Automatische Generierung objektorientierter Strukturen

Mario Jeckle
DaimlerChrysler Forschungszentrum Ulm
mario.jeckle@daimlerchrysler.com

Lore Kern-Bausch
Fachhochschule Augsburg
kern@informatik.fh-augsburg.de

1 Motivation und technische Herausforderung

Die zunehmende Komplexität moderner Systeme förderte die Akzeptanz des objektorientierten Paradigmas. Seine Abstraktionsmittel erlauben es, große Systeme übersichtlich zu gliedern. Neben objektorientierten Programmiersprachen entstanden seit Mitte der 1980er Jahre graphische Notationen zur Entwicklung solcher Systeme. Der mittlerweile bekannteste, und in der Praxis bedeutendste, unter ihnen ist die *Unified Modeling Language* (UML) [1].
Es zeigt sich, daß mit zunehmender Größe und Komplexität der Systeme, die Komplexitätsreduktion auf der Modellebene durch Komplexitätssteigerung auf der Sprachebene erkauft wurde.
Folglich nimmt die Möglichkeit zukünftige Anwender des entstehenden Systems in den Entwurfsprozess miteinzubeziehen bei Verwendung objektorientierter Modellierungssprachen eher ab. Damit sinkt letztlich die Qualität des entstehenden objektorientierten Systems, da das vorhandene Fachwissen nicht direkt in den Modellierungsprozess einfließen kann, sondern eines zusätzlichen interpretierenden Schrittes bedarf.
Darüber hinaus haben objektorientierte Modellierungssprachen, durch die Vielzahl angebotener Konzepte, mittlerweile ein Komplexitätsniveau erreicht, das die korrekte Abbildung großer Systeme erschwert.

Sowohl Entwicklungsprozesse als auch -werkzeuge sind daher gezwungen zunächst den Bruch zwischen Fachleuten und Systementwicklern zu überbrücken und im Weiteren die Umsetzung des erfassten Wissens in adäquate IT-Strukturen zu unterstützen.

Das Papier stellt einen Ansatz zur Generierung objektorientierter Strukturen aus konzeptuellen Schemata vor. Hierbei wird besonderes Augenmerk auf die einfache Einsetzbarkeit der verwendeten konzeptuellen Modellierungsnotation (graphisch und textuell) und -methode, dem E3R-Modell und dem Vorgehensmodell, zur Benutzerpartizipation durch eine überschaubare Menge orthogonaler Grundkonstrukte gelegt. Ausgehend vom entwickelten konzeptuellen Schema werden durch einen Algorithmus objektorientierte Strukturen abgeleitet.

2 Die E3R-Methode zur Modellierung des konzeptuellen Schemas

Ausgangspunkt der Entwicklung des *semantically enriched extended entity relationship models* (E3R) war der Wunsch, auf der Grundlage verbreiteter und anerkannter Grundkonzepte ein Modell zu schaffen, das wegen der überschaubaren Anzahl an Basiskonzepten schnell erlernt und eingesetzt werden kann.
Zusätzlich zum Modell existiert eine Vorgehensweise zur Gewinnung des konzeptuellen Schemas in E3R-Notation.

2.1 Historie und Qualitätsaspekte

Die Graphik der Abbildung 1 zeigt die historischen Vorläufer des E3R-Modells. Durchgezogene gerichtete Kanten verbinden Modelle mit ihren expliziten Vorgängern. Unterbrochene Linien versinnbildlichen Entwicklungseinflüsse. Die Knoten beinhalten neben den Namen des oder der Autoren, die abkürzende Bezeichnung sowie das Veröffentlichungsjahr. Knoten mit unterbrochenen Begrenzungslinien deuten Einflüsse an, die nicht explizit in Modellen manifestiert sind. Modifikationen ohne Namensänderungen werden durch ein nachgestelltes „+“ dargestellt. Im rechten Bildteil ist die Familie der Entity-Relationship-Abkömmlinge dargestellt, die sich am klassischen Entity-Relationship Modell (ERM) nach Chen [2] orientieren.
Die linke und mittlere Bildhälfte illustriert die Entwicklung Datenmodellen, die die Semantik in den Vordergrund rücken. Als gemeinsames Merkmal sei hier das Rollenkonzept angeführt.

Das E3R-Modell führt Entwicklungsrichtungen zusammen. Hohes Ziel dieser Entwicklung war es, vergleichsweise einfache Grundkonzepte mit großer Mächtigkeit zu bieten, die zur Beschreibung verschiedenster auch komplexer Sachverhalte eingesetzt werden können. Hierzu orientiert sich E3R stark an der natürlichen Sprache, was die Benutzerpartizipation in den frühen Analyse- und Modellierungsphasen signifikant erleichtert.

Als Konsequenz entstehen konzeptuelle Schemata die folgenden Qualitätsansprüchen genügen:

- *Darstellung der gesamten relevanten Information*, insbesondere Abbildung existierender semantischer Regeln
- *Sukzessive Verfeinerbarkeit* ausgehend von der dargestellten Informationsstruktur
- *Praktische Benutzbarkeit* sowie die *Lesbarkeit* für Fachexperten durch Orientierung der angebotenen Modellierungskonstrukte an der natürlichen Sprache
- *Einheitliche Realitätsabbildung* durch klare und eindeutige Definition der Semantik der angebotenen Modellierungsprimitive
- *Verständlichkeit und Genauigkeit* durch eine überschaubare Menge an Modellierungsprimitiven
- *Ableitung verschiedenster Implementierungssichten* durch systemunabhängige Formulierung und damit portable Spezifikation
- *Sichten*- und *Anwendungsneutralität* durch Modellierung des Problembereichs ohne spätere Implementierungssysteme zu berücksichtigen oder frühzeitige Optimierungen vorzunehmen[1]
- *Änderungslokalität* durch Darstellung der Information, so daß sich spätere Modifikationen nicht auf andere (nicht betroffene) Modellelemente auswirken

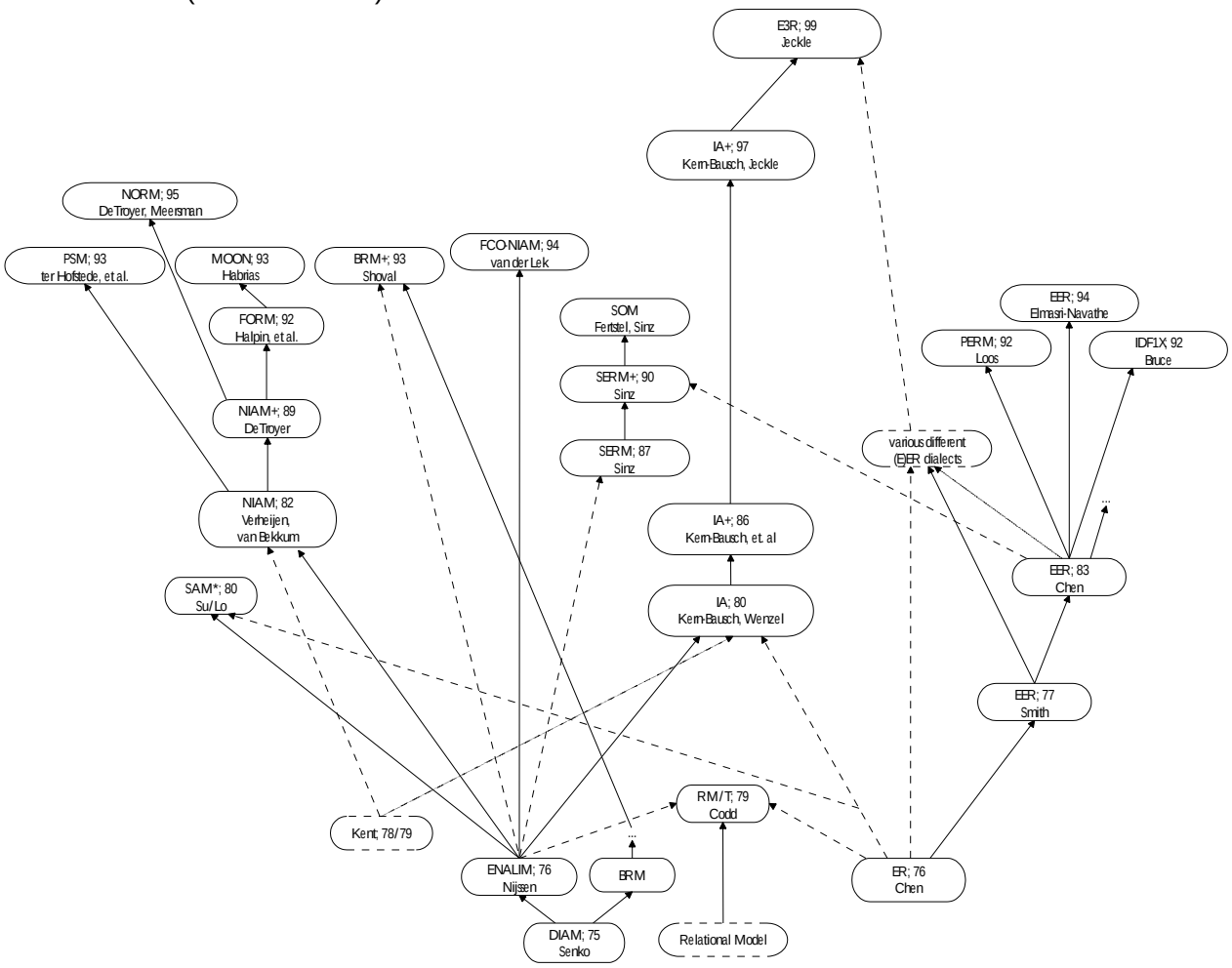

Abbildung 1: Vorläufer und Historie des E3R-Modells

2.2 Strukturen und graphische Darstellung

Generell lassen sich zwei Ansätze für konzeptuelle Modellierungssprachen unterscheiden. Die eine Klasse bildet *Objekte* der realen Welt, ihre *Eigenschaften* und *Beziehungen* zu anderen Objekten ab, die andere verzichtet zugunsten durchgängiger Modellierung aller Entitäten als Objekte, auf die Eigenschaften, und behandeln diese wie Objekte. Während ersterer bereits eine gewisse Vorstrukturierung des abzubildenden Realitätsauschnittes (*universe of discourse (UoD))* vornimmt, gibt der zweite der Gleichbehandlung aller Elemente des UoD den Vorzug.

Zum zweiten Ansatz ist das E3R-Modell zu zählen. Es kennt ausschließlich *Entitäten* bzw. *Entitätstypen* als Darstellungsmittel für Objekte. Sie bilden die Grundlage der abzubildenden Informationsstrukturen und entsprechen Klassen von Nomen der natürlichen Sprache, die im Sinne eines Wortfeldes (quasi) synonym gebraucht werden. In der graphischen E3R-Notation werden Entitätstypen als eineindeutig benannte Recht-

[1] Auch hier besitzt Knuths historischer Ausspruch *premature optimization is the root of all evil* seine Gültigkeit. Aus: Knuth, D. E.: *Structured Programming with go to Statements*, ACM Computing Surveys, Vol. 6, No. 4, December, 1974, p. 268.

ecke. Das Beispielschema der Abbildung 2 stellt das aus der Literatur (u.a. [3, pp. 145-148]) bekannte *Company*-Schema dar.

Die Beziehungen zwischen Entitätstypen werden durch Beziehungstypen (symbolisiert durch Rauten, die ebenfalls eineindeutig benannt sind) dargestellt. E3R unterstützt beliebige Beziehungstypen, mindestens vom Grade zwei. Damit sind neben binären auch höhergradige (n-äre) abbildbar. In Abbildung 2: der Beziehungstyp *Address*, der die Entitätstypen *State, Street* und *City* verbindet.
Grundlegender Beitrag zu einem Beziehungstypen sind die *Rollen*. Jeder beteiligte Entitätstyp trägt eine solche zur Beziehung bei. So nimmt der Entitätstyp *Employee* innerhalb der Beziehung *DeptEmp* die Rolle *works in* ein, während das *Department* die Rolle *employs* spielt. Wie anhand des Beispielschemas ersichtlich, kann ein Entitätstyp durchaus mehrere verschiedene Rollen gegenüber desselben anderen Entitätstypen einnehmen. Das Konzept der Rolle führt zu deutlich offensichtlicheren und realitätsnäheren Abbildungen der Informationsstruktur.
Jede Rolle wird durch ein *Mitgliedschaftsintervall* ergänzt, daß die Kardialität des rollentragenden Entitätstypen innerhalb der betrachteten Beziehung wiedergibt. So ist ein *Employee* in der Rolle *works in* genau einem (1:1) *Department* in der Rolle *employs* zugeordnet. Andererseits beschäftigt jedes *Department* mindestens einen, jedoch eine nach oben nicht begrenzte Anzahl („beliebig viele“), Mitarbeiter. Das offene Mitgliedschaftsintervall wird durch ein *n* als obere Grenze versinnbildlicht.
Besondere Bedeutung kommt der *Repäsentationsbeziehung* zwischen Entitätstypen zu. Dieser, als rollen- und kardinalitätslose gerichtete Kante dargestellte, Beziehungslose ordnet Entitätstypen eine physische Repräsentation – im Sinne eines Datentyps – zu. So werden im Beispiel den Entitästypen *PName, MiddleInitial* und *LName* Zeichenketten (*String)* als physische Repräsentation zugrunde gelegt. So können anwendungsbedingte Restriktionen hinsichtlich zugelassener Werte abgebildet werden.

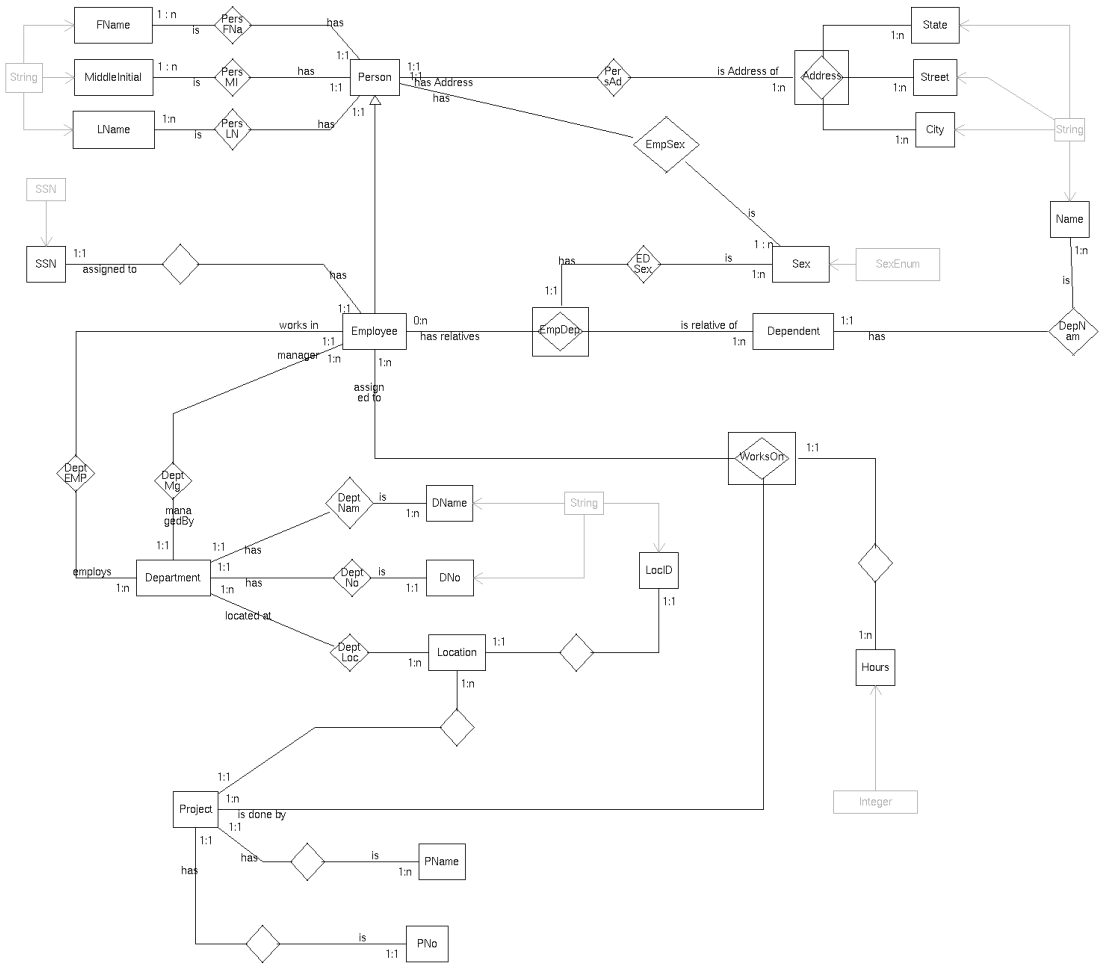

Abbildung 2: Beispielschema in graphischer E3R-Notation unter Markierung der Zusammenhangskomponenten

Um gegenüber Änderungen stabile Informationsstrukturen zu erhalten müssen Beziehungstypen *semantisch irreduzibel* formuliert sein. Dies bedeutet, jeder Beziehungstyp des Schemas deckt einen Informationsgehalt ab, der durch weitere Zerlegung verloren ginge. Sofern ein Beziehungstyp zerlegbare Zusammenhänge ausdrückt sind diese Informationen einzeln, d.h. durch mehrere Beziehungstypen, darzustellen.

Damit werden die ermittelten Zusammenhänge so unabhängig voneinander dargestellt, daß spätere Änderungen oder Erweiterungen lokal, genau an den betroffenen (d.h. zu ändernden) Entitäts- und Beziehungsstypen eingebracht werden können.

Ferner bietet E3R ein hybrides Beziehungskonstrukt an, das die Charakteristika von Beziehungs- und Entitätstyp vereinigt. Es erlaubt Beziehungstypen selbst Beziehungen einzugehen. Im Beispiel der Abbildung 2: *WorksOn,* der Verbindung zwischen *Employee* und *Project* wird die Anzahl der für ein Projekt durch einen Mitarbeiter aufgewendeten Stunden (Entitätstyp *hours*) zugeordnet.
Bereits durch dieses Beispiel ergibt sich intuitiv die Semantik dieses Informationskonstrukts. Es wird in allen denjenigen Fällen eingesetzt, die eine Zuordnung des Entitätstypen zu ausschließlich einem der an der Beziehung beteiligten Entitäten nicht die reale Semantik abdeckt. Beispiel: Die Zuordnung der erbrachten Stunden ist weder allein einem Mitarbeiter – dieser kann für mehrere Projekte unterschiedliche Zeit aufwenden – noch einem Projekt – für jedes Projekt können durch verschiedene Mitarbeiter verschiedene Zeiten verbucht werden – zuzuschreiben.

Als weitere, im Beispielschema nicht benutzte Komponenten, bietet das E3R-Modell Vererbungskonstrukte (dargestellt als gerichtete Kante vom *Employee* zu dessen Generalisierung *Person*) und Möglichkeiten zur Einbringung von Metainformation an.
Metainformation kann auf allen Ebenen des konzeptuellen Schemas formuliert werden. So können schemaglobale Einschränkungen ebenso formuliert werden, wie Konsistenzbedingungen, die sich ausschließlich auf einen Entitäts- oder Beziehungstypen beziehen. Neben der Formulierung in einer beliebigen natürlichen oder formalen Notation kann auch E3R selbst zur Darstellung dieser Einschränkungen benutzt werden.

2.3 XML-basierte textuelle Darstellung

Ebenso wie die vorgestellte graphische Notation definiert E3R auch eine textuelle Sprache gleicher Mächtigkeit. Diese Sprache wird automatisiert aus der UML-Notation des E3R-Metamodells gewonnen (siehe 3.3). Hierzu wird die durch das World Wide Web Consortium standardisierte Metasprache XML [4] eingesetzt.

Das nachfolgende Fragment zeigt einen Ausschnitt der XML-Darstellung des Schemas aus Abbildung 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<E3R xmlns="http://www.jeckle.de/e3r">
  <Model>
    <NamedElement.name>Company</NamedElement.name>
    <Model.defines>
      <Entity>
        <NamedElement.name>City</NamedElement.name>
        <Entity.isAllquantor xmi.value="false" />
        <Entity.isAbstract xmi.value="false" />
        <Entity.representedBy>
          <Representation>
            <NamedElement.name>String</NamedElement.name>
          </Representation>
        </Entity.representedBy>
      </Entity>
      <Relationship>
        <NamedElement.name>WorksOn</NamedElement.name>
        <Relationship.isObjectified>
          <Entity>
            <NamedElement.name>WorksOn</NamedElement.name>
          </Entity>
        </Relationship.isObjectified>
        <Relationship.consistsOf>
          <Role>
            <NamedElement.name>assigned to</NamedElement.name>
            <Role.minCardinality>1</Role.minCardinality>
            <Role.maxCardinality>n</Role.maxCardinality>
            <Role.isAdopted>
              <Entity>
                <NamedElement.name>Employee</NamedElement.name>
              </Entity>
            </Role.isAdopted>
          </Role>
```

3 Transformation der konzeptuellen in objektorientierte Strukturen

3.1 Qualitative Anforderungen

Standen eingangs qualitative Anforderungen an konzeptuelle Schemata insbesondere im Kontext der einfachen Anwendbarkeit der entstehenden Sprache für den Modellierer im Vordergrund, so betont der Qualitätsbegriff im objektorientierten Umfeld stärker auf die technische Umsetzung.
Information hiding [5] zur Realisierung des objektorientierten Konzepts der *Kapselung*. Damit sollen höher aggregierte Informationseinheiten – im objektorientierten Sprachgebrauch *Klassen* – geschaffen werden, die zusammenhängende Informationen umfassen, und somit die Wartbarkeit erleichtern.
Kopplung und *Kohäsion* im Sinne von Yourdon [6]. Während eine Klasse einen möglichst hohen Kohäsionswert aufweisen soll, der den *inneren Zusammenhalt*, der aggregierten Elemente ausdrückt, soll die durch den Kopplungsfaktor ausgedrückte Abhängigkeit zwischen den verschiedenen Klassen möglichst minimiert werden.
Der vorzustellende Generierungsalgorithmus sucht den Kapselungsgedanken, sowie minimale Kopplung bei maximaler innerer Kohäsion optimal zu verwirklichen. Dagegen bleiben andere Qualitätskriterien wie Metriken (vgl. [7]) unberücksichtigt, da sie statischen Auswertungen erst nach Abschluß des Modellierungsprozesses angewendet werden.

3.2 Bildung der Informationszusammenhänge

Zunächst werden zusammenhängige Informationseinheiten im konzeptuellen Schema ermittelt. Hierzu werden *Zusammenhangskomponenten* gebildet, dergestalt, daß alle Rollen mit Mitgliedschaftsintervall 1:1 markiert werden. Entlang dieser Beziehungen liegende Entitäts- und Beziehungstypen werden graphisch zu Zusammenhangskomponenten gruppiert. Außerhalb von Zusammenhangskomponenten liegende Konstrukte werden gesondert behandelt.
In einem folgenden Schritt werden die einzelnen in den Zusammenhangskomponenten enthaltenen Entitätstypen betrachtet. Durch Analyse der Beziehungstypen innerhalb einer Zusammenhangskomponte wird diese weiter aufgespalten, sofern sich darin mehrere Entitätstypen mit Identität (im objektorientierten Sinne) befinden.

3.3 Ableitung von statischen Klassenstrukturen

Aus den modifizierten Zusammenhangskomponenten werden Klassenstrukturen. Abbildung 3 zeigt (in UML-Notation [1]) das Ergebnis, wobei in einem weiteren Optimierungsschritt u.a. die in im vorhergehenden Schritt nicht in Zusammenhangskomponenten aufgenommenen Entitäts- und Beziehungstypen transformiert wurden. Der Vergleich zum ursprünglichen konzeptuellen Schema zeigt die Erhaltung aller Entitätstypen und Rollen auf. Mithin wurde keine relevante Information durch den Generierungsprozeß verloren.
Zunächst werden die Klassen selbst mit ihren Attributen aufgebaut, sowie die notwendigen Assoziationen gebildet. Zusätzlich sind die aus Konsistenzgründen notwendigen Einschränkungen formuliert. So beispielsweise der durch den Beziehungstyp *DeptNo* ausgedrückte Informationszusammenhang in das Attribut *DNo* der Klasse *Department* übersetzt, und zusätzlich ein in *Object Constraint Language* [1, pp. 7-1 – 7-50] formulierter Ausdruck annotiert, der die im konzeptuellen Schema dargestellte Eindeutigkeit formal spezifiziert.
Offensichtlich enthält das Klassendiagramm deutlich weniger Assoziationen als im konzeptuellen Schema als Beziehungstypen modelliert wurden. Dies liegt in der Realisierung der Kapselung begründet.

3.4 Applikationsorientierte Optimierung

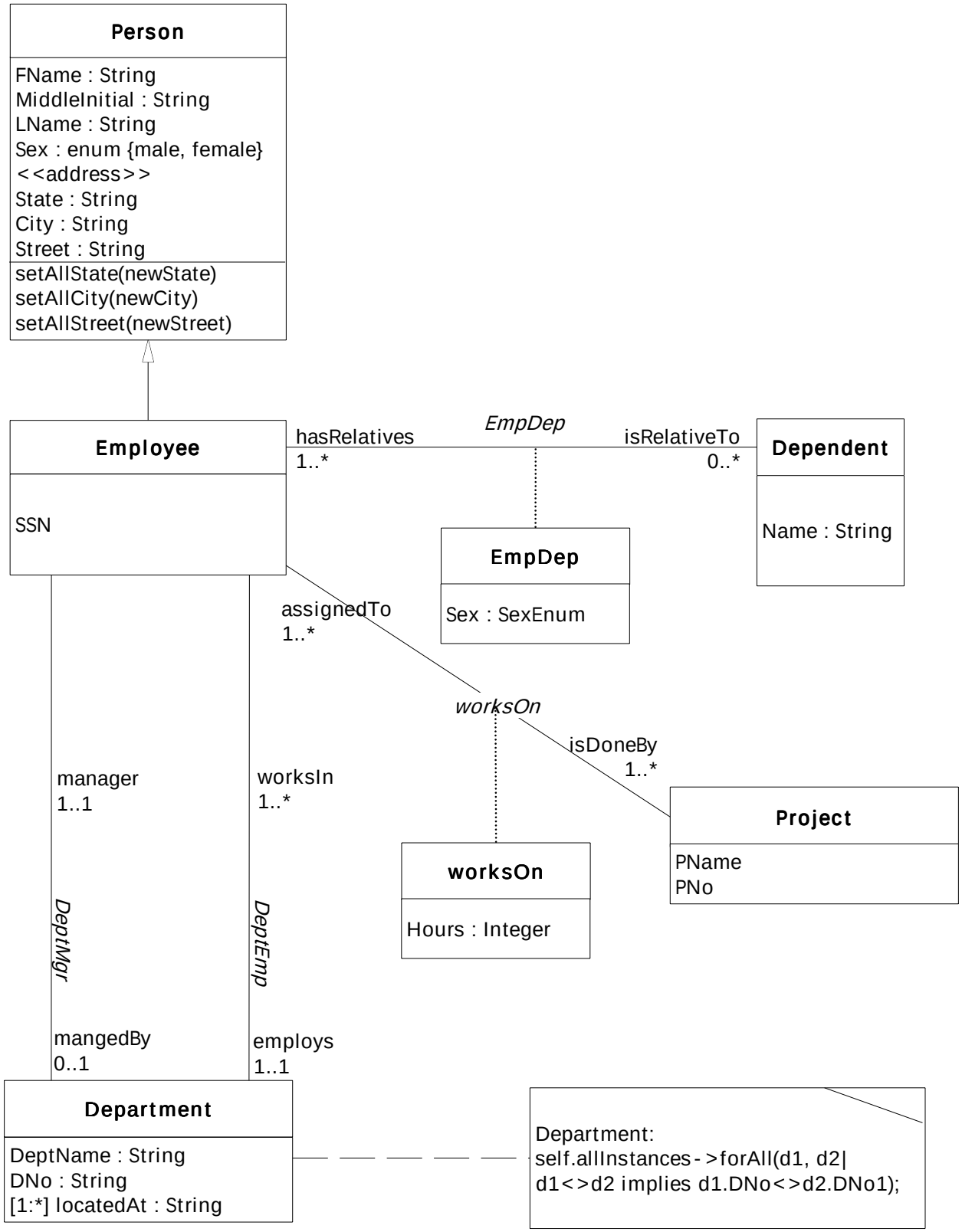

Abbildung 3: Transformierte und optimierte Klassenstrukturen

Nach Abschluß des Generierungsprozesses entsteht ein anwendungssprezifisches, jedoch sichtenneutrales, Klassenschema. Dies bedeutet, daß die gesamte im konzeptuellen Schema formulierte Semantik des Problembereichs abgebildet wurde, jedoch noch keine optimierenden Maßnahmen, durch Einbringung spezifischer Sichten, vorgenommen wurden.

Spezifische Sichten bedeuteten stets einen Informationsverlust, da alle im konkreten Anwendungsfalle nicht benötigten Sichten verloren gehen. Dennoch bietet das Generierungskonzept in einem abschließenden Schritt die Möglichkeit der kontrollierten Einbringung von Applikationssichten an.

Im Beispiel der Abbildung 3 wird die zunächst eigenständige Klasse *Address* zugunsten einer Applikationsanforderung durch *Employee* absorbiert.

3.5 Gewinnung dynamischer Strukturen

Zusätzlich zu den bisher skizzierten statischen Strukturen aus Klassen, deren Attributen und ihren Assoziationen lassen sich auch aus der im konzeptuellen Schema formulierten Semantik gewisse dynamische Aspekte generieren. Hierbei handelt es sich in erster Linie um konsistenzgarantierende Methoden denotationeller Semantik. Im betrachteten Beispiel: die Modifikationsmethoden der Attribute *State, Street* und *City* der Klasse *Person* (*setAll...*). Im Gegensatz zu den üblicherweise nicht explizit angegebenen *Accessor-* und *Mutator-Operationen* schreiben diese Werte in allen Person-Objekten, sofern diese übereinstimmen. Dies wird durch die dargestellte Semantik des konzeptuellen Schemas, konkret den Beziehungstypen *PersAdr*, gerechtfertigt. Der Entitätstyp *Address* verfügt über keine Repräsentation, somit kann ihre Identität nicht über eine explizite Wertgleichheit definiert werden. Die Rolle *is address of* mit Mitgliedschaftsintervall *1:n* (siehe Abbildung 2) ist so zu interpretieren, daß ein und dieselbe Adresse mehreren Personen zugeordnet werden kann. Daraus resultieren prinzipiell zwei Änderungsszenarien der Adresse, die sich an folgenden Anwendungsfällen illustrieren lassen: Zum einen die Adressänderung ausgehend vom Adressinhaber, der *Person,* beispielsweise durch Umzug, zum Anderen die Adressänderung ausgehend von der Adresse selbst, beispielsweise durch Umbenennung der Straße oder Stadt[2]. Da letzterer Fall nicht der üblichen Interpretation entspricht, bedarf er gesonderter Behandlung. Der Generierungsprozess ermittelt die notwendigen Operationen und fügt sie in das entstehende Klassenschema ein. Zusätzlich werden für die Umsetzung der Operationen in Methoden UML-Zustandsautomaten (in der Graphik nicht dargestellt) definiert und dem UML-Modell hinzugefügt.

4 Zusammenfassung

Die verwendete E3R-Notation unterstützt Modellierer und Fachexperten mit ihren einfachen und intuitiven Konstrukten die Umsetzung des Problembereichs in ein konzeptuelles Schema.
Der vorgestellte Generierungsalgorithmus transformiert das konzeptuelle Schema in objektorientierte Strukturen.
Es sei unbestritten, daß erfahrende objektorientierte Modellierer ein ähnliches Ergebnis auch pragmatisch erzielen können. Jedoch bleiben dabei zwei zentrale Aspekte außer acht. Zunächst die Benutzerpartizipation vom frühen Stadium der Analyse an bis zur Realisierung. Damit läßt sich das Akzeptanzverhalten der späteren Anwender deutlich steigern, da sie zu allen Phasen des Entwicklungsprozesses aktiv miteinbezogen sind. Durch die E3R-Notation und den an der natürlichen Sprache orientierten Modellierungsprozeß können die entstehenden Schemata gemeinsam mit den Fachmann modelliert und zu jedem Zeitpunkt validiert werden.
Des weiteren ermöglichen die mit dem Generierungsalgorithmus verwirklichten Qualitätskriterien eine beschleunigte Erzeugung von objektorientierten Schemata. Dies offenbart sich insbesondere bei der Einbettung des vorgestellten Algorithmus in einen iterativen inkrementellen Entwicklungsprozeß als Vorteil, da bei Änderungen oder Erweiterungen der Spezifikation automatisiert logisch richtige Implementierungsstrukturen gewonnen werden können.

Literatur

[1] OMG (Hrsg.): *OMG Unified Modeling Language*, Version 1.3, Framingham, MA, USA, 1999. verfügbar über www.jeckle.de

[2] Chen, P. P. S.: *The Entity-Relationship model - toward a unified view of data*, ACM Transactions on Database Systems, pp. 9-36, Vol. 1, No.1, March, 1976.

[3] Elmasri, R.; Navathe, S. B.: *Fundamentals of Database Systems,* 2nd Edition, Benjamin Cummings Publishing, 1994.

[4] World Wide Web Consortium (eds.): *Extensible Markup Language (XML) 1.0,* W3C-Recommendation 1998-02-10,

[5] Parnas, D. L.: *On the criteria to be used in decomposing systems into modules*, Communications of the ACM, Vol. 15, No. 12, December, 1972.

[6] Yourdan, E.; Constantine, L.: *Structured design: Fundamentals of a discipline of computer program and systems design*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1979.

[7] McCabe, T.: *Development tools – quality assurance*, in: Proceedings 9th international Conference EDP System and Software Quality Assurance, Washington D.C., USA, 1990.

[2] Hier sei nur die Umbenennung von St. Petersburg in Leningrad (1924) und wieder zurück (1991), oder Chemnitz/Karl-Marx-Stadt (1953, 1990), Saigon/Ho-Chi-Mingh-Stadt (1975), Yerba Buena/San Francisco (1847) angeführt.