Vervollständigung des Constraint-basierten Assoziationskonzeptes von UML 2.0*

Carsten Amelunxen, Andy Schürr
Fachgebiet Echtzeitsysteme
Technische Universität Darmstadt
D-64283 Darmstadt
{carsten.amelunxen, andy.schuerr}@es.tu-darmstadt.de

Abstract: Die kürzlich erschienene neue Version 2.0 der UML bietet im Vergleich zu früheren Versionen neuartige Möglichkeiten der Strukturmodellierung mittels Klassendiagrammen. Eine wesentliche Neuerung ist ein Satz mengentheoretischer Constraints zur Anwendung auf Assoziationsenden. In diesem Papier zeigen wir, welche Seiteneffekte diese Constraints haben und legen dar, dass das Konzept in seiner derzeitigen Form irreführend erläutert und nicht vollständig ist. Wir zeigen auf, welche Defizite diese Lücke aus Anwendersicht und für die Spezifikation der UML an sich hat und bieten eine Lösung zur Vervollständigung des Assoziationskonzeptes durch Schließung dieser Lücke.

1 Einleitung

Im Laufe der letzten Jahre hat sich die Unified Modeling Language (UML) [Obj05b] als weit verbreiteter Modellierungsstandard etabliert. Die UML ist eine umfassende Modellierungssprache, die verschiedene Teilsprachen für unterschiedliche Modellierungsansätze unter einem gemeinsamen Kontext vereint. Zur Spezifikation der einzelnen Teilsprachen wird eine kleine Menge strukturbildender Modellierungskonstrukte als Metamodellierungssprache MOF 2.0 [Obj03] extrahiert. Dieser strukturbildenden Teilsprache (Klassendiagramme) kommt also aus zweierlei Hinsicht eine wichtige Bedeutung zu. Zum einen stellt sie einen äußerst populären Aspekt der UML dar, zum anderen ist sie für die Spezifikation der UML von besonderer Bedeutung.

Im Zuge der Weiterentwicklung hin zu Version 2.0 flossen neue Modellierungskonstrukte zur Strukturmodellierung in die Sprachspezifikation ein. Die UML 2.0 Klassendiagramme verfügen u. a. über ein, gegenüber früheren Versionen, erweitertes Assoziationskonzept. Assoziationen, als tragender Bestandteil der Strukturmodellierung in UML 2.0, können durch eine Menge vordefinierter Constraints in definierter Beziehung zu anderen Assoziationen stehen. Dieses Konzept bietet viele Vorteile hinsichtlich der Ausdrucksstärke und

*Eine ausführliche Version mit adäquaten Erläuterungen samt formaler und visueller Unterstützung kann unter http://www.es.tu-darmstadt.de/download/publications/amelunxen/internal_amelunxen_01.pdf eingesehen werden

der Übersichtlichkeit von Diagrammen, da es Zusammenhänge, die alternativ nur durch die Object Constraint Language (OCL) [Obj05a] ausgedrückt werden könnten, elegant in die Diagrammsyntax einfließen lässt. Das Assoziationskonzept stellt einen zentralen Bestandteil der Strukturmodellierung in UML 2.0 dar und ist somit aufgrund der Architektur der UML-Spezifikation, sowie der hervorgehobenen Bedeutung der Klassendiagramme sowohl für das Verständnis von UML an sich als auch für die Anwendung der Klassendiagramme von essenzieller Bedeutung.

Wir werden im Folgenden zeigen, dass dieses Assoziationskonzept, trotz der vielen Vorteile, missverständlich ist und semantisch Implikationen hervorruft, die syntaktisch nicht repräsentiert werden. Damit setzt dieser Beitrag auf die in [ABS04] getroffenen Aussagen auf. Die dargestellten Beobachtungen sind relevant für die tägliche Modellierungspraxis, wie wir bei der Realisierung des Metamodellierungsrahmenwerkes MOFLON feststellen mussten.

Kapitel 2 führt ein Beispiel ein, an dem sich die Diskussionen der einzelnen Beziehungen in Kapitel 3 orientieren. Anschließend werden in Kapitel 3 alle Beziehungen einzeln diskutiert und die Implikationen der Beziehungen dargestellt. Die Diskussion wird durch OCL-Constraints formal unterstützt. Aus der Diskussion ergibt sich die Notwendigkeit einer Ergänzung des Constraint-basierten Assoziationskonzeptes. Kapitel 4 gibt einen Überblick über verwandte Arbeiten und ähnliche Ansätze in anderen Modellierungssprachen. Abschließend werden die gewonnenen Erkenntnisse in Kapitel 5 zusammengefasst.

2 Beispiel

Zur Demonstration des Assoziationskonzeptes in UML 2.0 wird im Folgenden ein vereinfachtes Metamodell eines Projektes in der Programmiersprache C [KR78] aus dem Bereich des Reengineerings vorgestellt. Als Grundlage dient ein simples Metamodell eines Dateisystems. Dieses Dateisystem wird für die Anforderungen eines C-Projektes hinsichtlich der Quellcodestrukturierung über unterschiedliche Dateitypen spezialisiert.

In dem Beispiel aus Abbildung 1 ist das vereinfachte Konzept eines Dateisystems durch das Paket *FileSystem* modelliert. Die Konzepte des Dateisystems werden durch die merge Beziehung[1] in das C-Projekt (*CProject*) übernommen. Die Beziehung zwischen einem Verzeichnis und einer Datei geht im Kontext eines C-Projektes über die bloße Enthaltensein-Beziehung hinaus, was durch eine zusätzliche Assoziation zwischen *CProject::Directory* und *CProject::File* wiedergegeben wird. Die durch das Assoziationsende *projectFile* repräsentierte Rolle, die eine Datei aus Sicht eines Projektverzeichnisses einnimmt, wird als abgeleitete Vereinigung aller Untermengen[2] definiert.

Verzeichnisse nehmen im Rahmen eines C-Projektes eine besondere Rolle ein, daher wird die Klasse *Directory* spezialisiert durch *SourceFolder* und *IncludeFolder*. Der Vorgabe, dass ein *IncludeFolder* nur *HeaderFiles* beinhalten darf, wird durch die zusätzliche Assoziation zwischen *IncludeFolder* und *HeaderFile* entsprochen. Diese spezielle Assozia-

[1] Hier nicht näher erläutert. Genaueres unter ([Obj04], S. 167 ff.).

[2] In diesem Beispiel aus Übersichtlichkeitsgründen beschränkt auf eine einzige.

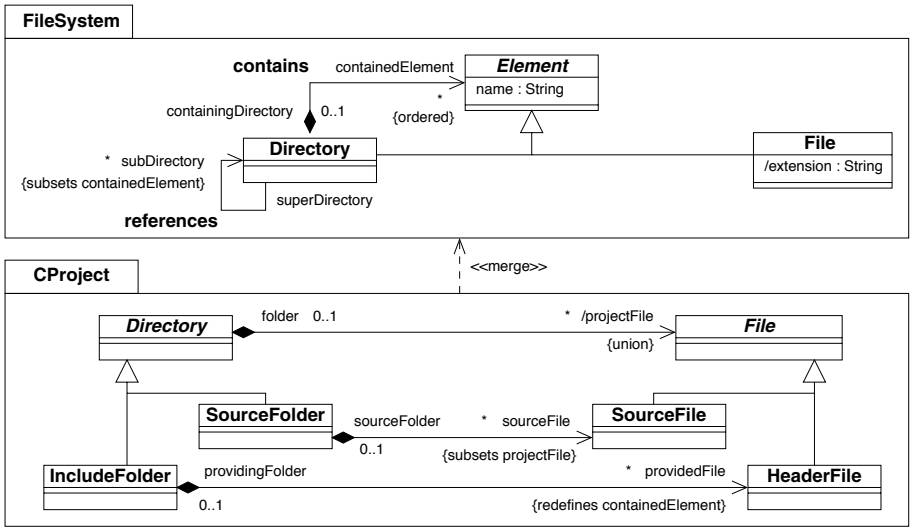

Figure 1: Vereinfachtes Metamodell für C-Projekte

tion soll die generelle Assoziation zwischen *Directory* und *Element* ersetzen bzw. im Fall der Verwendung mit einem *IncludeFolder* auf *HeaderFiles* begrenzen, was mit der Redefinition des Assoziationsendes *containedElement* durch *providedFile* ausgedrückt wird.

Analog existiert ein spezielleres Verhältnis zwischen einem *SourceFile* und einem *SourceFolder*. In einem *SourceFolder* können über die Rolle *containedElement* beliebige Elemente eines Dateisystems enthalten sein. Im Kontext eines C-Projektes nehmen jedoch nur *SourceFiles* die Rolle einer Projektdatei ein. Daher ist *sourceFile* eine Untermenge von *projectFile*.

Alle folgenden Erläuterungen der einzelnen Konzepte werden anhand des zuvor beschriebenen Beispiels vorgenommen. Damit sind die Aussagen zwar zu Gunsten einer leichteren Nachvollziehbarkeit gebunden an einen spezifischen Anwendungsfall, die All-ge-meingül-tig-keit sollte jedoch leicht übertragen werden können.

3 Assoziationen in UML 2.0

Das neue Assoziationskonzept von UML 2.0 ermöglicht die Untermengenbildung zwischen und Redefinition von Assoziationsenden. Allen Beziehungen zwischen Assoziationsenden gemein ist der Charakter einer Schreibabkürzung für ausführliche Constraints. Jede Beziehung könnte prinzipiell gleichwertig durch einen OCL-Constraint ausgedrückt werden, jedoch erhöht die Verwendung der vordefinierten Beziehungen auf Assoziationsenden die Lesbarkeit der Diagramme außerordentlich.

Assoziationsenden können prinzipiell als Mengen von Instanzen aufgefasst werden. Die Beziehungen zwischen Assoziationsenden haben daher einen stark mengentheoretischen Charakter. Bekannte mengentheoretische Zusammenhänge werden aufgegriffen, um populären Anwendungsszenarien von Constraints einfache Ersatzkonstrukte zuzuordnen. Die einzelnen Beziehungen und ihre Implikationen werden in den folgenden Kapiteln ausführlich diskutiert und dabei der mengentheoretische Charakter herausgestellt.

3.1 Assoziationsmodell

Um die Auswirkungen der mengentheoretischen Konstrukte auf Assoziationsenden erfassen zu können, ist die Struktur des Assoziationsmodells auf der nächst niedrigeren Metaebene von entscheidender Bedeutung. In der Literatur werden verschiedene Assoziationsmodelle diskutiert [Ö97], [RG98]. UML definiert zwar nicht ein detailliertes Instanzmodell, wie dies z. B. bei MOF 2.0 der Fall ist ([Obj03], S. 61), gibt jedoch eindeutige Hinweise, wie Assoziationsinstanzen, im Folgenden als Link bezeichnet, zu interpretieren sind: „*A link is a tuple with one value for each end of the association, where each value is an instance of the type of the end*“([Obj05b], S. 116).

Die folgenden Betrachtungen beschränken sich auf den leicht zu demonstrierenden Fall einer binären Assoziation, können aber auf den allgemeinen Fall einer n-ären Assoziation übertragen werden. Ein Link verbindet jeweils eine Instanz der beteiligten Typen mit den Instanzen der übrigen beteiligten Typen. Das von UML vorgesehene Assoziationsmodell ist also ein Modell basierend auf Relationen. Bezogen auf das Beispiel aus Abbildung 1 stellt sich das Assoziationsmodell exemplarisch in Gleichungen 1 dar.

context Directory

$$\text{inv} : containedElement \rightarrow \text{forAll}(e|e.containingDirectory \rightarrow \text{includes}(\text{self})) \quad (1)$$

Es existiert also für jedes Element in der Menge eines Assoziationsendes jeweils ein Gegenstück in den Mengen der übrigen Assoziationsenden. Das Tupel dieser Elemente ist ein Link. Die Mengen der Assoziationsenden können nicht unabhängig von der Menge aller Assoziationsinstanzen, der Linkmenge, betrachtet werden. Über die Linkmenge ergeben sich Abhängigkeiten zwischen den Mengen aller Assoziationsenden.

3.2 Untermengenbildung

Die einfachste Beziehung zwischen Assoziationsenden ist die Untermengenbeziehung, welche durch den Constraint subsets angegeben wird. Bezogen auf das Beispiel aus Abbildung 1 bedeutet dies, dass alle Instanzen, die in der Menge *sourceFile* geführt werden, durch den subsets-Constraint auch in der Menge *projectFile* zu führen sind.

Eine Untermengenbeziehung auf einem Ende einer Assoziation impliziert Untermengenbeziehungen auf allen anderen Enden einer Assoziation[3]. Konkret bedeutet dies, dass ein subsets-Constraint auf einem Assoziationsende subsets-Constraints auf allen Assoziationsenden impliziert. Eine Untermengenbeziehung bezieht sich semantisch also auf die gesamte Assoziation, da die Mengen der Assoziationsenden nicht unabhängig voneinander betrachtet werden können. Es ergeben sich im Falle einer binären Assoziation drei (im allgemeinen n-ären Fall $2^n - 1$) Fälle von Untermengenbeziehungen, die alle dieselbe Situation beschreiben.

3.3 Vereinigte Untermengen

Zur Ergänzung der Untermengenbeziehung sieht UML 2.0 die Möglichkeit vor, Assoziationsenden, welche Ziel einer Untermengenbeziehung sind und somit eine Obermenge darstellen, als Vereinigung aller Untermengen zu kennzeichnen. Eine vereinigte Obermenge besteht ausschließlich aus den Elementen ihrer Untermengen (siehe Gleichung 2). Durch die Vereinigung wird eine direkte Instanzierung ausgeschlossen, die Menge ist abgeleitet. Da die Vereinigung von Untermengen nur eine Erweiterung der Untermengenbeziehung ist, ist auch die Vereinigung von Untermengen implikationsbehaftet.

context SourceFolder
inv : $(projectFile \rightarrow$ includesAll$(sourceFile))$ and
$((projectFile - sourceFile) \rightarrow$ isEmpty$())$ (2)

Ein union-Constraint impliziert also die Vereinigung der Untermengen auf allen Enden der Assoziation[4]. Steht an einem Ende einer Assoziation ein union-Constraint, impliziert dies union-Constraints auf allen Enden der Assoziation. Analog zur Untermengenbildung bezieht sich die Vereinigung von Untermengen auf die gesamte Assoziation. Es existieren somit auch analog zur Untermengenbeziehung, mehrfache Notationsmöglichkeiten zur Beschreibung derselben Semantik. Eine Assoziation mit mindestens einem Assoziationsende vereinigter Untermengen entspricht einer abstrakten Assoziation und wird üblicherweise zwischen abstrakten Klassen angewandt, ist aber nicht auf diese beschränkt.

3.4 Redefinition

Assoziationsenden können in UML 2.0 redefiniert werden. Die (kovariante) Redefinition wird verwendet, um redefinierbare Features, u. a. Assoziationsenden, in einer Vererbungshierarchie neu zu definieren. Ein redefinierendes Feature tritt an die Stelle des redefinierten und übernimmt dessen semantische Funktion, da redefinierte Features

[3] Auf die vollständige Herleitung muss an dieser Stelle aus Platzgründen verzichtet werden. Die Herleitung kann der zuvor erwähnten vollständigen Version entnommen werden.

[4] Die vollständige Herleitung der Implikation kann der ausführlichen Version entnommen werden

nicht vererbt werden. Aus mengentheoretischer Sicht lässt sich dieser Zusammenhang wie folgt formulieren: „*...In this case, given a set of specific instances for the other ends of both associations, the collections denoted by the redefining and redefined ends are the same.*“([Obj04], S. 116).

Durch die Forderung, dass redefinierendes und redefiniertes Assoziationsende dieselbe Menge beschreiben, folgt aufgrund der Typbeziehung zwischen *HeaderFile* und *Element*, dass die redefinierte Menge *containedElement* bezogen auf eine Instanz der Klasse *IncludeFolder* ausschließlich Instanzen der Klasse *HeaderFile* enthalten kann[5] (siehe Gleichung 3 und 4). Diese Beziehung hat zwar auch Implikationen, wie wir im Folgenden zeigen werden, jedoch wird keine gleichartige Beziehung impliziert, wie dies bei einer Untermengenbeziehung der Fall ist. Eine Redefinition auf einem Assoziationsende impliziert keine weiteren Redefinitionen auf den anderen Assoziationsenden, schließt diese aber auch nicht aus.

context IncludeFolder

$$\text{inv} : containedElement = providedFile \tag{3}$$

$$\text{inv} : containedElement \rightarrow \text{select}(f|\ \text{not}\ f.\text{oclIsKindOf}(HeaderFile)) \rightarrow \text{isEmpty}() \tag{4}$$

Das Beispiel aus Abbildung 1 bestätigt diese Feststellung. Die Redefinition des Assoziationsendes *containedElement* durch *providedFile* soll ausdrücken, dass ein *IncludeFolder* ausschließlich *HeaderFiles* enthalten kann. Durch die Redefinition wird nicht ausgeschlossen, dass eine Header-Datei nicht auch in einem beliebigen Verzeichnis liegen kann, also eine Instanz von *HeaderFile* nicht auch mit einer Instanz vom Typ *Directory* über *containingDirectory* verlinkt werden kann. Es existieren zwar $2^n - 1$ Mög-lichkei-ten auf einer n-ären Assoziation Redefinitionen anzugeben, jedoch deckt jede dieser Möglichkeiten semantisch einen separaten Fall ab. Die Redefinition ist deshalb tatsächlich semantisch dem Assoziationsende zuzuschreiben und nicht der Assoziation als Ganzes.

Aus der Gleichsetzung von *containedElement* und *providedFile* folgt mindestens, dass *providedFile* in *containedElement* enthalten ist und somit impliziert eine Redefinition eine Untermengenbeziehung auf allen weiteren Enden der redefinierenden Assoziation. Die Implikation einer Untermengenbeziehung ist zweifelsfrei richtig, jedoch gilt tatsächlich ein strengerer Zusammenhang als eine einfache Untermengenbeziehung. Aufgrund der Gleichsetzung von *containedElement* und *providedFile* ist die Menge der Links zwischen einer Instanz von *IncludeFolder* und einer Instanz von *HeaderFile* in beiden Assoziationen gleich.

Als Implikation der Redefinition, wie sie in Gleichung 5 dargestellt ist, fällt auf, dass die Mengen *providingFolder* und *containingDirectory* gleich sind bezogen auf Instanzen der Klasse *HeaderFile*, jedoch nicht absolut identisch im Sinne einer Redefinition. Die Implikationen einer Redefinition können nicht vollständig mit den zur Verfügung stehenden Möglichkeiten erfasst werden. Wir werden daher im Folgenden zeigen, dass zur Ver-

[5] Die Tatsache, dass die Oberklasse *Element* abstrakt ist, ist dabei bedeutungslos

vollständigung des Assoziationskonzeptes eine weitere Mengenbeziehung erforderlich ist.

context HeaderFile
inv : $(providingFolder \rightarrow$ includesAll$(containingDirectory \rightarrow$
select$(d|d.$oclIsKindOf$(IncludeFolder))))$ and
$(containingDirectory \rightarrow$ includesAll$(providingFolder))$ (5)

3.5 Gleichsetzung

Wie zuvor gezeigt wurde, geht bereits ohne die Berücksichtigung praktischer Erwägungen die Notwendigkeit einer weiteren Beziehung direkt aus den bisherigen Constraints hervor. Die Implikation einer Redefinition auf die anderen Enden einer Assoziation ist weder durch einen subsets-, noch durch einen union- und auch nicht durch einen redefines-Constraint zu erfassen. Wir plädieren daher den Zusammenhang, wie er zuvor in Gleichung 5 dargestellt wurde, als Gleichheit von Assoziationsenden durch den Constraint equals aufzunehmen. Die Einführung eines equals-Constraints vervollständigt das Konzept der mengentheoretischen Verknüpfung von Assoziationsenden und Attributen nicht nur auf theoretische Art und Weise, sondern bietet auch konkrete Vorteile.

Es muss demnach eine Möglichkeit geben, um auszudrücken, dass Links der *contains*-Assoziation zwischen zwei *Directory*-Instanzen auch in der *references*-Assoziation geführt werden, ohne *contains* dabei einzuschränken. Exakt diese Semantik erfüllt die von uns vorgeschlagene equals-Beziehung. Erst mit der equals-Beziehung ist man in der Lage, das Dateisystem im Sinne von UML 2.0 und ohne den Einsatz expliziter OCL-Constraints korrekt zu beschreiben. Die equals-Beziehung erfüllt aus mengentheoretischer Sicht die Funktion einer Selektion und fügt sich somit nahtlos und elegant in die bisherige Sammlung mengentheoretischer Constraints ein. Der grundsätzliche Unterschied zur Redefinition besteht darin, dass die generellere Assoziation in ihrer Funktion nicht eingeschränkt wird. Semantisch erfüllt sie eine leichtere Form der Redefinition, ist aber strenger als eine Untermengenbeziehung und bildet somit eine nützliche Brücke in Situationen, in denen eine Redefinition zu streng ist, eine Untermengenbeziehung aber nicht ausreicht. Das Beispiel aus Abbildung 1 wird also erst semantisch korrekt, wenn der Constraint {*subsets containedElement*} des Assoziationsendes *subDirectory* ersetzt wird durch {*equals containedElement*}.

context Directory
inv : $(superDirectory \rightarrow$ includesAll$(containingDirectory \rightarrow$
select$(f|f.$oclIsKindOf$(Directory))))$ and
$(containingDirectory \rightarrow$ includesAll$(superDirectory))$ (6)

Aufgrund des Assoziationsmodells treten auch bei der Gleichsetzung zweier Assoziationsenden Implikationen für alle anderen beteiligen Assoziationsenden auf. Da es sich bei der Gleichsetzung um eine Kombination von Untermengenbeziehungen handelt ergibt sich die

Implikation aus Gleichung 6 leicht aus dem Prinzip der Herleitung einer Untermengenimplikation. Die folgende Auflistung fasst alle für ein vollständiges und konsistentes Assoziationskonzept erforderlichen Constraints zusammen, indem zu jedem Constraint die Semantik bezüglich des Assoziationsendes und die Semantik für die komplette Assoziation aufgelistet werden. Als Implikation sind die Constraints angegeben, die für die übrigen Assoziationsenden gelten, falls der jeweilige Constraint angewendet wird.

- subsets: Untermenge eines Assoziationsendes; impliziert subsets auf allen Enden. Alle Links der Untermengenassoziation sind auch Links der Obermengenassoziation.
- union: Vereinigung von Assoziationsenden; impliziert union auf allen Enden. Die Obermengenassoziation kann nicht direkt instanziert werden. Die Menge der Links ergibt sich aus allen Untermengenassoziationen.
- equals: Gleichheit von Assoziationsenden; impliziert equals. Zusätzlich zur geltenden Untermengensemantik sind alle typkompatiblen Links, der vom Constraint betroffenen Assoziation (Obermengenassozation) auch Links der Constraint-tragenden Assoziation (Untermengenassoziation).
- redefines: Redefinition von Assoziationsenden; impliziert equals. Pro Redefinition wird eine Instanzierungsvariante der redefinierten Assoziation ausgeschlossen, ansonsten gilt die Semantik einer Gleichheit.

4 Verwandte Arbeiten

Das Konzept der Assoziation in Modellierungssprachen geht zurück in die Anfänge der objekt orientierten Programmierung und der damit verbunden Modellierung [Rum87]. Seit den Anfängen wurden immer wieder Versuche unternommen, das Konzept der Assoziation genauer zu erfassen [Nob97], [Ö97] und gegen das Konzept der Attribute abzugrenzen [Rum96b]. Die praktische Relevanz des Assoziationskonzeptes lässt sich u. a. auch an der Vielzahl von Ansätzen und Varianten zur Abbildung von Assoziationen auf Programmiersprachen ablesen [Rum96a], [GdCL03], [ABS04]. Neben dem klassischen Assoziationsansatz, der letztendlich Einzug in die UML hielt, existieren viele Ansätze, die Assoziationen in einem komplexeren, allerdings nicht in einem der UML 2.0 vergleichbaren Zusammenhang, beschreiben [KO98], [HS98]. Das Problem der Redefinition von Assoziationen wurde bereits auch für andere Modellierungssprachen beschrieben [Fra98]. Für das Assoziationskonzept früherer UML Versionen existiert mit der Arbeit von P. Stevens [Ste01] ein Ansatz zur Beschreibung der Semantik, der auch in großen Teilen von UML 2.0 noch Gültigkeit besitzt. Dieser Ansatz geht aber nicht auf die Problematik der Constraint basierten Beziehungen zwischen Assoziationsenden ein und kann somit auch nicht zur Lösung der dargestellten Problematik in UML 2.0 herangezogen werden.

5 Zusammenfassung

Die vorausgegangenen Untersuchungen haben gezeigt, dass die Mengen der Assoziationsenden nicht als unabhängige Mengen betrachtet werden können. Die Abhängigkeit von der Linkmenge ist immer gegeben. Daraus resultiert, dass das Constraint-basierte Assoziationskonzept von UML 2.0 sich semantisch nicht ausschließlich auf die Enden einer Assoziation bezieht, sondern auf die komplette Assoziation. Folgende Beobachtungen lassen sich zusammenfassen: subsets ist semantisch gesehen eine Eigenschaft der Assoziation als Ganzes und nicht ihrer Enden, union an einem (oder mehreren) Enden ist semantisch gleichwertig mit der Tatsache, dass die Assoziation abstrakt ist, redefines auf einem Assoziationsende impliziert eine neue, in der täglichen Modellierungspraxis fehlenden (equals) Beziehung auf den anderen Assoziationsenden.

Die Anwendung der Constraints auf Assoziationsenden kann aber aus zweierlei Gründen nicht in Frage gestellt werden. Zum einen hat die Redefinition zwar Auswirkungen auf die komplette Asszіation, es ist aber trotzdem semantisch relevant, welches Assoziationsende eine Redefinition ist bzw. redefiniert wird. Daher kann die Redefinition nicht verlustfrei durch eine die komplette Assoziation betreffende Beziehung ersetzt werden. Zum anderen ist es syntaktisch nicht zu vertreten, die Beziehungen der Assoziation zuzuordnen, da entweder die Qualität der Diagramme durch zusätzliche Linien leiden würde oder auf konsequente Assoziationsbenennung zurückgegriffen werden müsste, was sich in Namensräumen mit eindeutigen Namen, zusätzlich zu der ohnehin schwierigen Aufgabe der Assoziationsbenennung, schwierig umsetzen ließe. Assoziationsenden in der Funktion einer Rolle hingegen lassen sich in der Regel leicht benennen.

Die Kurzschreibweise für Constraints am Assoziationsende hat sich bewährt, da in den trivialen Fällen ein kompletter Constraint wenig intuitiv und zu unübersichtlich wäre. Sowohl im praktischen Alltag, als auch in der Spezifikation der UML selbst, finden sich Stellen an denen jede der Beziehungen ihren Zweck sinngemäß und unersetzlich erfüllt. Die Beziehungen sind gewinnbringend einsetzbar, wenn man sich der tatsächlichen Semantik bewusst ist, wie wir sie zuvor dargelegt haben.

Es sei an dieser Stelle nochmal ausdrücklich erwähnt, dass sich die Notwendigkeit der equals-Beziehung bereits aus dem bisherigen Assoziationskonzept ergibt. Die Einführung der zusätzlichen Beziehung ist also nicht bloß der Versuch, einem Sachverhalt, der auch über einen herkömmlichen OCL-Constraint erfasst werden könnte, eine abkürzende Schreibweise zuzuordnen. Vielmehr wird die Beziehung gebraucht, um das bisherige Assoziationskonzept in sich konsistent und mit all seinen Implikation erfassen zu können.

Die dargestellten Erkenntnisse sind Teil unserer Arbeit an dem Metamodellierungsrahmenwerk MOFLON. In Zukunft werden wir eine vollständige formalisierte statische und dynamische Semantik von MOF 2.0 vorlegen, für die die dargestellten Beobachtungen relevant sind und in Zusammenhang mit der übrigen Schnittmenge von MOF 2.0 und UML 2.0 gebraucht werden.

References

[ABS04] C. Amelunxen, L. Bichler, and A. Schürr. Codegenerierung für Assoziationen in MOF 2.0. In *Proceedings of the Modellierung 2004*, März 2004.

[Fra98] Ulrich Frank. The MEMO Object Modelling Language (MEMO-OML). *Arbeitsberichte des Instituts für Wirtschaftsinformatik. Universität Koblenz*, 10, Juni 1998.

[GdCL03] G. Genova, C.R. del Castillo, and J. Llorens. Mapping UML Associations into Java Code. *Journal of Object Technology*, 2(5):135–162, September/Oktober 2003.

[HS98] B. Henderson-Sellars. Clarifying Specialized Forms of Association in UML and OML. *Journal of Object-Oriented Programming*, 11(2):47–54, Februar 1998.

[KO98] K. Østerbye and J. Olsson. Scattered Associations in Object-Oriented Modeling. *Nordic Workshop on Programming Environment Research*, Juni 1998.

[KR78] B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[Nob97] J. Noble. Basic Relationship Patterns. In *Proceedings of the European Conference on Pattern Languages of Program Design (EuroPLOP'97)*, Irsee, Deutschland, 1997.

[Ö97] G. Övergaard. A Formal Approach to Relationships in The Unified Modeling Language. In Manfred Broy, Derek Coleman, Tom S. E. Maibaum, and Bernhard Rumpe, editors, *Proceedings PSMT'98 Workshop on Precise Semantics for Modeling Techniques*. Technische Universitaet Muenchen, TUM-I9803, April 1997.

[Obj03] Object Management Group. *Meta Object Facility (MOF) 2.0 Core Specification*, Oktober 2003. ptc/04-10-05.

[Obj04] Object Management Group. *UML 2.0 Infrastructure Specification*, November 2004. ptc/04-10-14.

[Obj05a] Object Management Group. *OCL 2.0 Specification*, Juni 2005. ptc/2005-06-06.

[Obj05b] Object Management Group. *UML 2.0 Superstructure Specification*, August 2005. formal-05-07-04.

[RG98] M. Richters and M. Gogolla. On Formalizing the UML Object Constraint Language OCL. In T.-W. Ling, S. Ram, and M.L. Lee, editors, *Proc. 17th International Conference on Conceptual Modeling (ER)*, volume 1507, pages 449–464. Springer-Verlag, 1998.

[Rum87] J.E. Rumbaugh. Relations as Semantic Constructs in an Object-Oriented Language. In *Proceedings of the ACM Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA'87)*, pages 466–481, Orlando, Florida, USA, Oktober 1987. ACM Press.

[Rum96a] J. Rumbaugh. Models for Design: Generating Code for Associations. *Journal of Object-Oriented Programming*, 8(9):13–17, Februar 1996.

[Rum96b] J.E. Rumbaugh. A Search for Values: Attributes and Associations. *Journal of Object-Oriented Programming (JOOP)*, 9(3):6–8, 49, 1996.

[Ste01] P. Stevens. On Associations in the Unified Modeling Language. In M. Gogolla and C. Kobryn, editors, *UML 2001 – The Unified Modeling Language. Modeling Languages, Concepts, and Tools: 4th International Conference*, pages 361–375, Toronto, Canada, Oktober 2001. Springer Verlag.