werden die abgeleiteten Elemente entsprechend in Blau visualisiert. Im Falle der Assoziationen repräsentiert die Linie den Typ der Assoziation und wird dementsprechend blau dargestellt (Punkt 4 in Abbildung 3.17). Abbildung 3.17 beinhaltet alle der hier beschriebenen Modellierungselemente.

3.5 Layout

Das Layout von UML-Klassendiagrammen befasst sich hauptsächlich mit der Positionierung von Klassen im Diagramm. Es wird dabei eine gute Lesbarkeit und leichte Verständlichkeit des zugrunde liegenden Modells angestrebt. Um diese zu erreichen, berücksichtigen die Layoutalgorithmen bei ihrer Arbeit bestimmte ästhetische Kriterien. Die wichtigsten davon sind nach empirischen Untersuchungen die Minimierung der Anzahl der Kantenkreuzungen und -knicken, horizontale Beschriftungen, Vereinigung der Vererbungspfeile – Vererbungsgabeln, Länge der Kantenwege u.a.[13]. Sie zielen auf eine hohe Übersichtlichkeit der Visualisierung ab. Das im Rahmen dieser Arbeit verwendete Layout erfüllt jedoch primär eine andere Aufgabe [8] [9].

3.5.1 Positionierung der Elemente

In Abschnitt 3.4.1 wurde für die Beschreibung der Modellattribute der Begriff Festigungsgrad eingeführt. Er repräsentiert die Wertigkeit der Modellattribute für die Bestimmung der Gesamtstruktur. Nach einem ähnlichen Prinzip können die Typen beschrieben werden. Sie unterscheiden sich nach der Zusammensetzung ihrer Attribute und durch die Anzahl der Wiki-Seiten des entsprechenden Typs. Aus der Kombination dieser Merkmale lassen sich Metriken für die Positionierung der Typen innerhalb des Diagramms ermitteln. Ein Beispiel für eine solche Metrik ist das Verhältnis der Anzahl der Modellattribute aus Definitionen zur Gesamtanzahl der Modellattribute eines Typs. Verschiedene Varianten der Metriken für die Positionierung der Typen werden im Abschnitt 3.5.2 analysiert.

Das Layout für die Visualisierung des aus Hybrid Wikis abgeleiteten Modells erfolgt also primär auf Basis der Metriken. Diese Metriken drücken die Wertigkeit der einzelnen Typen für das Gesamtmodel nach unterschiedlichen Kriterien aus. So beschreibt z.B. das o.g. Verhältnis von der Anzahl der Modellattributen aus Definitionen zur Gesamtanzahl der Modellattribute, im welchen Maße ein Typ durch Definitionen beschrieben wird – den Definitionsgrad des Typs. Ist dieses Verhältnis nahe 1, so stellt der zugehörige Typ ein fest definiertes Element der Struktur dar. Überwiegen jedoch die aus den Daten abgeleiteten Modellattribute, so bildet der Typ einen undefinierten *diffusen* Teil der Gesamtstruktur.

Die Positionierung der Typen erfolgt analog zu dem in 3.4.1.1 vorgestellten Ansatz zur

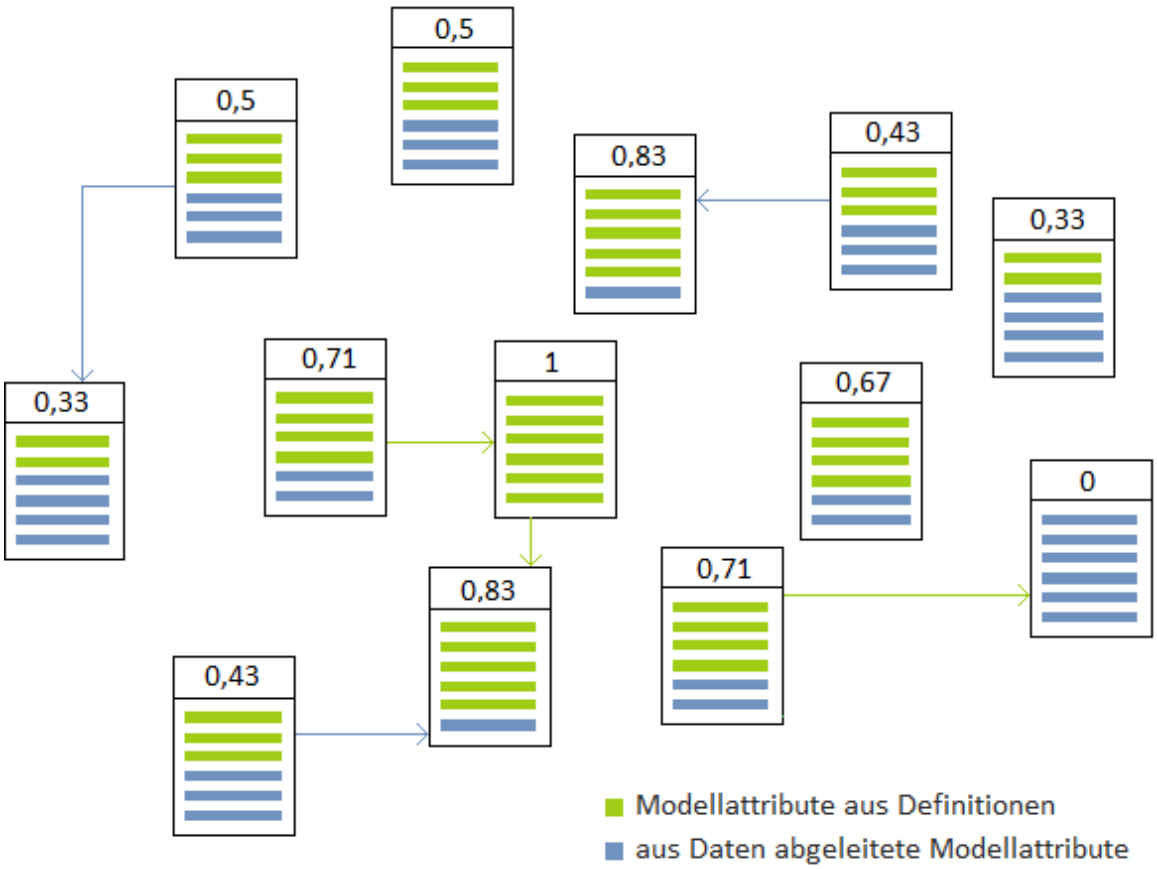


Abbildung 3.18: Layout auf Basis des Definitionsgrads der Typen

Positionierung der Modellattribute. Die Typen mit den höchsten Metrikwerten werden mittig platziert. Die steigende Entfernung von der Mitte signalisiert niedrigere Werte. Abbildung 3.18 zeigt eine schematische Darstellung des Layouts auf Basis des o.g. Definitionsgrads. Assoziationen werden wie Modellattribute in die Berechnung einbezogen. Anstelle der Typnamen stehen die entsprechenden Metrikwerte.

3.5.2 Metriken

Es sind unterschiedliche Metriken für die Positionierung der Typen denkbar. Die resultierenden Layouts unterscheiden sich in der Übersichtlichkeit des Diagramms. Zwar besitzt die Kommunikation der Wertigkeit von Typen für das Layout eine höhere Priorität, jedoch muss die Visualisierung lesbar und verständlich sein. Als Gütemaß für die entstandenen Layouts wurden die Anzahl der Kantenkreuzungen, -knicke sowie die Länge der Kantenwege gewählt, denn sie stellen für die Lesbarkeit des Diagramms die bedeutendsten ästhetischen Kriterien dar [13].

Die in der Abbildung 3.18 zugrunde liegende Metrik – der Definitionsgrad des Typs – kann weiterentwickelt werden:

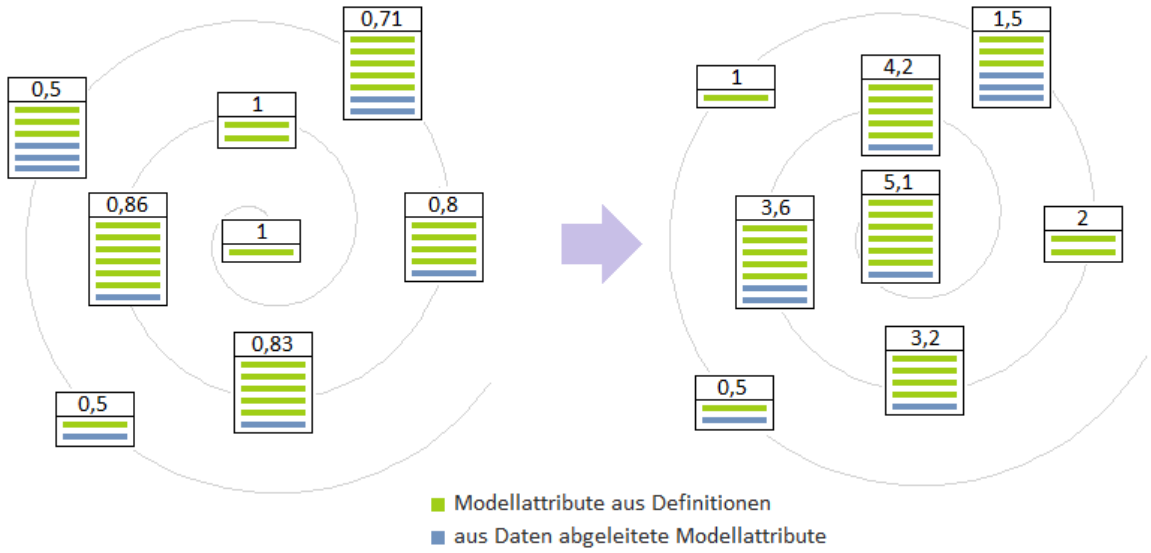


Abbildung 3.19: Verwendung der modifizierten Formel für den Definitionsgrad

$$\frac{(\textit{Anzahl der Modellattribute aus Definition})^2}{\textit{Gesamtanzahl der Modellattribute}}$$

Hier wird die Anzahl der Modellattribute aus Definitionen durch das Potenzieren stärker gewichtet. Durch diese Erweiterung wird es vermieden, dass Typen, die aus einem einzigen Modellattribut mit Definition bestehen und somit den Definitionsgrad 1 besitzen, eine höhere Wertigkeit haben als Typen, bei denen 8 von 10 Modellattribute aus Definitionen stammen. Es wird also nicht nur das Verhältnis, sondern auch die Anzahl der Modellattribute (stärker) berücksichtigt. Abbildung 3.19 zeigt die Auswirkung der Formeländerung auf das Layout.

Die Metrik kann durch die Berücksichtigung der Attributtypen modifiziert werden. So werden die Modellattribute mit Typdefinition stärker gewichtet als kombinierte Modellattribute – Modellattribute, die zwar eine Definition besitzen, deren Typ jedoch aus der Datenbasis abgeleitet wird. Die Formel für diese Metrik könnte folgendermaßen aussehen:

$$\frac{\textit{Anz. der Modellattr. mit Typdefinition}*1{,}5 + \textit{Anz. der Modellattr. ohne Typdefinition}}{\textit{Gesamtanzahl der Modellattribute}}$$

Die o.g. Varianten der Metrik Definitionsgrad besitzen einen Nachteil: die besondere Stellung der Typen mit ein- und ausgehenden Assoziationen wird nicht berücksichtigt. Die resultierenden Diagramme besitzen also viele Kantenkreuzungen, -knicke, lange Kantenwege u.a., was die Übersichtlichkeit und Nützlichkeit der Visualisierung drastisch reduziert. Typen mit Assoziationen sollen möglichst mittig positioniert werden [8]. Dazu wird folgende Formel vorgeschlagen:
$Assoc_{Definition}$:= Anzahl der Assoziationen aus Definitionen
$Assoc_{Daten}$:= Anzahl der Assoziationen ohne Definition

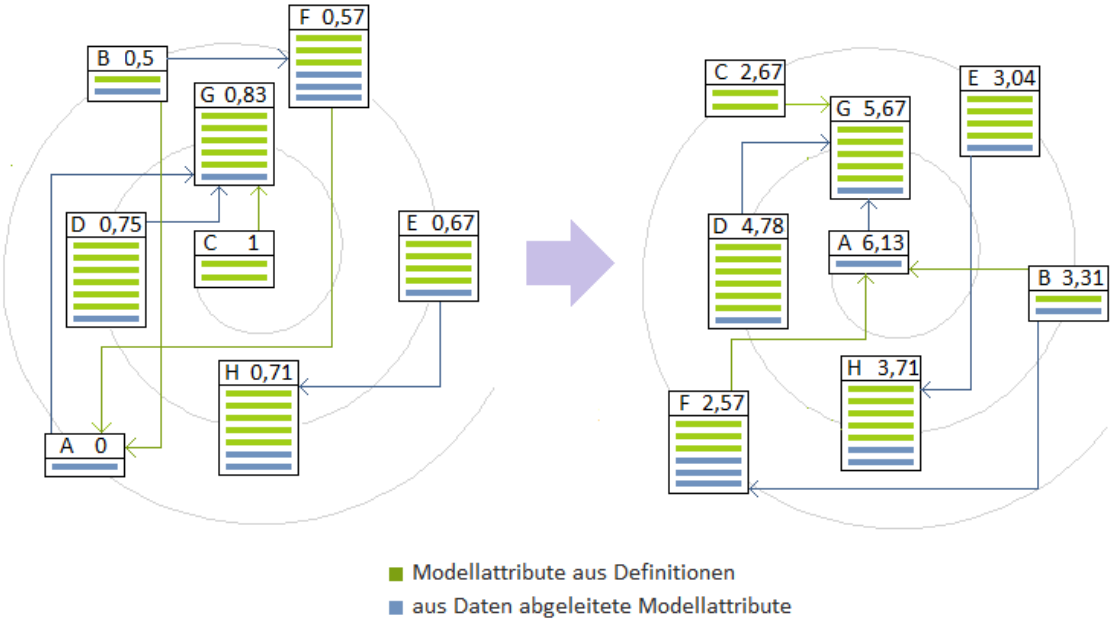


Abbildung 3.20: Layout mit Berücksichtigung der Assoziationen

$Assoc_E :=$ Anzahl der eingehenden Assoziationen
$Attr_{Definition} :=$ Anzahl der Modellattribute aus Definitionen

$$\frac{(Assoc_{Definition} * 2 + Assoc_{Daten} * 1,5 + Assoc_E)^2 + Attr^2_{Definition}}{\textit{Gesamtanzahl der Attribute und Assoziationen}}$$

Diese Formel berücksichtigt neben dem Definitionsgrad auch die Assoziationen eines Typs. Es werden sowohl eingehende, als auch ausgehende Assoziationen betrachtet. Des Weiteren besitzen die Assoziationen mit Definitionen eine stärkere Gewichtung. Die Gewichtungsfaktoren für AssocDefinition, AssocDaten und AssocE (also 2, 1.5 und 1) wurden durch Ausprobieren ermittelt. Diese Formel könnte zusätzlich durch unterschiedliche Gewichtung der kombinierten Assoziationen und Modellattribute verfeinert werden, da sie aber ohnehin ziemlich komplex ist, wurde aus Gründen der Einfachheit darauf verzichtet. Abbildung 3.20 zeigt zwei Layouts eines Modells mit unterschiedlichen Metriken. Rechts von dem Typnamen beinhalten die Typen entsprechende Metrikwerte. Das linke Layout verwendet den einfachen Definitionsgrad (Anzahl der Modellattribute mit Definitionen/ Gesamtanzahl der Attribute). Rechts werden die Assoziationen in die Berechnung einbezogen. Dadurch erhält man kürzere Kantenwege und weniger Kantenkreuzungen, da die Typen mit mehreren Assoziationen in der Mitte positioniert sind.

Durch die Berücksichtigung der Assoziationen erzielt die Positionierung von Typen bessere Ergebnisse in Bezug auf die Übersichtlichkeit. Deswegen wird diese Formel für die Implementierung verwendet.

4 Umsetzung

Die Umsetzung der entwickelten Modellableitungs- und Darstellungskonzepte setzt sich aus drei Komponenten zusammen. Als erstes wurde die Modellableitungsstrategie umgesetzt, die aus Hybrid Wikis die notwendigen Informationen extrahiert. Diese wird im Abschnitt 4.1 beschrieben. Im nächsten Schritt wurde die webbasierte Visualisierung der ermittelten Informationen realisiert. Ihre Funktionsweise und die Implementierungsdetails beinhaltet das Kapitel 4.2. Anschließend wurden die beiden Komponenten in die Collaboration-Plattform Tricia integriert, um sie bequem und ohne Medienbrüche direkt aus Tricia nutzen zu können. Die Integration wird im Kapitel 4.3 vorgestellt.

4.1 Modellableitung

Die Collaboration-Plattform Tricia stellt mit Plugins eine einfache Erweiterungsmöglichkeit der eigenen Funktionalität zur Verfügung [18]. Die Modellableitung wurde aus diesem Grund als Tricia-Plugin umgesetzt. Abbildung 4.1 zeigt die Struktur des Plugins *hybridWiki2model*.

Die gesamte Funktionalität der Modellableitungsstrategie ist in zwei Klassen gekapselt: Definitions2Model und CombinedModel (Punkt 1 in Abbildung 4.1). Sie ist sequenziell aufgebaut und entspricht im Wesentlichen der in 3.1 und 3.3 beschriebenen Vorgehensweise. Der Modellableitungsprozess wird durch den Aufruf der URL $../initModel?wikiName =< NamedesWikis >$ angestoßen. Da die Modellableitung pro Wiki erfolgt, erwartet der Aufruf einen Wiki-Namen als Parameter. Durch den URL-Aufruf wird die Klasse initModelHandler (Punkt 2 in Abbildung 4.1) instanziiert. Sie erzeugt das HybridWiki2modelPlugin (Punkt 3 in Abbildung 4.1). Abbildung 4.2 zeigt einen Ausschnitt aus dem Sequenzdiagramm, das diese Ereignisse darstellt. Die eigentliche Modellableitung beginnt erst im nächsten Schritt. Da die Modellableitungsfunktionalität sich auf zwei Klassen konzentriert, macht es wenig Sinn, diese anhand eines Klassendiagramms zu erklären. Ein Sequenzdiagramm eignet sich für diesen Zweck ebenfalls nicht, denn die Implementierung beinhaltet in den zwei erwähnten Klassen eine Vielzahl von Objektrekursionen. Eine Objektrekursion liegt vor, wenn bei der Ausführung einer Methode eine andere Methode des gleichen Objekts aufgerufen wird [19, S 224]. Aus diesen Gründen wird die Implementierung anhand eines UML2 Aktivitätsdiagramms veranschaulicht. Das Diagramm beinhaltet nur die wichtigsten zum Verständnis notwendigen Methodenaufrufe. Die horizontalen Teilbereiche repräsentieren die Klassen *Definitions2Model* und *CombinedModel*.

Die folgenden Nummerierungspunkte entsprechen den Punkten in der Abbildung 4.3

1. An dieser Stelle endet das Sequenzdiagramm aus der Abbildung 4.2. Hier wird die CombinedModel-Klasse aus dem HybridWiki2ModelPlugin instanziiert. Als Parameter bekommt der Konstruktor den Wiki-Namen.

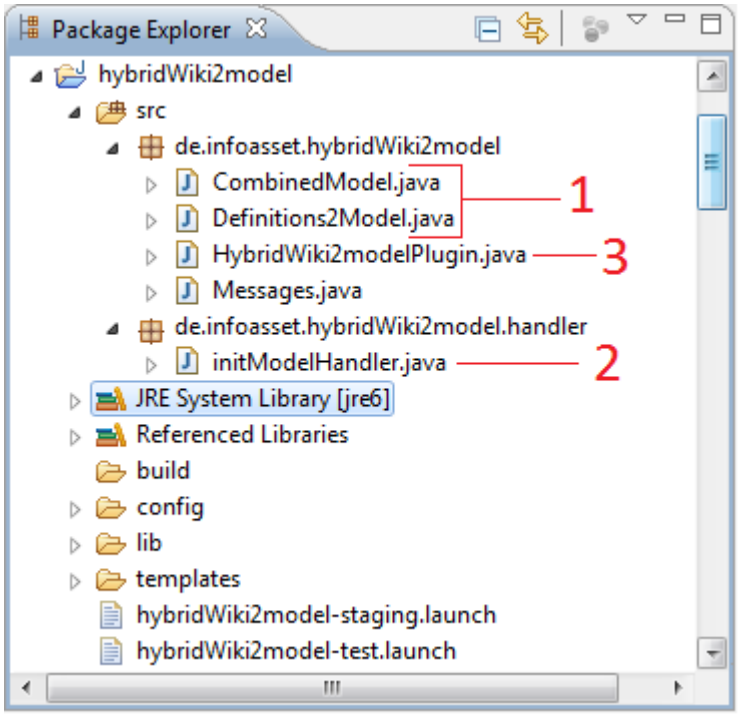


Abbildung 4.1: Struktur des Plugins zur Modellableitung

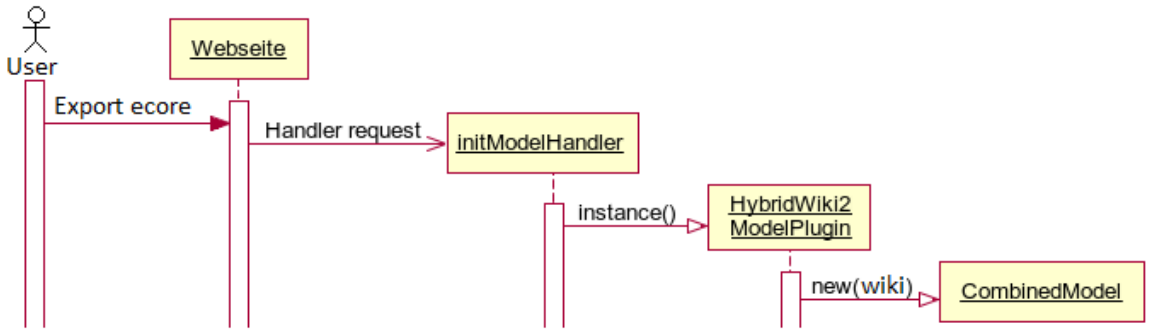


Abbildung 4.2: Ausschnitt aus dem Sequenzdiagramm für die Modellableitung

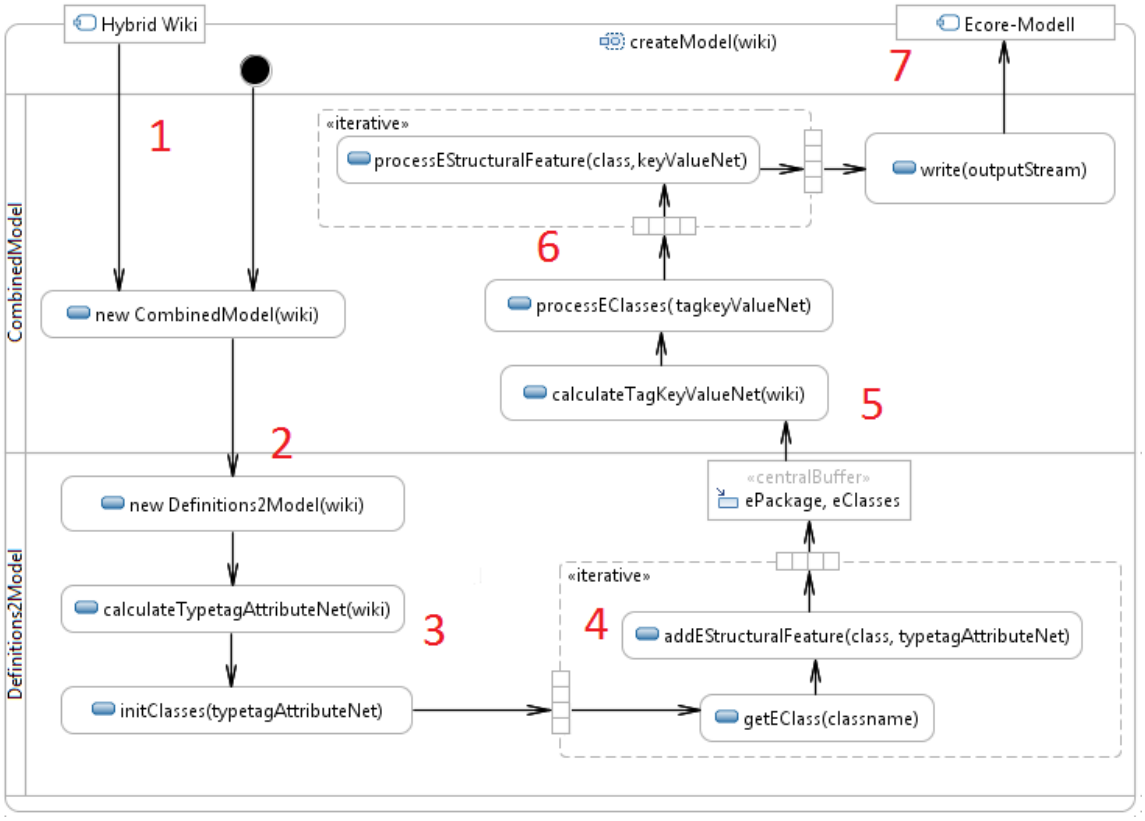


Abbildung 4.3: Umsetzung der Modellableitung dargestellt mit UML2 Aktivitätsdiagramm

2. Bei der Erstellung des kombinierten Modells wird zuerst das Modell aus den Definitionen abgeleitet. Dazu wird die Klasse Definitions2Model instanziiert.

3. Die Methode calculateTypetagAttributeNet() liefert ein assoziatives Array, das als Schlüssel die TypeTags-Namen und als Werte die Listen mit den zugehörigen Attributdefinitionen besitzt. Die Datenstruktur sieht folgerndermaßen aus:
$Map < String, List < HybridPropertyDefinition >>$. Diese Daten werden als Parameter an die Methode initClasses() weitergegeben.

4. initClasses() iteriert über die Schlüssel und erzeugt aus ihnen mit Hilfe der Methode getEClass() die entsprechenden Klassen im Ecore-Format. Die leeren Klassen werden zusammen mit den zugehörigen Attributdefinitionen an die Methode addEStructuralFeature() übergeben. Hier werden aus den Attributdefinition entweder Assoziationen oder Modellattribute im Ecore-Format erzeugt und der entsprechenden Klasse zugeordnet. Sobald eine Assoziationen mehrere Typen besitzt, wird aus diesen Typen ein Supertyp gebildet und als Zieltyp in die Assoziation eingesetzt. Die Attribute und Assoziationen werden an dieser Stelle mit zusätzlichen Informationen versehen: ob Attribut und Attributtyp definiert sind und die Attributdefinition „strict“ ist. Nach diesem Schritt liegt das Modell aus den Definitionen vor. Es wird nun an das kombinierte Modell zur Weiterverarbeitung übergeben.

   Anmerkung: die aus Hybrid Wikis abgeleiteten Informationen werden im Ecore-Format (Abschnitt 2.3) erfasst. Wie in Abschnitt 2.3 bereits erwähnt, orientiert sich Ecore jedoch an den Modellierungskonzepten der UML-Klassendiagramme und besitzt daher keine Repräsentationen für die Hybrid Wikis-spezifischen Informationen. Deswegen gab es Überlegungen, eine eigene Datenstruktur als Zielformat des abgeleiteten Modells zu entwerfen. Nun bietet aber Ecore mit Extended Metadata-Annotationen ein passendes Mittel, um zusätzliche Informationen zu speichern. Aus diesem Grund wurde auf eine neue Datenstruktur verzichtet.

5. Hier wird ähnlich zum Punkt 3 ein assoziatives Datenfeld mit Typen als Schlüssel und Attributlisten als Werten errechnet. Die Typen und Attribute stammen aber im Gegensatz zu Punkt 3 nicht aus den Definitionen, sondern von den Wiki-Seiten. Ein weiterer Unterschied zu Punkt 3 liegt darin, dass zu einem Attribut alle auf den Wiki-Seiten vorhandenen Werte angegeben werden.

6. Die Methode processEStructuralFeature() betrachtet die aus den Definitionen stammenden Typen und Attribute und ergänzt diese um fehlende Informationen. Es werden entweder neue Attribute hinzugefügt oder die existierenden um fehlende Bestandteile (Typ, Multiplizität) erweitert. Außerdem werden hier die Anzahl der Instanzen sowie die abgeleitete Multiplizität ermittelt.

7. Anschließend wird das Ergebnis der Modellableitung an den Benutzer übergeben. Abbildung 4.4 zeigt eine beispielhafte Ecore-Datei aus der Modellableitung.

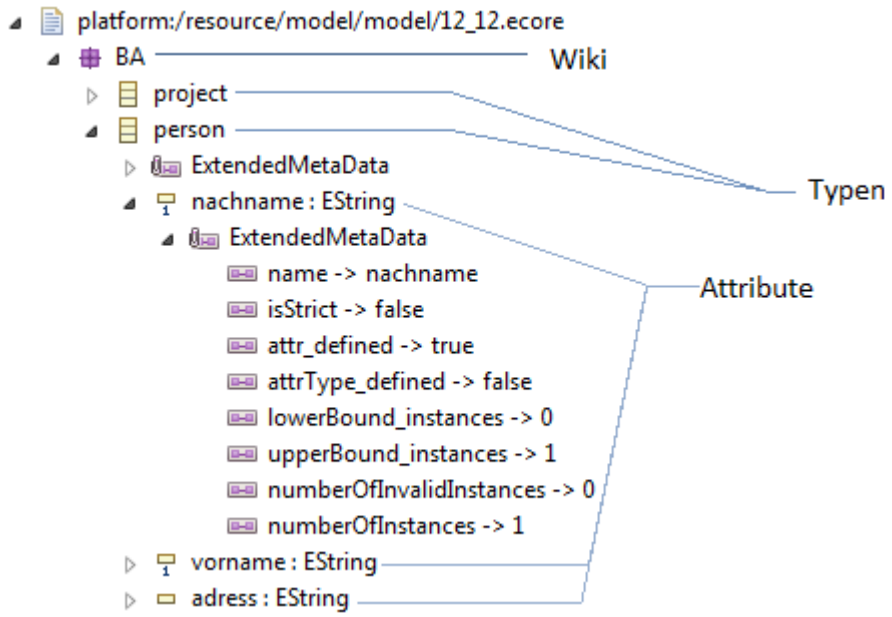


Abbildung 4.4: Ergebnis der Modellableitung

4.2 Visualisierung

Dieses Kapitel beschreibt die prototypische Implementierung der Modelldarstellung. Es existieren nur wenige Anforderungen daran. Die Visualisierung soll web-basiert sein und möglichst viele Informationen aus dem Modell übersichtlich und verständlich kommunizieren. Außerdem wird keine Interaktivität verlangt. Diese Tatsachen ermöglichen einen hohen Freiheitsgrad in Bezug auf die verwendeten Mitteln. Es existiert eine Fülle von Technologien, die eine web-basierte Visualisierung von Informationen ermöglichen. Welche davon und warum diese im Rahmen der Arbeit verwendet wurde, beschreibt das Kapitel 4.2.1 Wie die Technologie zum Einsatz kommt und die Implementierungsdetails werden im Abschnitt 4.2.2 vorgestellt.

4.2.1 Auswahl der Basistechnologie

Die Grundlage der visuellen Darstellung des abgeleiteten Modells bilden die Basiselemente des UML-Klassendiagramms. Mit ihrer Hilfe können die strukturellen Bausteine des Modells visualisiert werden. Um die Hybrid-Wikis-spezifischen Informationen darzustellen, werden sie um neue Konzepte erweitert.

Am Anfang war es geplant, die Visualisierung mit Hilfe einer JavaScript-Bibliothek zum Zeichnen wie z.B. Raphaël umzusetzen. Damit lassen sich einfache graphische Formen wie Vierecke, Linien usw. darstellen. Da aber einige Open-Source-Projekte existieren, die sich mit web-basierter Modellierung und Darstellung von UML-Klassendiagrammen beschäftigen, wurde die Entscheidung getroffen, die Visualisierung auf Basis eines Open-Source-Modellierungswerkzeugs umzusetzen. Dies erspart eine Menge Arbeit, die im umgekehrten Falle in die Implementierung der Basiselemente des Klassendiagramms inves-

tiert werden müsste.

Im ersten Schritt wurden einige Projekte ausgewählt, die den Anforderungen entsprachen. Die wichtigsten davon waren die Quelloffenheit der Software sowie der web-basierte Ansatz. Außerdem sollten die Kandidaten nicht allzu komplex sein. Anschließend wurden die Projekte auf die Verwendbarkeit analysiert. Es kamen folgende Modellierungswerkzeuge in die engere Auswahl:

- *UmlCanvas* ist ein mit JavaScript implementiertes Modellierungstool. Die Erzeugung von Klassendiagrammen wird über die JavaScript API sowie mit Hilfe einer textuellen DSL (domain specific language) gesteuert. Zum Erstellen von Graphiken wird das HTML5-Element Canvas verwendet. Es ermöglicht das dynamische Rendern von Rastergraphiken und wird von allen gängigen Internetbrowsern unterstützt. Canvas wird über JavaScript gesteuert [6][1]. Das Projekt UmlCanvas existiert seit 2009 und liegt in einer produktiven Version vor. Es wird weiterentwickelt (letztes Update am 18.11.2011).
- *RaphUml* ist eine kleine JavaScript-Bibliothek zum Erstellen von Klassendiagrammen. Sie steht unter MIT-Lizenz. Die Modellierung erfolgt mit Hilfe einer textuellen DSL. RaphUml liegt derzeit in der Version 0.3 vor. Ein automatisches Layout der Klassendiagramme ist in Planung. Zurzeit müssen aber die Klassen noch manuell positioniert werden. Die letzte Änderung am Projekt wurde am 3.09.2010 vorgenommen. RaphUml verwendet zum Erstellen von Graphiken die JavaScript-Bibliothek *Raphaël*. Raphaël arbeitet mit Vektorgraphiken in SVG und VML-Format und wird von allen gängigen (sowie älteren) Internetbrowsern unterstützt. Die JavaScript-Bibliothek vereinfacht die Handhabung von SVG. RaphUml setzt außerdem die jQuery-Bibliothek ein, die nützliche implementierungstechnische Funktionen zur Verfügung stellt [5][4].
- Ein weiteres Tool, das als mögliche Basis für die Visualisierung betrachtet wurde, ist die am Lehrstuhl für Software Engineering for Business Information Systems an der TUM verwendete Modellierungssoftware *Oryx*. Oryx ist ein WYSIWYG-Editor für UML. Es überschreitet aber den erforderlichen Funktionsumfang und würde in einem erheblichen Mehraufwand resultieren. Ein Vorteil von Oryx ist jedoch die Möglichkeit Ecore-Dateien zu visualisieren.

Für die Umsetzung der Visualisierung wurde RaphUml ausgewählt. Sein großer Vorteil ist die Verwendung von Vektorgraphiken, da es bei einer hohen Anzahl der Modellelemente eine Skalierung notwendig sein kann. Vektorgraphiken ermöglichen diese ohne Qualitätsverluste. Außerdem ist die Komplexität von RaphUml geringer als die von UmlCanvas. RaphUml beinhaltet um ein Vielfaches weniger Codezeilen. Das Fehlen eines automatisierten Layouts kann als Vorteil angesehen werden, da eine neue Layoutstrategie umgesetzt wird.

4.2.2 Implementierungsdetails

Die Umsetzung der Visualisierung wurde auf Basis von RaphUml durchgeführt. Zur Darstellung des Modells werden die JavaScript-Bibliotheken Raphaël, die modifizierte RaphUml (hybridUml genannt) sowie jQuery in die Webseite eingebunden. Die Erstellung

eines Klassendiagramms erfolgt mit Hilfe einer DSL. Die ursprüngliche DSL wurde durch Hybrid-Wikis-spezifische Elemente erweitert. So sieht z.B. die Deklaration des Typs person aus:

```
type person (15)
--
name:Text {multiplicity=(1),isStrict=false,attrDefined=true...}
adresse:Text {multiplicity=(0..*),isStrict=true,attrDefined=true...}
gebDatum:Date {multiplicity=(1),isStrict=true,attrDefined=false...}
```

Assoziationen werden nach folgendem Schema deklariert:

```
assoc <Quelltyp> <Attributname> <Zieltyp> {zus. Informationen}
```

Für Vererbungsbeziehungen wird *extends* als Rollenname verwendet. So sieht beispielsweise die Assoziation *arbeitgeber* aus der Abbildung 3.17 aus:

```
assoc person arbeitgeber unternehmen{multiplicity=(0..*),isStrict...}
```

Der Parser für die Visualisierung wurde mit Hilfe des JS/CC Parser Generators erstellt. Der generierte Quellcode wurde in die hybridUml-Bibliothek integriert. JS/CC ist eine plattformunabhängige Open-Source-Software zur Entwicklung von Tokenizer (Lexer) und Parser. Sie liefert den JavaScript-Code als Ergebnis [2].

JavaScript ist zwar keine klassenbasierte objektorientierte Sprache, verfügt aber über Konzepte, die Klassen simulieren können [10]. Aus diesem Grund wird das Datenmoell von hybridUml mit einem Klassendiagramm veranschaulicht. Abbildung 4.5 zeigt eine vereinfachte Darstellung des Datenmodells. Es existieren weitere Methoden und Objekte, die implementierungstechnische Aspekte decken und im Diagramm nicht enthalten sind. So werden im Diagramm z.B. keine Raphaël-Erweiterungen, Parser-Objekte u.a. dargestellt.

HybridUml verfügt über Repräsentationen für die aus Hybrid Wikis abgeleiteten Modellelemente. In Abbildung 4.5 findet man die Typen, Attribute, Assoziationen und Enums. Die Modellelemente werden beim Parsen der textuellen Eingabe identifiziert. Dabei werden ihre Repräsentationen in hybrtidUml initialisiert. Die ursprüngliche Datenstruktur aus RaphUml wurde stark modifiziert. Neben den Hbird-Wikis-spezifischen Erweiterungen wurden die Funktionalitäten zur Darstellung von Enumerationen sowie reflexiven Assoziationen hinzugefügt. Die Methoden zum Zeichnen wurden komplett überarbeitet.

Nachdem die Repräsentationen der Modellelemente in hybridUml initialisiert wurden, wird der layouter (Punkt 1 in Abbildung 4.5) gestartet. Er berechnet die in Abschnitt 3.5.2 vorgestellten Metriken für die Typen. Anhand der berechneten Metrik wird die Position der Typen auf dem Diagramm ermittelt. Abhängig von der Anzahl und Länge der zugehörigen Modellattribute wird ihre Höhe und Breite bestimmt. Zum Zeichen der Typen ist die Methoden draw() in der Klasse Type (Punkt 3 in der Abbildung 4.5) zuständig. Die Enums werden beim Layout nicht berücksichtigt. Im übrigen werden sie analog zu den Typen visualisiert.

Die Funktionalität zur Visualisierung von Assoziationen stellt die Klasse straightLineRouter (Punkt 2 in Abbildung 4.5) zur Verfügung. Nachdem alle Typen auf dem Diagramm

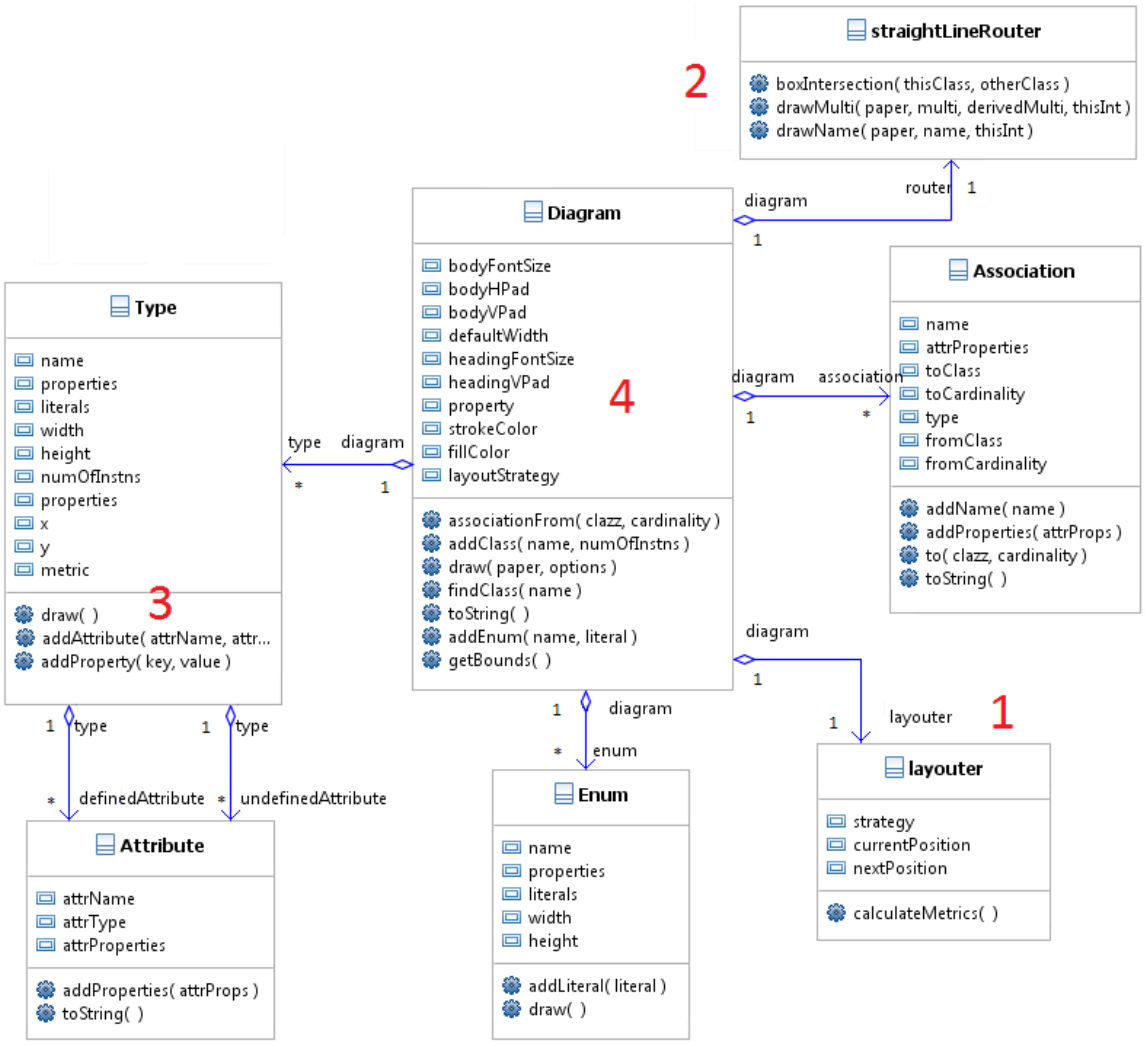


Abbildung 4.5: Datenstruktur von hybridUml

bereits dargestellt sind, werden aus ihren Positionen die Koordinaten für die Assoziationen berechnet. HybridUml kenn zwei Arten von Assoziationen: die gerichteten Assoziationen sowie die Vererbungsbeziehungen.

Das Aussehen vom Diagrammen ist konfigurierbar. Es können die Füll- und Linienfarben, die Schriftgröße sowie die Mindestbreite für Typen angepasst werden. Die entsprechenden Attribute befinden sich in der Klasse Diagramm (Punkt 4 in Abbildung 4.5).

4.3 Integration in Tricia

Um die Modellableitung zusammen mit der Visualisierung bequem nutzen zu können, mussten die Medienbrüche zwischen den beiden Tools beseitigt werden. Wie aus den Abschnitten 4.1 und 4.2 bereits bekannt, wurden die beiden Werkzeuge separat voneinander

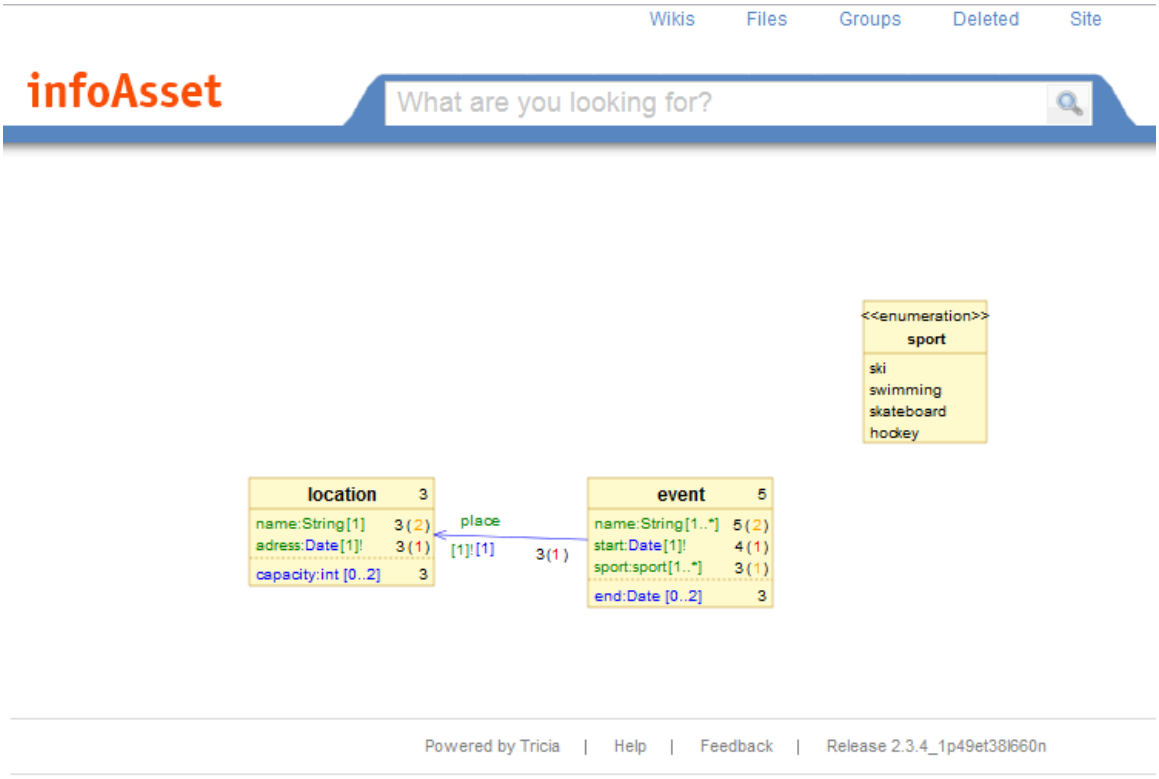


Abbildung 4.6: Modellvisualisierung in Tricia

implementiert. Dies ermöglichte eine leichtere Beherrschbarkeit der Teilprojekte. Außerdem ermöglicht die textuelle Benutzerschnittstelle von hybridUml einfaches und schnelles Testen der Software.

Tricia verfügt über nützliche Funktionen zur Ausgabe von formatierten Inhalten. Diese Funktionen werden vor allem zur Generierung von Webseiten verwendet. Die von Tricia generierten HTML-Seiten besitzen in der Regel ein Template, das mit notwendigen Informationen ergänzt wird. Auf dieser Idee basiert die Integration der Modellableitung und der Visualisierung in Tricia. Anstatt wie in 4.1 beschrieben nach der Modellableitung eine Ecore-Datei zurückzugeben, wird aus dem Modell mit Hilfe des Tricia Templating Systems eine textuelle Ausgabe nach den in 4.2.2 beschriebenen Regeln generiert. Diese Eingabe wird von hybridUml geparst und anschließend visualisiert. Dafür wurden die notwendigen JavaScript-Bibliotheken in das entsprechende Template eingebunden. Somit ist die Benutzung der Modellableitung aus Hybrid Wikis sowie der Visualisierung des abgeleiteten Modells ohne Medienbrüche direkt aus Tricia möglich. Abbildung 4.6 zeigt ein visualisiertes Modell in Tricia.

5 Zusammenfassung

Im Rahmen dieser Bachelorarbeit wurde die Modellierung von emergenten Informationsstrukturen am Beispiel der Hybrid Wikis analysiert und prototypisch umgesetzt. Eine Besonderheit von Hybrid Wikis liegt in der Vereinigung von zwei unterschiedlichen Modellierungskonzepten. Es wurde eine Modellableitungsstrategie entworfen und umgesetzt, die die unterschiedlichen Konzepte berücksichtigt und ein kombiniertes Modell liefert. Zur Darstellung des resultierenden Modells wurden neue Visualisierungskonzepte entwickelt. Als Grundlage für die Konzepte dient das UML-Klassendiagramm. Die Modellvisualisierung wurde auf Basis einer Open-Source-Modellierungssoftware implementiert. Anschließend wurden die Modellableitung und -visualisierung in die Collaborations-Plattform Tricia integriert.

5.1 Probleme und Schwierigkeiten

Im Laufe der Arbeit traten ab und zu einige Schwierigkeiten und Probleme auf. So führte z.B. ein Softwarefehler in Hybrid Wikis zu einer fehlerhaften Implementierung der Modellableitungsstrategie. Die Zieltypen der Assoziation in Attributdefinitionen waren „und"-verknüpft – die referenzierten Wiki-Seiten mussten alle angegebenen Typen besitzen. Dieser Sachverhalt konnte durch die Mehrfachvererbung repräsentiert werden. Die Idee wurde umgesetzt und die Verwendung der Vererbung wurde auf alle Aspekte der Modellableitungsstrategie übertragen. Eine weitere Schwierigkeit bereitete die mangelnde Dokumentation der verwendeten Werkzeuge. So musste beispielsweise die Funktionalität des im Abschnitt 4.2.2 erwähnten Parser-Generators hauptsächlich durch Ausprobieren angeeignet werden, was in einem erheblichen Zeitaufwand resultierte. Eine Schwierigkeit stellte die kontinuierliche Entwicklung der Hybrid Wikis und Tricia dar. So musste die Modellableitungstrategie an die Änderungen der Basissoftware angepasst werden, da es z.B. neue Datentypen eingeführt wurden, die bei der Modellableitung ebenfalls erfasst werden müssen.

5.2 Ausblick

Aus dieser Arbeit ergaben sich einige Ideen und Verbesserungsvorschläge für Hybrid Wikis und für die im Rahmen der Arbeit entwickelten Werkzeuge.

Eine mögliche Erweiterung für Hybird Wikis stellt die Generalisierung dar. Derzeit können die Vererbungsbeziehungen unter den Typen simuliert werden – etwa durch die gleichzeitige Angabe des allgemeineren Typs – die Generalisierung ist aber nicht eindeutig. Dieses Konzept würde die Modellierung effizienter machen.

Eine Verbesserungsmöglichkeit sowohl für die Hybrid Wikis als auch für die im Rahmen dieser Arbeit entwickelte Modellableitung stellt die Ermittlung der am häufigsten

verwendeten Datentypen für ein Attribut. Derzeit wird für die aus den Daten abgeleiteten Modellattribute der allgemeinste unter den instanziierten Attributen vorhandene Datentyp gewählt – Text. Die Verbesserungsmöglichkeit besteht darin, den am meisten verwendeten Datentyp eines Attributs zu ermitteln und den sowohl in der Modellableitung als auch für die Attributvorschläge auf den Wiki-Seiten zu verwenden. So könnte z.B. ein Datentyp, der bei mindestens 50 Prozent der Attributinstanzen angegeben ist, als dominanter Datentyp des Attributs in das abgeleitet Modell einfließen. Diese Idee benötigt aber für das Modell zusätzlich eine Visualisierungsmöglichkeit für erratene Informationen.

Die in 4.2.1 beschriebene Integration in Tricia könnte effizienter Implementiert werden. Der Zwischenschritt nach der Modellableitung – die Generierung der textuellen Repräsentationen – stellt einen Umweg dar. Am Lehrstuhl für Software Engineering for Business Information Systems (sebis) an der Technischen Universität München wird zur Visualisierung von Informationen ein Tool verwendet, das direkt aus Java Code web-basierte Visualisierungen erstellt – das SyCaTool. Um die Visualisierung mit SyCaTool zu ermöglichen, müssen die im Rahmen dieser Arbeit entwickelten Visualisierungskonzepte in das SyCaTool übertragen werden.

Literaturverzeichnis

[1] The canvas element. http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html, (abgerufen am 29.08.2011).

[2] Js/cc parser generator. http://jscc.jmksf.com/, (abgerufen am 10.09.2011).

[3] Omg unified modeling languagetm (omg uml), infrastructure version 2.4.1.

[4] Raphaël - javascript library. http://raphaeljs.com/, (abgerufen am 10.12.2011).

[5] Raphuml. https://github.com/jkrasnay/raphuml, (abgerufen am 30.08.2011).

[6] Umlcanvas. https://github.com/christophevg/UmlCanvas, (abgerufen am 29.08.2011).

[7] *EMF: Eclipse Modeling Framework, Second Edition*. 2004.

[8] Holger Eichelberger. Aesthetics of class diagrams. *First IEEE International Workshop on Visualizing Software for Understanding and Analysis, Vissoft 2002, Paris*, pages 23–31, 2002.

[9] Holger Eichelberger. Uml class diagrams - state of the art in layout techniques. *International Workshop on Visualizing Software for Understanding and Analysis, Amsterdam*, pages 30–34, 2003.

[10] David Flanagan. *JavaScript: Das umfassende Referenzwerk, 5. Auflage*. 2007.

[11] C. Neubert F.Matthes and A. Steinhoff. Hybridwikis: Empowering users to collaboratively structure information. *In 6th International Conference on Software and Data Technologies (ICSOFT)*, 2011.

[12] Martin Fowler. *UML Konzentriert, 3. Auflage*. 2004.

[13] J.-A. Allder und D. Carrington H. C. Purchase. Graph layout aesthetics in uml diagrams: User preferences. *Journal of Graph Algorithms and Applications*, 2002.

[14] F. Matthes and C. Neubert. Enabling knowledge workers to collaboratively add structure to enterprise wikis. *In 12th European Conference on Knowledge Management - ECKM 2011*, 2011.

[15] A. McAfee. Enterprise 2.0: the dawn of emergent collaboration. *MIT Sloan Management Review*, 2006.

[16] E.J. Sinz O.K. Ferstl. *Grundlagen der Wirtschaftsinformatik.5 Auflage*. 2006.

[17] Bernhard Rumpe. *Modellierung mit UML. Springer 2004*. 2004.

[18] C. Neubert S. Buckl, F. Matthes and C. Schweda. A lightweight approach to enterprise architecture modeling and documentation.