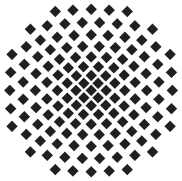

# Universität Stuttgart

Sonderforschungsbereich 627

Umgebungsmodelle für
mobile kontextbezogene Systeme

SFB 627 Bericht Nr. 2005/01

# Das Nexus Relationen- und Topologiekonzept

Datum: 24. November 2004

Autor(en): Bruno Arbter, Tobias Drosdol,
Frank Dürr, Matthias Großmann,
Nicola Hönle, Steffen Volz

CR Klassifikation: E.2, H.2.1, H.2.8

**Center of Excellence 627**
Spatial World Models for
Mobile Context-Aware Applications



**Sprecher des SFB:**
Prof. Dr. Kurt Rothermel
Institut für Parallele und Verteilte Systeme
Universitätsstraße 38
70569 Stuttgart
Deutschland

NEXUS

www.nexus.uni-stuttgart.de

# Inhaltsverzeichnis

# 1 Einleitung

Im Rahmen der Taskforce Topologie sollten Konzepte zur Repräsentation der Topologie raumbezogener Objekte innerhalb des AWM entwickelt werden. Das Arbeitsgebiet wurde aber bei Formierung der Taskforce auf die Entwicklung eines generellen Ansatzes zur Modellierung von Relationen in Nexus ausgedehnt, da die Abspeicherung von topologischen Beziehungen auf dem Fundament eines Gesamtkonzeptes für Relationen aufgebaut werden muss. Im Folgenden wird dieses Relationen-Konzept detailliert erläutert. Anschließend wird dann das Nexus-Topologie-Konzept vorgestellt.

# 2 Das Relationenkonzept von Nexus

Ein Schwerpunkt der Diskussionen innerhalb der Taskforce Topologie lag darin, zu entscheiden, welche Art von Relationen innerhalb von Nexus verwendet werden sollen. Grundsätzlich bestand die Möglichkeit, entweder leichtgewichtige (*lightweight*) oder schwergewichtige (*heavyweight*) Relationen einzuführen. Darüber hinaus musste festgelegt werden, wie Relationen aufgebaut werden bzw. welche innere Struktur sie aufweisen müssen. Eine weitere Aufgabe der Taskforce Topologie bestand darin, eine Lösung für die Frage zu finden, wie Relationen in Nexus zu verwalten sind. Schließlich war zu entscheiden, an welcher Stelle Relationen innerhalb der Objekthierarchie des AWM stehen sollten. Den hier genannten Fragestellungen ist im Folgenden jeweils ein Kapitel gewidmet.

## *2.1 Heavy- oder Lightweight Relationen?*

Gemäß der abstrakten Spezifikation des OpenGIS Konsortiums zum Thema Relationen [OGC 1999] wurden auch in der Diskussion um Nexus-Relationen die Begriffe 'lightweight' und 'heavyweight' verwendet.

Nach der Definition des OGC haben diese beiden Arten von Relationen die folgenden Eigenschaften:

**Lightweight Relationen:**

- Sind Teil eines referenzierten Objekts
- Haben keine Attribute
- Sind zweistellig

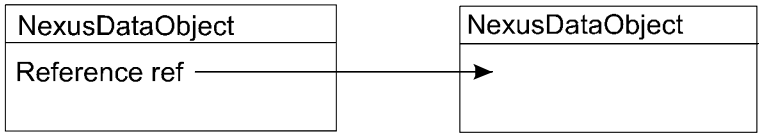


**Abb. 1: Darstellung einer Lightweight-Relation.**

**Heavyweight Relationen:**

- Existieren als eigenständiges Objekt
- Können Attribute haben
- Sind n ≥ 2 stellig

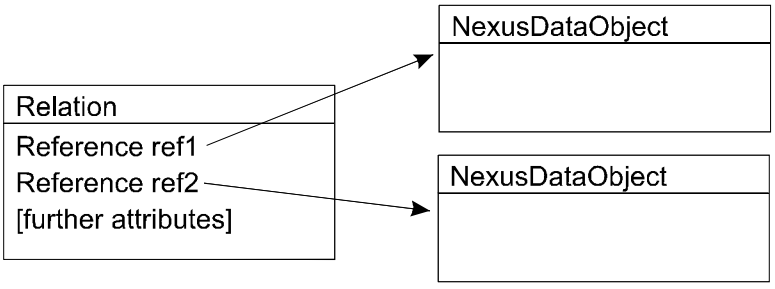


**Abb. 2: Darstellung einer Heavyweight-Relation.**

Heavyweight Relationen können flexibler eingesetzt werden. Eine heavyweight Relation kann jederzeit zwischen beliebigen Objekten definiert werden, während lightweight Relationen im Typ des einbettenden Objekts berücksichtigt werden müssen, d.h. bereits bei der Definition des Typs muss festgelegt werden, welche lightweight Relationen Objekte des Typs einbetten können.

Bei heavyweight Relationen kann zusätzlich eine Restriktion stattfinden, indem festgelegt wird, welche Objektklassen an einer Relation teilnehmen dürfen (um beispielsweise auszuschließen, dass andere Objekte als Städte an einer Partnerstadt-Relation teilnehmen können).

Für höherstellige Relationen (n > 2) sind heavyweight Relationen zwingend erforderlich. Daher müssen diese auf jeden Fall in Nexus unterstützt werden. Die Frage bestand nun darin, ob sowohl heavyweight als auch lightweight Relationen in Nexus verwendet werden sollen oder ob wir uns für einen Typ entscheiden sollten. Um eine durchgängige und klare Modellierung zu gewährleisten, hat sich die Taskforce darauf geeinigt, lightweight Relationen ***zunächst*** nicht aufzunehmen. Sollte später jedoch eine Optimierung der Verarbeitung von Relationen nötig sein, kann man sich diese Frage noch einmal stellen. Schon vorhandene lightweight Relationen sollen aus Gründen der Rückwärtskompatibilität allerdings weiterhin in der bestehenden Form – zumindest mittelfristig - unterstützt werden.

## *2.2 Struktur von Relationen*

Grundsätzlich können verschiedene strukturelle Eigenschaften eines Relationentyps unterschieden werden:

- Stelligkeit
- (Nicht-)Symmetrie
- Typisierung der referenzierten Objekte

Diese Eigenschaften und ihre Umsetzung im AWM werden im Folgenden erläutert.

### 2.2.1 Stelligkeit

Die Stelligkeit einer Relation wird durch die Zahl der referenzierten Objekte bestimmt. Die folgende Abbildung zeigt ein einfaches Beispiel einer zweistelligen Relation "TwinCity" (Partnerstadt). Zweistellige Relationen können auch als binär bezeichnet werden.


**Abb. 3: Darstellung einer zweistelligen Relation.**

### 2.2.2 Symmetrie

Eine binäre Relation R wird als symmetrisch bezeichnet, wenn gilt (a,b) ∈ R ⇔ (b,a) ∈ R. Ein Beispiel für eine solche binäre symmetrische Relation ist die oben dargestellte Relation "TwinCity". Das heißt, es gilt z.B. TwinCity(Stuttgart, Brünn) ebenso wie TwinCity(Brünn, Stuttgart). Im Gegensatz dazu haben die einzelnen referenzierten Objekte bei nicht-symmetrischen Relationen verschiedene Rollen. Ein Beispiel hierfür ist die topologische Relation "Contains", da hier unterschieden werden muss, welches Objekt ein anderes enthält.

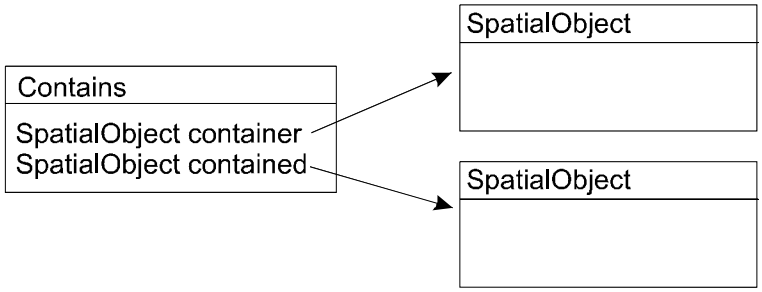


**Abb. 4: Darstellung einer nicht-symmetrischen Relation.**

Dabei muss zusätzlich beachtet werden, dass zweistellige Relationen nur einen sehr einfachen Fall der Symmetrie darstellen. Eine symmetrische (bzw. nicht-symmetrische) Relation kann beliebig viele Objekte referenzieren (entsprechend ihrer Stelligkeit). Die Referenzen einer Relation können auch nur teilweise symmetrisch sein. Beispielsweise könnte die Relation „Surrounds“ eine Reihe von Objekten referenzieren, die ein anderes Objekt geographisch umschließen. Sie beschreibt also eine Menge von Gebieten, die gemeinsam einen geschlossenen Kreis um das umschlossene Gebiet bilden.[1] Die Referenzen auf die umschließenden Objekte sind dabei untereinander symmetrisch; zu der Referenz auf das umschlossene Element sind sie jedoch nicht symmetrisch.

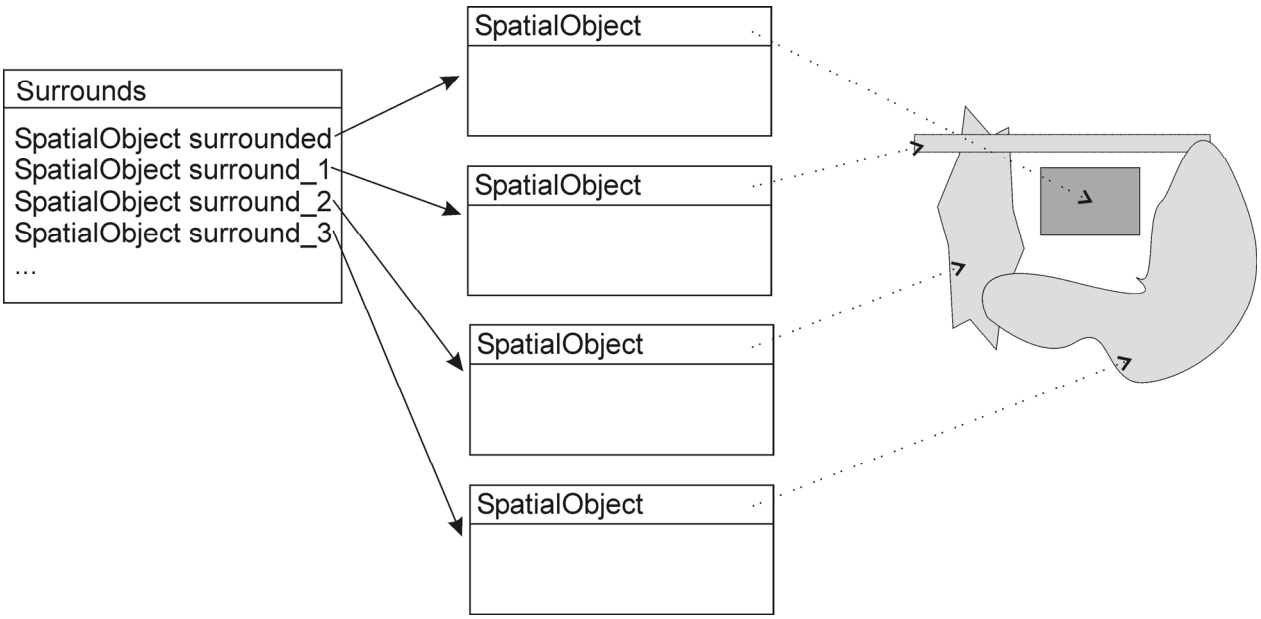


**Abb. 5: Darstellung einer teilweise symmetrischen Relation**

Bezüglich der Modellierung sind nicht-symmetrische Relationen unkritisch, da die einzelnen Objekte anhand des referenzierenden Attributs einer Relation unterschieden werden können, wie in Abb. 4 dargestellt.

Bei symmetrischen Relationen dagegen ergibt sich ein Problem bei dieser Art der Modellierung, das anhand der oben eingeführten Partnerstadtrelation erläutert werden soll. Gegeben sei die symmetrische Relation TwinCity mit zwei Attributen "city1" und "city2", die jeweils Referenzen auf die Partnerstädte darstellen. Wird nun ohne weitere Vorkehrungen eine Relation TwinCity(city1="Stuttgart", city2="Brünn") modelliert, so könnte eine Anfrage nach TwinCity(city1="Brünn", city2="Stuttgart") nicht beantwortet werden, da "Stuttgart" != "Brünn" gilt. Um dieses Problem zu lösen, sind folgende Ansätze möglich:

1. Symmetrische Relation werden mehrfach modelliert, d.h. es erfolgt eine Modellierung sowohl von TwinCity(city1="Stuttgart", city2="Brünn") als auch von TwinCity(city1="Brünn"; city2="Stuttgart").
   - Nachteile: Deutlich höherer Modellierungsaufwand und mögliche Inkonsistenzen
   - Vorteil: Kann problemlos in die bestehende Infrastruktur integriert werden.

[1] Eine solche Relation lässt sich nicht durch eine entsprechende Anzahl zweistelliger Relationen ersetzen, da dabei die Semantik verloren gehen würde.

2. Die Relation wird im Objektschema als symmetrisch gekennzeichnet.. Durch eine Infrastrukturerweiterung kann dann die Anfrage TwinCity(city1="Stuttgart", city2="Brünn") entsprechend aufgelöst und sowohl nach TwinCity(city1="Stuttgart", city2="Brünn") als auch nach TwinCity(city1="Brünn", city2="Stuttgart") gesucht werden (transparent für den Benutzer).
   - Nachteile: Aus einer Anfrage an die Föderation resultieren mehrere Anfragen (bei binären symmetrischen Relationen zwei). Eine Infrastrukturerweiterung ist notwendig.
   - Vorteil: Keine mehrfache Modellierung von symmetrischen Relationen.
3. Die Attribute einer symmetrischen Relation werden als Mengen modelliert. Beispiel: TwinCity(cities={"Stuttgart,Brünn"}).
   - Nachteile: Einführung eines Mengentyps notwendig. Die Stelligkeit einer Menge muss dabei im Schema festgelegt werden können.
   - Vorteile: Keine mehrfache Modellierung von symmetrischen Relationen. Kein erhöhter Aufwand bei der Anfragebearbeitung.
4. Symmetrische Referenzen werden durch Mehrfachattribute (anstatt Mengen) modelliert werden. Beispiel: Zwei Instanzen des „city"-Attributs im TwinCity Objekt.
   - Nachteile: Die Stelligkeit einer Relation kann im Schema nicht festgelegt werden. Im AWM existieren bereits Mehrfachattribute mit einer bestimmten Semantik; es ist keine Unterscheidung zwischen „echten" Mehrfachattributen und durch Mehrfachattribute simulierte Mengen möglich.
   - Vorteil: Keine Infrastrukturerweiterung nötig.

Lösung 1 ist aufgrund ihrer Nachteile nicht attraktiv. Lösung 2 und Lösung 3 haben dieselben Vorteile und erfordern beide eine Erweiterung der Infrastruktur bzw. des AWM. Lösung 3 hat allerdings gegenüber 2 den Vorteil, dass der Aufwand zur Bearbeitung einer entsprechenden Anfrage geringer ist, da es nicht zur unter 2. beschriebenen Vervielfachung der Anfragen kommt. Lösung 4 ist nicht attraktiv, da Mehrfachattribute im AWM bereits vorkommen und eine andere Bedeutung als Mengen haben. Die Einführung eines Mengentyps (Lösung 3) ist damit aus unserer Sicht die beste Lösung.

Bei der Einführung eines Mengentyps in das AWM bzw. in AWML/QL muss beachtet werden, dass bei der Definition eines Mengen-Attributs eine Beschränkung der Kardinalität angegeben werden kann. Dies ist notwendig, um z.B. sicherzustellen, dass eine Partnerstadtbeziehung TwinCity tatsächlich binär ist, d.h. genau zwei Städte in Beziehung setzt. Außerdem sind für Anfragen entsprechende Operationen auf Mengen zu definieren und zu realisieren (Mengengleichheit, Schnitt, Differenz usw.).

### 2.2.3 Typisierung

Eine Referenz auf ein Objekt ist im AWM typischerweise eine NOL. Eine NOL, wie sie bisher in Nexus verwendet wird, kann dabei auf ein beliebiges „NexusDataObject" verweisen. Allerdings ist diese Flexibilität bei Relationen nicht unbedingt erwünscht. Die Referenzen der oben beschriebenen Partnerstadtrelation verweisen typischerweise nur auf Objekte der Klasse „City" oder deren Subklassen. Die Typisierung von Objektreferenzen (NOLs) schließt in solchen Fällen unerwünschte Modellierungen aus. Die Typisierung von NOLs gehört zu den offenen Problemen.

## *2.3 Speicherung/Verwaltung von Relationen*

Grundsätzlich existieren zwei Varianten, um Relationen zu speichern: entweder, sie erhalten eine Koordinate, sodass man sie auch über den Ortsbezug auffinden kann, oder Relationen erhalten keinen Ortsbezug. Die beiden Varianten werden in den folgenden Kapiteln erörtert und anschließend wird festgelegt, für welche Alternative wir uns entschieden haben.

### 2.3.1 Relationen mit Ortsbezug

Ein Relationenobjekt hat eine geometrische Ausdehnung (Extent), die auf definierte Weise aus den referenzierten Objekten (bzw. ihren geometrischen Attributen) gebildet wird. Damit kann eine Relation über den vorhandenen Ortsbezug - wie jedes andere SpatialObject auch - ermittelt werden. Relationen ohne direkten Ortsbezug bzw. zwischen nicht räumlichen Objekten sind allerdings mit diesem Ansatz nicht realisierbar, da der entsprechende Ortsbezug nicht herleitbar ist. Außerdem ist zu beachten, dass der Extent von Objekten, sofern er sich aus der Geometrie (bzw. der konvexen Hülle) der referenzierten Objekte zusammensetzt, bei großen Entfernungen zwischen den referenzierten Objekten so groß wird, dass eine ortsbezogene Suche nach Relationen ebenfalls sehr ineffizient wäre.

Für die Realisierung von ortsbezogenen Relationen sind zwei Varianten denkbar:

**Statischer Ortsbezug:**

Alle Relationen sind Unterklassen von StaticObject – haben also eine feste (statische) geometrische Ausdehnung. Diese Realisierung ermöglicht eine einfache Integration in die vorhandene Nexus-Infrastruktur. Die Herleitung eines statischen Ortsbezugs ist aber nur möglich, wenn auch alle referenzierten Objekte ihrerseits eine feste geometrische Ausdehnung besitzen. Deshalb können an den Relationen nur Objekte des Typs StaticObject teilnehmen. Mobile Objekte – mit einem dynamisch wechselnden Aufenthaltsort – und Objekte ohne Ortsbezug können nicht referenziert werden.

**Dynamischer Ortsbezug:**

Alternativ könnten alle Relationen als Unterklassen von „SpatialObject“ modelliert werden. Dadurch könnten zumindest auch Objekte des Typs „MobileObject“ an Relationen teilnehmen. Allerdings sind für die Verwaltung der Relationen dann (vermutlich sehr umfangreiche) Infrastrukturerweiterungen notwendig, da sich der Ortsbezug einer Relation aufgrund der Mobilität eines referenzierten Objekts ständig ändern könnte. Dieser Ansatz bietet damit bzgl. des Realisierungsaufwands keinen Vorteil mehr gegenüber der Möglichkeit, Relationen komplett ohne Ortsbezug zu definieren.

### 2.3.2 Relationen ohne Ortsbezug

Relationen sind Unterklassen von NexusDataObject. Weil sie damit keinen expliziten Ortsbezug mehr haben, gibt es innerhalb der bisherigen Infrastruktur keine effiziente Möglichkeit, sie zu finden, wenn man die NOL der Relation nicht kennt.

Eine Möglichkeit, dieses Problem zu lösen, besteht im Einfügen einer Rückwärtsreferenz zu jeder Referenz. Die Rückwärtsreferenzen werden auf den Context Servern gespeichert. Jedes Objekt bekommt ein spezielles Attribut, das für jedes auf das Objekt verweisende Relationsobjekt folgende Information enthält:

- NOL des Relationsobjekts
- Typ des Relationsobjekts
- Rolle des Objekts innerhalb der Relation

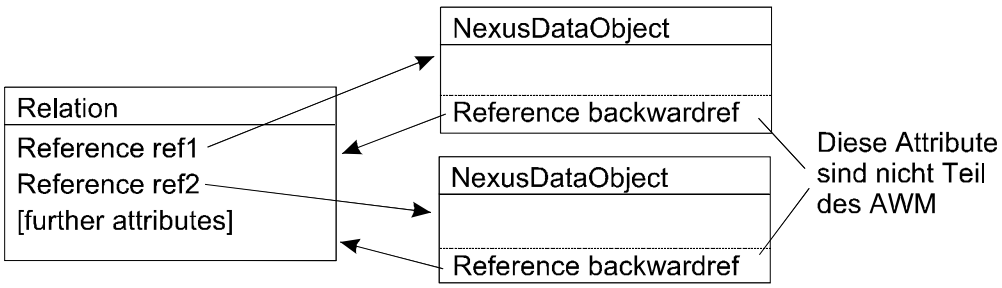


**Abb. 6: Referenzen und Rückwärtsreferenzen bei Relationen.**

Die beiden letzten Informationen sind nicht unbedingt erforderlich, da sie auch aus dem Relationsobjekt abgelesen werden können, allerdings erspart die Speicherung im Objekt häufig den Zugriff auf das Relationsobjekt.

Das Attribut ist nur eine Art Hinweis und nicht Teil der Modelldaten, da es keine neue Information repräsentiert, sondern lediglich ein aus technischen Gründen angelegtes Duplikat der im Relationsobjekt bereits vorhandenen Information ist. Es kann daher nicht durch Anwendungen mittels AWQL modifiziert werden. Context Server aktualisieren den Wert selbstständig untereinander, wenn Relationsobjekte eingefügt bzw. verändert werden. Dazu ist eine Erweiterung der Context Server notwendig: Context Server untersuchen, ob eingefügte bzw. geänderte Objekte Attribute enthalten, die andere Objekte referenzieren (NOLs). Die in Frage kommenden Typen lassen sich aus dem Klassenschema leicht ablesen. Falls ja, wird an jeden Context Server, der ein referenziertes Objekt enthält (diese sind direkt aus den NOLs bestimmbar), eine Nachricht geschickt, die ihn anweist, die Rückwärtsreferenzen seiner Objekte zu aktualisieren. Abb. 7 zeigt den Ablauf einer Einfügeoperation.

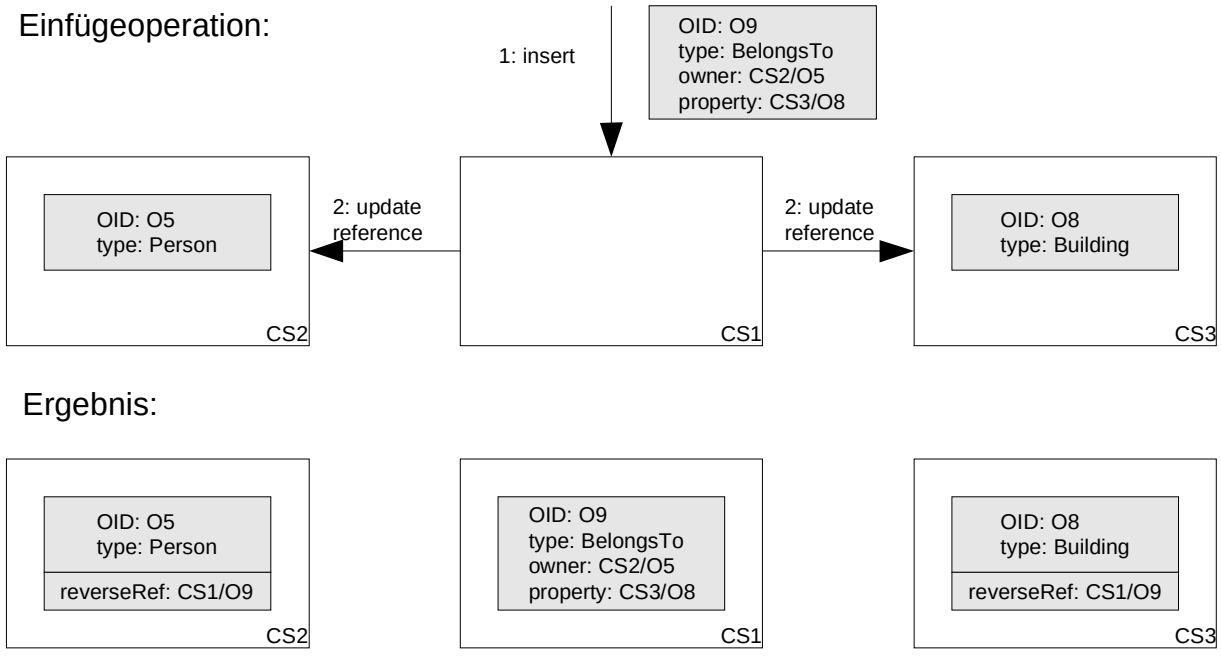


**Abb. 7: Aktualisierung von Rückwärtsreferenzen**

Die Aktualisierung der Rückwärtsreferenzen kann aus verschiedenen Gründen fehlschlagen, z.B. weil ein Context Server nicht erreichbar ist oder Schreibzugriffe generell nicht zulässt. In solchen Fällen sind zwei verschiedene Vorgehensweisen denkbar. Entweder können sich Context Server im Sinne eines „Best Effort"-Ansatzes darauf beschränken, die Rückwärtsreferenzen zu aktualisieren, für die das möglich ist. Anwendungen müssen dann damit rechnen, u.U. unvollständige Ergebnisse auf Anfragen zu erhalten. Alternativ können alle Aktualisierungen von Rückwärtsreferenzen zusammen mit der auslösenden Einfüge- oder Änderungsoperation in einer Transaktion ablaufen, d.h. falls eine der Aktualisierungen fehlschlägt, werden alle anderen Aktualisierungen und die Operation selbst rückgängig gemacht. Bei dieser Variante können Objekte auf unkooperativen Context Servern grundsätzlich nicht an Relationen teilnehmen.

Zunächst ist vorgesehen, dass Context Server bei der Beantwortung von Anfragen die Rückwärtsreferenzen im AWML-Dokument an die entsprechenden Objekte anhängen, so dass sie von den Anwendungen selbst ausgewertet werden. Falls zukünftig auch Joins unterstützt werden sollen, kann die Auswertung auch durch die Plattform vorgenommen werden.

### 2.3.3 Realisierung

Wir haben uns aufgrund der gegebenen Nachteile der ortsbezogenen Speicherung primär für letztere Variante, also für die Speicherung von Relationen ohne Ortsbezug, entschieden. Relationen enthalten allerdings ein optionales Attribut, mittels dessen eine Geometrie bzw. ein Extent spezifiziert werden kann, so dass prinzipiell auch eine direkte ortsbezogene Suche möglich ist (was insbesondere für Relationen zwischen Mehrfachrepräsentationen relevant ist).

Zunächst sollen nur heavyweight Relationen unterstützt werden, da dies keine Einschränkung der Funktionalität bedeutet und eine einheitliche Handhabung ermöglicht. Die vorgeschlagene Realisierung ist allerdings prinzipiell auch in der Lage, lightweight Relationen zu verwalten.

## *2.4 Die Relationenhierarchie*

Aus dem obigen Entschluss, Relationen ohne Ortsbezug zu modellieren bzw. diesen nur optional zu ermöglichen, ergibt sich, dass Relationen direkte Unterklassen von NexusDataObject sind. Die folgende Abbildung (Abb. 8) zeigt die Basisklassen für verschiedene Arten von Relationen.

NexusRelation: Basisklasse aller Relationen

- <optional> extent: räumliche Ausdehnung

Des Weiteren werden folgende Arten von Relationen unterschieden:

- SpatialRelation: Räumliche Beziehungen
  - DistanceRelation: Distanzen zwischen geographischen Objekten
  - DirectionalRelation: Richtungen zwischen geographischen Objekten
  - TopologicalRelation: Topologische Beziehungen
- ThematicRelation: Thematische Relation (im Gegensatz zu räumlichen und temporalen Relationen).
- TemporalRelation: Zeitliche Beziehungen
- MultirepresentationalRelation: Beziehungen zwischen Mehrfachrepräsentationen

Die topologischen Relationen werden in Kapitel 3.3.1 erläutert. Die anderen Klassen von Relationen werden im Folgenden kurz besprochen und nach Bedarf verfeinert.

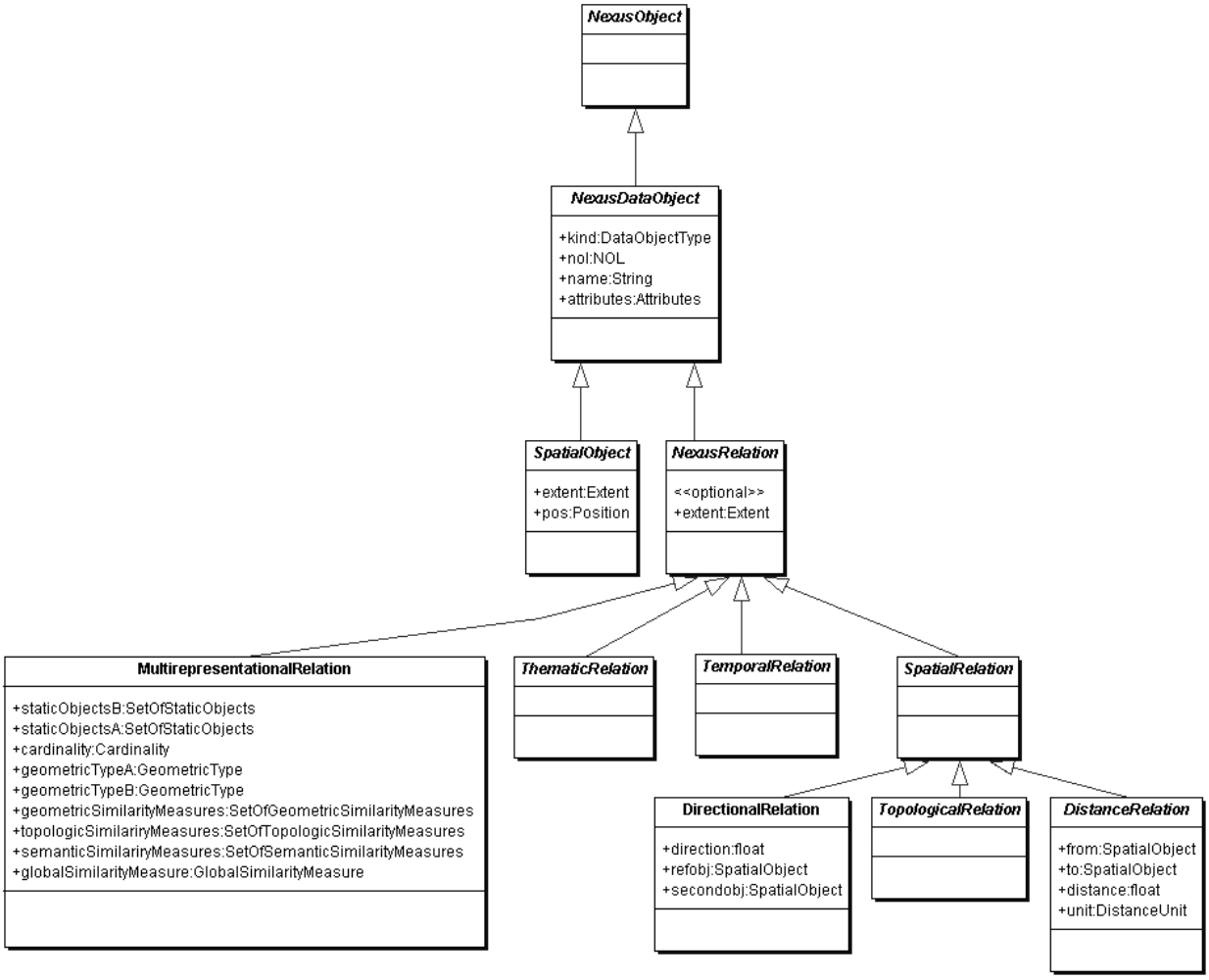


**Abb. 8: Die Relationenhierarchie in Nexus.**

### 2.4.1 Raumbezogene Relationen

Aus dem Raumbezug von Objekten ergeben sich die folgenden Relationstypen:

- Distanzrelationen
- Richtungsbezogene Relationen
- Topologische Relationen

Diese Relationen wurden für den 2D-Bereich erstellt. Begibt man sich in die dritte Dimension, so hat man es z.B. bei der Modellierung von Gebäuden mit Wänden, Türen, Fenstern, Räumen, Möbeln, etc. zu tun. Für die Objekte im 3D-Bereich reichen die topologischen Relationen des 2D-Bereichs vermutlich nicht aus. Daher benötigen wir eine Hierarchie typisierter Relationen für 3D-Objekte, um beispielsweise ausdrücken zu können, dass ein Tisch auf dem Boden steht, etc. Diese Art von typisierten Relationen für 3D-Objekte wird in der nahen Zukunft entwickelt. Die Ergebnisse fließen dann hier ein.

Im Folgenden werden die Distanz-Relationen und die richtungsbezogenen Relationen vorgestellt.

## Distanz-Relationen

Distanz-Relationen beschreiben verschiedene Arten von Distanzen, die zwischen geographischen Objekten definiert werden können. Je nach Kontext kann eine Distanz unterschiedlich definiert sein. Die Unterscheidung erfolgt durch entsprechende Subklassen der allgemeinen Distanz-Relation.

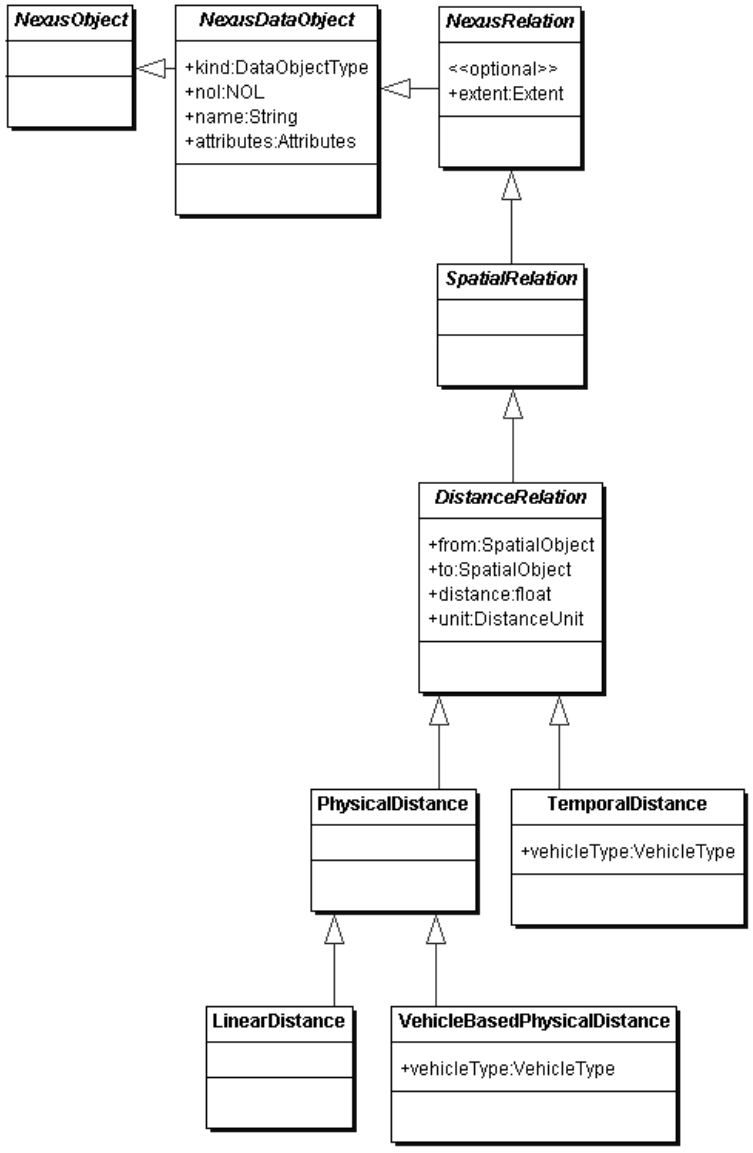


**Abb. 9: Distanz-Relationen in Nexus.**

- DistanceRelation: Allgemeine Distanz zwischen Objekten. Je nach Kontext kann diese unterschiedlich definiert sein (Unterscheidung erfolgt durch Subklassen)
  - from, to: Distanz von ... nach ...
  - distance: Wert der Entfernung
  - unit: Einheit (abhängig von der Art der Entfernung)

- PhysicalDistance: Räumliche Entfernung

- VehicleBasedPhysicalDistance: Zurückzulegende Entfernung bei Verwendung eines bestimmten Fahrzeuges; die Fortbewegungsart „zu Fuß“ (pedestrian) ist ein Sonderfall dieser Entfernungsklasse.

- LinearDistance: Luftlinie

- TemporalDistance: Zeitliche Entfernung (wie lange braucht man von A nach B)
  - vehicleType: Art der Fortbewegung

### Richtungsbezogene Relationen

Richtungsbezogene Relationen beschreiben die Richtung ausgehend von einem Bezugsobjekt. Liegt z.B. das Objekt B östlich bzgl. des Objekts A, so kann dies durch eine DirectionalRelation mit den Attributen refobj = A, secondobj = B, direction = 90 beschrieben werden.

- refobj: Referenzobjekt bzgl. dessen die Richtung angegeben wird

- secondobj: Objekt, dessen Richtung bzgl. des Referenzobjekts beschrieben wird

- direction: Richtung in Grad; Werte wie „östlich von“, „nordwestlich von“, usw. können durch entsprechende Gradangaben (90°, 315°, ...) angegeben werden

## 2.4.2 Thematische Relationen

Thematische Relationen beschreiben Beziehungen zwischen Objekten, die nicht in erster Linie räumlich oder zeitlich bedingt sind.

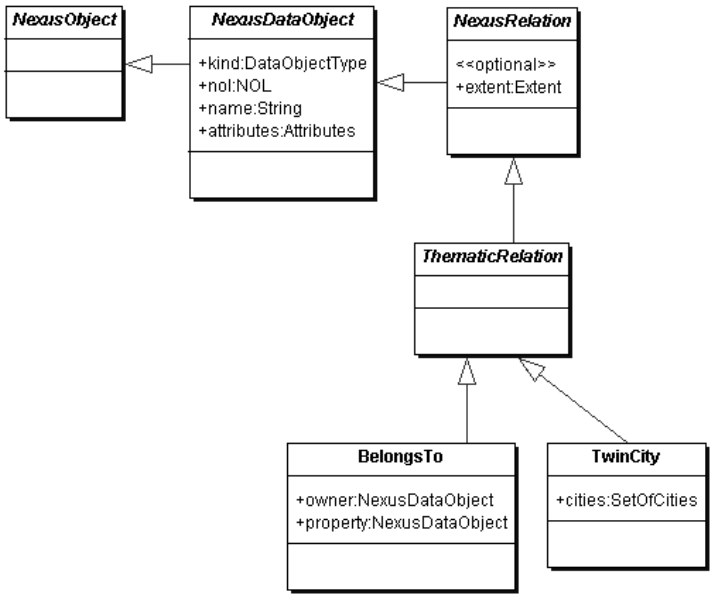


**Abb. 10: Thematische Relationen in Nexus.**

## 2.4.3 Temporale Relationen

Temporale Relationen drücken einen zeitlichen Bezug zwischen den temporalen Basistypen Zeitpunkt und Zeitraum bzw. Zeitdauer aus. Basis für die Definition von temporalen Relationen sind z.B. die Arbeiten von Allen (Beschreibung der Relationen zwischen zwei Zeiträumen in [Allen 1983]).

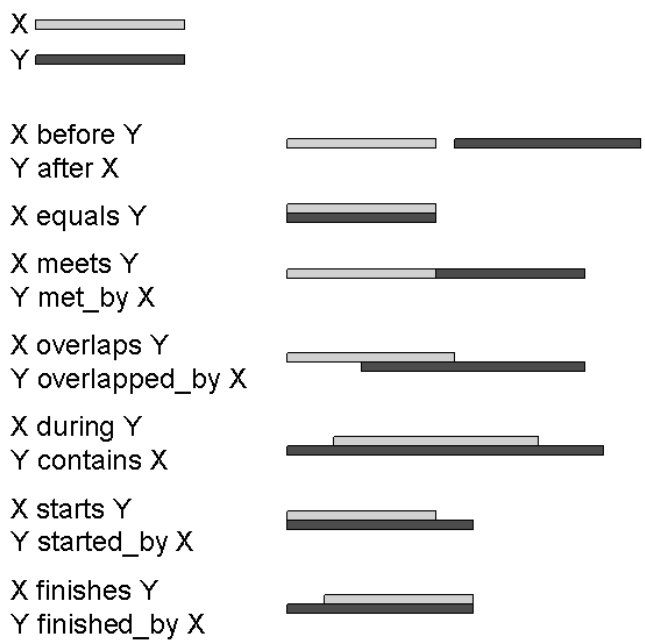


**Abb. 11: Die Allen-Operatoren zur Beschreibung temporaler Relationen zwischen Zeiträumen [Allen 83].**

Die folgenden Basis-Objekttypen für Zeitraum-Relationen (und analog, sofern sinnvoll, für Zeitpunkte) sollen in unser Modell aufgenommen werden:

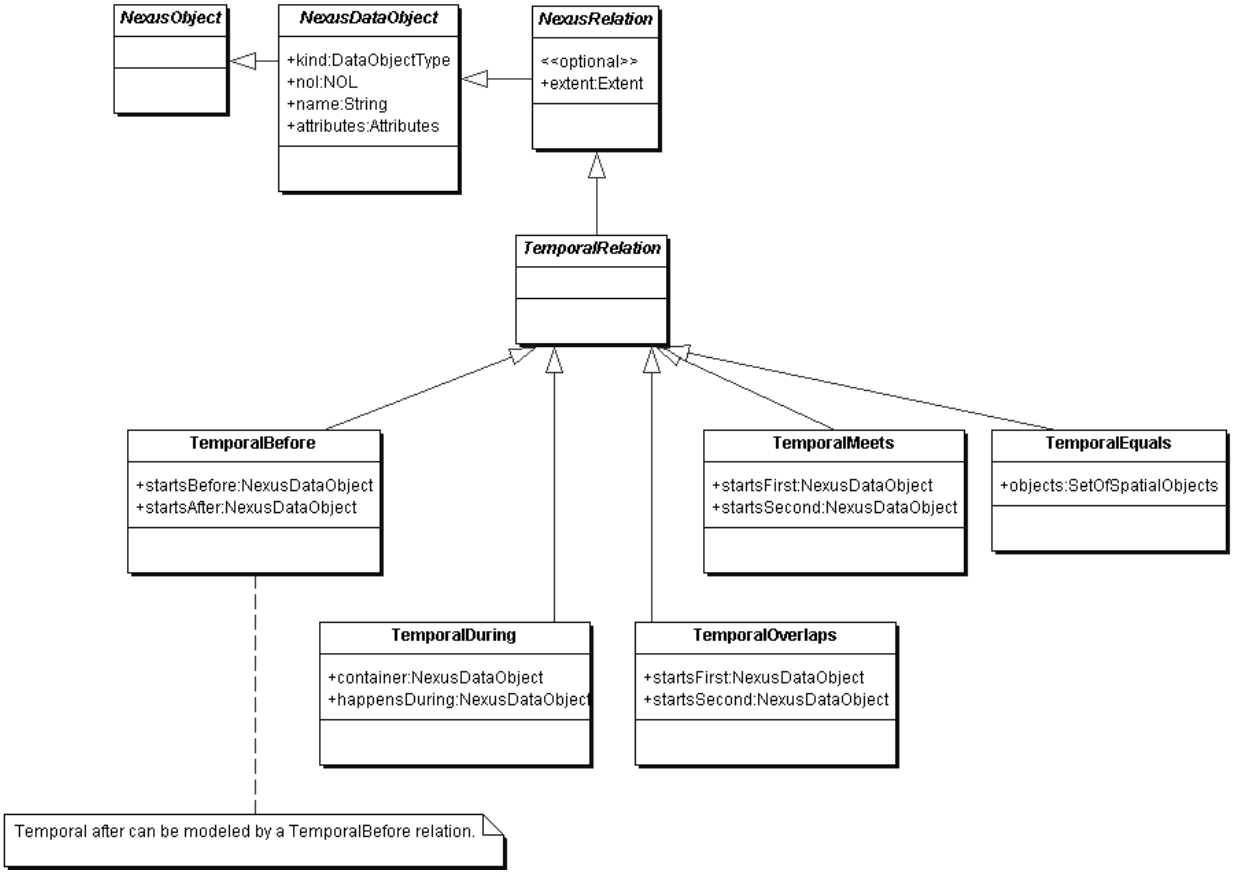


**Abb. 12: Zeitliche Relationen in Nexus.**

- TemporalBefore: Zeitliche Vorgängerbeziehung; das zweite Ereignis findet nach Beendigung des ersten statt.
  - startsBefore: das zeitlich erste Ereignis
  - startsAfter: das zeitlich auf das erste folgende Ereignis
  - Inverse Relation: TemporalAfter

- TemporalDuring: Ein Ereignis erfolgt während eines anderen Ereignisses.
  - container: Ereignis, das das zweite Ereignis zeitlich umgibt
  - happensDuring: das Ereignis, das während des ersten Ereignisses stattfindet
  - Inverse Relation: TemporalContains

- TemporalEqual: Zeitliche Gleichzeitigkeit
  - objects: zweistellige Menge von Ereignissen, die gleichzeitig ablaufen
  - Inverse Relation: Keine inverse Relation, da symmetrisch.

- TemporalOverlaps: Zeitliche Überlappung; das zweite Ereignis startet, nachdem das erste Ereignis gestartet, aber bevor das erste Ereignis beendet ist.
  - startsFirst: erstes Ereignis
  - startsSecond: zweites Ereignis
  - Inverse Relation: TemporalContains

Folgende (ebenfalls zu den Allen-Operatoren gehörende) Relationen können bei Bedarf ebenfalls in unser Modell aufgenommen werden:

- TemporalMeets: Zeitlicher Anschluss; das zweite Ereignis startet sofort bei Beendigung des ersten.
  - startsFirst: erstes Ereignis
  - startsSecond: zweites, anschließendes Ereignis
  - Inverse Relation: TemporalMetBy

- TemporalStarts: Zwei Ereignisse starten zur selben Zeit, das erste Ereignis wird früher beendet.
  - stopsFirst: erstes Ereignis
  - stopsSecond: zweites Ereignis
  - Inverse Relation: TemporalStartedBy

- TemporalFinishes: Zwei Ereignisse werden zur selben Zeit beendet, das zweite Ereignis startet später. (Achtung: Bei Allen startet das erste Ereignis später als das zweite!)
  - startsFirst: erstes Ereignis
  - startsSecond: zweites Ereignis
  - Inverse Relation: TemporalFinishedBy

Als Basis-Objekttypen für Relationen zwischen Zeitdauern sollen aufgenommen werden:

- DurationShorterThen: Zeitdauer 1 ist kleiner als Zeitdauer 2.
  - durationShorterThen: Objekt 1 mit kleinerer Zeitdauer.
  - durationLongerThen: Objekt 2 mit größerer Zeitdauer.
  - Inverse Relation: DurationLongerThen
- DurationEqual: Die Zeitdauern sind gleich.
  - objects: Menge von Objekten gleicher Zeitdauer.

Für alle Relationen-Modellierungen gilt, dass inverse Relationen nicht durch eigene Objekttypen modelliert werden müssen, sondern durch die nicht-inverse Relation mit vertauschten Attributen dargestellt werden können.

### 2.4.4 Relationen zwischen mehrfach repräsentierten Objekten

In Nexus kommt es aufgrund der Offenheit des Systems, die eine Mehrfachspeicherung ein und desselben Realweltobjektes innerhalb der Plattform zulässt, zur Problematik der Mehrfachrepräsentationen. Zum Zwecke der optimierten Verwaltung (gemeinsame Fortführung) und optimierten Verarbeitung (Verschmelzung zur Erzeugung einer konsistenten Datenbasis, Analyse) von Mehrfachrepräsentationen sollen auch deren Relationen untereinander explizit modelliert werden. Das bislang entwickelte Modell wurde für die Modellierung der Beziehungen raumbezogener, mehrfach repräsentierter Daten konzipiert.

Zwei Mehrfachrepräsentationen sind über eine Instanz der Klasse MultirepresentationalRelation miteinander zu verbinden. Diese Klasse hat keine Subklassen, sondern beschreibt in ihren Attributen die Art der Relation. Ein Teil der Attribute besteht aus Ähnlichkeitsmaßen, die den Grad der Übereinstimmung der Repräsentationen charakterisieren.

- MultirepresentationalRelation: Zwei Repräsentationen A und B sind über eine MultirepresentationalRelation miteinander verbunden.
  - SetOfStaticObjects staticObjectsA: Objekte, die die erste Repräsentation aufbauen.
  - SetOfStaticObjects staticObjectsB: Objekte, die die zweite Repräsentation aufbauen.
  - Cardinality cardinality: Kardinalität der Relation von 1:0 bis n:m.
  - GeometricTypeA: Geometrischer Typ der Repräsentation A (Point, MultiPoint, LineString, ...).
  - GeometricTypeB: Geometrischer Typ der Repräsentation B (Point, MultiPoint, LineString, ...). Bislang wird vorausgesetzt, dass alle Objekte einer Repräsentation denselben geometrischen Typ aufweisen müssen.
  - SetOfGeometricSimilarityMeasures geometricSimilarityMeasures: Sammlung verschiedener geometrischer Ähnlichkeitsmaße, abhängig vom geometrischen Typ (z.B. Längenunterschied oder durchschnittliche Liniendistanz bei Linien, Hausdorff-Distanz bei Linien und Flächen, Zentroiddistanz bei Flächen, etc.).
  - SetOfTopologicSimilarityMeasures topologicSimilarityMeasures: Sammlung verschiedener topologischer Ähnlichkeitsmaße (z.B. Anzahl adjazenter Objekte, graphbasierte Indikatoren wie Erreichbarkeit oder Exzentrizität von Knoten, etc.).
  - SetOfSemanticSimilarityMeasures semanticSimilarityMeasures: Sammlung verschiedener semantischer Ähnlichkeitsmaße (z.B. Anzahl korrespondierender Attribute, Übereinstimmung von Angaben zur Datenqualität).
  - GlobalSimilarityMeasure globalSimilarityMeasure: Ähnlichkeitsmaß, das aus der Berechnung der Teilmaße resultiert.

## *2.5 Offene Probleme*

### 2.5.1 Relationen zwischen Relations-Objekten

Es ist bisher noch nicht klar, ob es auch Relationen zwischen Relations-Objekten geben wird.

### 2.5.2 Mehrfach repräsentierte Relationen

Die Problematik der mehrfach repräsentierten Relationen wird zunächst nicht betrachtet.

### 2.5.3 Typisierung von NOLs

Unter typisierten NOLs verstehen wir Referenzen, die nur auf AWM-Objekte eines bestimmten Typs verweisen können. Erste Überlegungen bzgl. der möglichen Realisierung solcher NOLs ergaben die folgenden Implementierungsvarianten:

- NOL-Hierarchie: Zu jeder AWS-Klasse wird ein entsprechender NOL-Typ eingeführt. Durch die Ableitung neuer NOL-Typen von NOL-Supertypen entsprechend der Vererbungshierarchie des AWS wird sichergestellt, dass eine NOL, die auf ein Objekt der Klasse K1 verweist, auch auf Objekte verweisen kann, die Subtypen der Klasse K1 darstellen (ähnlich den typisierten Zeigern oder Referenzen in gängigen objektorientierten Programmiersprachen wie C++ oder Java). Beispielsweise könnte damit eine MobileObject-NOL auch auf Objekte vom Typ „Person“ oder „Vehicle“ verweisen.

- Festlegung des Typs einer NOL als Meta-Attribut im Schema

Da die Diskussion der Vor- und Nachteile und Auswirkungen dieser und evtl. weiterer Ansätze den Rahmen dieses Reports sprengen würde, wird dieses Thema hier nicht weiter vertieft.

# 3 Das Topologie-Konzept in Nexus

Topologie-Konzepte existieren in verschiedenen Forschungsbereichen. An dieser Stelle sollen all jene Arbeiten aufgeführt werden, die bei der Diskussion um die Entwicklung eines Topologie-Konzeptes für die Nexus-Plattform eine Rolle gespielt haben, bevor dieses schließlich selbst erläutert wird.

## *3.1 Verwandte Arbeiten*

Die folgenden Ansätze wurden beim Entwurf des Nexus-Topologie-Konzepts untersucht.

### 3.1.1 Das 9-Intersection Model und das erweiterte 9-Intersection Model

Um die topologischen Beziehungen von raumbezogenen Objekten abzuspeichern, entwickelten [Egenhofer und Herring 1991] das so genannte 9-Intersection Model. Es geht davon aus, dass Objekte in Geo-Informationssystemen definierte, scharfe und eindeutige Grenzen haben. Diese Annahme stimmt aber nicht mit der Realität überein. Daher haben [Clementini und Di Felice 1996] versucht, das 9-Intersection Model anzupassen. Im folgenden sollen die beiden Ansätze kurz skizziert werden.

#### Das 9-Intersection Modell nach Egenhofer und Herring:

Egenhofer und Herring gehen bei ihrem Ansatz von folgenden Gegebenheiten aus:

- eine Fläche ist eine geschlossene homogene zweidimensionale Teilmenge des $\Re^2$
- eine Linie ist eine eindimensionale homogene Teilmenge im $\Re^2$ mit exakt zwei Endpunkten und keinen Selbstüberschneidungen
- ein Punkt ist eine 0-dimensionale Teilmenge des $\Re^2$
- für Punkt, Linie und Fläche lassen sich folgende topologischen Beziehungen eindeutig definieren
  - innerhalb (0)
  - Grenze ($\delta$)
  - außerhalb (-)
- Bsp.: Fläche A

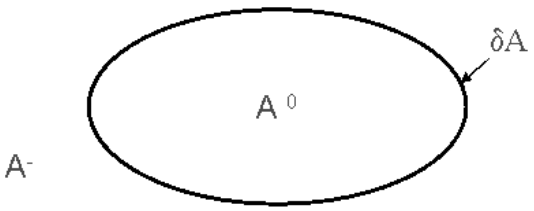


**Abb. 13: Topologisch relevante Einheiten einer Fläche: Inneres, Äußeres und Grenze.**

Alle topologischen Relationen zwischen zwei Objekten A und B lassen sich klassifizieren, indem man alle Schnittmengen zwischen dem Inneren (0), der Grenze ($\delta$) und dem Äußeren (-) von A und B betrachtet. Daraus ergibt sich eine 3 x 3 Matrix: die 9-intersection Matrix (siehe Abb. 14).

$$\begin{pmatrix} A^0 \cap B^0 & A^0 \cap \delta B & A^0 \cap B^- \\ \delta A \cap B^0 & \delta A \cap \delta B & \delta A \cap B^- \\ A^- \cap B^0 & A^- \cap \delta B & A^- \cap B^- \end{pmatrix}$$

**Abb. 14: Die 9-Intersection Matrix.**

Jeder Eintrag in der Matrix kann entweder leer (0) oder nicht-leer (1) sein: es kann daher zwischen $2^9 = 512$ verschiedenen topologischen Relationen unterschieden werden. Für zwei einfache Flächen sind davon jedoch nur 8 realisierbar:

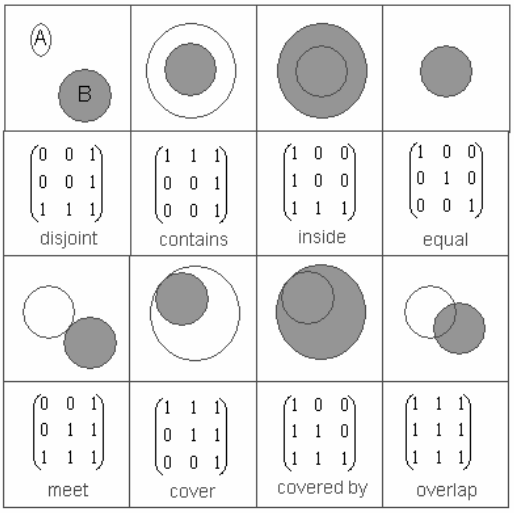


**Abb. 15: Mögliche topologische Beziehungen zwischen einfachen Flächen.**

## Das erweiterte 9-Intersection Model nach Clementini und Di Felice:

Bei Clementini und Di Felice werden nun die Grenzen von Objekten als Flächen repräsentiert:

- Definition 1: eine Fläche mit unscharfer Begrenzung A wird aus zwei Flächen $A_1$ und $A_2$ gebildet, für die gilt: $A_1 \subseteq A_2$ und $\delta A_1$ ist innere Begrenzung von A und $\delta A_2$ ist äußere Begrenzung von A. Die beiden Begrenzungen repräsentieren die Unschärfe einer Region und zeigen deren Minimal- und Maximalausdehnung an.

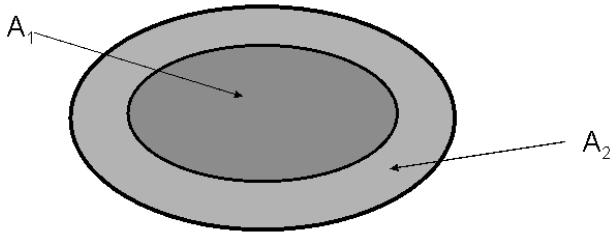


**Abb. 16: Objekte können unscharfe Grenzen haben, daher können auch Objektgrenzen als Flächen repräsentiert werden.**

- Definition 2: die unscharfe Grenze $\Delta A$ einer Region A ist eine zusammenhängende Teilmenge des $\Re^2$ mit einem Loch
  - $\Delta A$ ist die Fläche zwischen innerer und äußerer Grenze von A: $\Delta A = A_2 - A_1$
  - falls $A_1 = A_2$ dann ist A ein eindimensionaler Kreis
  - falls $\delta A_1 \cap \delta A_2 \neq \varnothing$ ist, ist A nicht homogen zweidimensional und kann eindimensionale Teile enthalten

- Definition 3: das Innere einer Region mit unscharfer Grenze ist: $A^0 = A_2 - \Delta A$
- Definition 4: Das Äußere einer Region mit unscharfer Grenze ist: $A^- = \Re^2 - A_2$

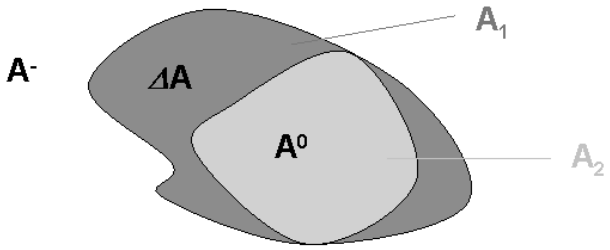


**Abb. 17: Beschreibung von Innerem, Äußerem und Grenze einer unscharfen Fläche.**

Um alle möglichen topologischen Relationen zwischen zwei Flächen mit unscharfen Grenzen zu bestimmen, wird das 9-Intersection Modell folgendermaßen umdefiniert:

$$\begin{pmatrix} A^0 \cap B^0 & A^0 \cap \Delta B & A^0 \cap B^- \\ \Delta A \cap B^0 & \Delta A \cap \Delta B & \Delta A \cap B^- \\ A^- \cap B^0 & A^- \cap \Delta B & A^- \cap B^- \end{pmatrix}$$

**Abb. 18: Die erweiterte 9-Intersection Matrix.**

Nun muss wiederum bestimmt werden, welche der 512 verschiedenen Kombinationsmöglichkeiten realisierbar sind. Ebenso wie bei Flächen mit scharfen Grenzen gibt es bei Flächen mit unscharfen Grenzen geometrische Bedingungen, die auf jeden Fall erfüllt sein müssen; z.B.: die äußeren Flächen müssen sich auf jeden Fall schneiden. Insgesamt lassen sich 12 unterschiedliche Bedingungen aufstellen, die die $2^9$ Matrix auf 44 mögliche topologische Relationen reduzieren.

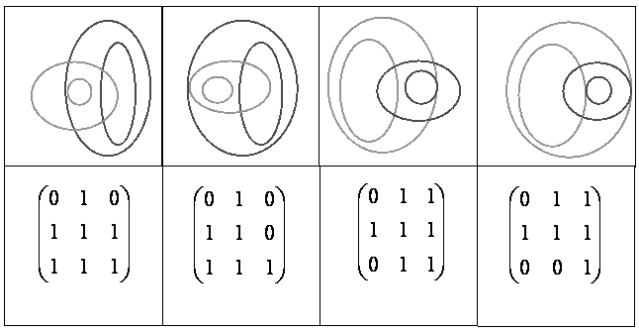


**Abb. 19: Einige der möglichen topologischen Varianten zwischen unscharfen Flächen.**

Von den 44 möglichen Kombinationen beschreiben viele sehr ähnliche Sachverhalte, sodass es hier für die meisten Fälle nicht nötig ist, zwei verschiedene topologische Relationen zu unterscheiden. Da die 44 topologischen Relationen sich zum Teil also nur graduell unterscheiden, werden sie in 12 Cluster eingeteilt:

- **disjoint, meet, nearly meet, nearly overlap, overlap, nearly covered by, nearly covers, covered by, covers, inside, contains, equal**

### 3.1.2 Ansätze aus dem Bereich Ubiquitous Computing

Im Ubiquitous Computing werden Ortsinformationen als eine wesentliche Kontextinformation angesehen. Ortsinformationen werden dabei auf Grundlage eines so genannten Lokationsmodells interpretiert, das auch topologische Informationen enthalten kann. Diese Lokationsmodelle sind meist auf wenige, spezielle Fragestellungen zugeschnitten. Zwei Klassen von Modellen sind bzgl. der Modellierung topologischer Relationen von Bedeutung:

Hierarchische Modelle basieren auf einer Hierarchie von Orten, die bzgl. der räumlichen Inklusion geordnet sind, d.h. die Inklusionsbeziehung ist hier die zentrale topologische Relation. Geht man davon aus, dass Orte sich nicht überlappen, so führt dieser Ansatz zu einem Baum [Jiang und Steenkiste 02]; können sich Orte überlappen, so führt dieser Ansatz zu einem Verband [Dürr und Rothermel 03].

Graphbasierte Modelle modellieren Orte und die Verbindung zwischen Orten. Solche Modelle eignen sich vor allem für die Navigation oder die Suche nach nächstgelegenen Objekten (Nachbarschaftsanfragen).

### 3.1.3 Region Connection Calculus

Im Bereich des qualitativen räumlichen Schließens (Qualitative Spatial Reasoning) werden verschiedene Kalküle eingesetzt, um räumliche Entitäten zu modellieren und Schlussfolgerungen über deren räumliche Beziehungen herzuleiten. Hierbei spielen auch topologische Beziehungen eine wichtige Rolle. Der Region Connection Calculus 8 (RCC8) unterstützt acht dieser topologischen Beziehungen [Randell und Cohn 89]:

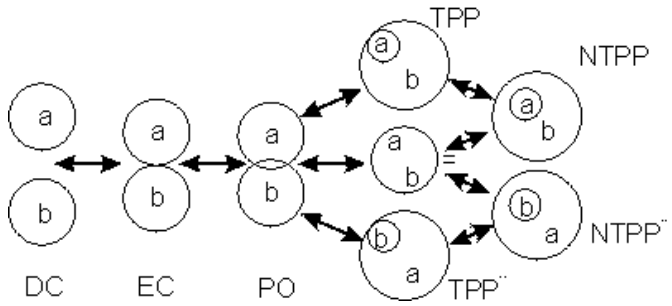


**Abb. 20: Topologische Beziehungen aus dem Bereich des Region Connection Calculus.**

### 3.1.4 OGC Topologie

In der abstrakten Spezifikation des Open GIS Consortium (OGC) [OGC99] werden zunächst die verschiedenen topologischen Objekte bzw. deren Typen eingeführt, die durch topologische Relationen in Beziehung gesetzt werden können. Das OGC unterscheidet zwischen den primitiven topologischen Objekten Node (Dimension 0), Edge (Dimension 1), Face (Dimension 2) und Solid (Dimension 3). Neben diesen primitiven Objekten existiert der Typ Complex, der mehrere topologische Primitive enthält. Diese topologischen Objekte können mit den entsprechenden geometrischen Objekten in Relation stehen. Die OGC Implementierungsspezifikation [OGC03] beschreibt die Umsetzung dieser topologischen Objekte in GML. Hier ein einfaches Beispiel:

```
<gml:PointProperty gml:id="point1">
  <gml:Point>
    <gml:coordinates>10, 10</gml:coordinates>
  </gml:Point>
</gml:PointProperty>
<gml:PointProperty gml:id="point2">
  <gml:Point>
    <gml:coordinates>20, 20</gml:coordinates>
  </gml:Point>
</gml:PointProperty>

<gml:Node gml:id="node1">
  <gml:PointProperty xlink:href="#point1">
</gml:Node>
<gml:Node gml:id="node2">
  <gml:PointProperty xlink:href="#point2">
</gml:Node>

<gml:Edge gml:id="edge1">
  <gml:directedNode orientation="-" xlink:href="#node1"/>
  <gml:directedNode orientation="+" xlink:href="#node2"/>
</gml:Edge>
```

**Abb. 21: Auszug aus GML: Realisierung topologischer Objekte.**

Es werden hier drei topologische Objekte definiert: zwei Nodes und eine Edge. Die Edge wird durch Referenzen auf die begrenzenden Nodes definiert (das Attribut "orientation" definiert, welcher Knoten der Startknoten (-) und welcher der Endknoten (+) der Kante ist). Die Nodes referenzieren die entsprechenden Geometrieobjekte (Point Features). Allerdings könnten die Nodes auch ohne diese Geometriebeschreibung definiert werden. Außerdem besteht die Möglichkeit, die Coboundaries der topologischen Objekte durch Referenzen auf die entsprechenden Topologieobjekte zu spezifizieren, also die Objekte, deren Boundary das topologische Objekt definiert (die Coboundary eines Node ist somit eine Menge von Edges, die den Knoten als Start- oder Endpunkt enthalten).

Auf topologischen Objekten sind verschiedene Operationen definiert:

- boundary(): Ermittelt alle topologischen Objekte, die auf dem Rand eines bestimmten topologischen Objekts liegen. So wird z.B. die Boundary einer Face durch eine Menge von Edges bestimmt oder im obigen Beispiel die Boundary einer Edge durch zwei Nodes. Allgemein ist die Begrenzung eines topologischen Objekts der Dimension n durch ein Objekt der Dimension n-1 definiert.

- interior(): Ermittelt alle topologischen Objekte, die im Inneren eines bestimmten topologischen Objekts liegen.

- exterior(): Ermittelt alle topologischen Objekte, die außerhalb eines bestimmten topologischen Objekts liegen (d.h. weder zur Menge Interior noch zur Menge Boundary gehören).

In GML ist die Boundary durch entsprechende Referenzen auf topologische Objekte definiert. So definieren z.B. die Knoten "node1" und "node2" im obigen Beispiel die Boundary der Kante "edge1". Entsprechend ist die Boundary höherdimensionaler Objekte definiert.

Die topologischen Relationen selbst werden laut [OGC99] durch den Schnitt der Mengen Boundary, Interior und Exterior definiert. Somit können z.B. die Relationen des 9-Intersection-Modells nach Egenhofer oder die Relationen nach Clementini und Di Felice ermittelt werden (siehe oben). Zur konkreten Umsetzung in GML - insbesondere zur notwendigen Ermittlung der Mengen Exterior und Interior - macht [OGC03] allerdings keine Aussagen.

## 3.2 Diskussion der verschiedenen Ansätze

Die verschiedenen oben dargestellten Ansätze sollen nun in diesem Abschnitt auf ihre Anwendbarkeit im Nexus-Kontext hin analysiert werden.

Betrachtet man die oben beschriebenen Ansätze, so erscheinen vor allem das 9-Intersection-Modell nach Egenhofer und das Modell nach Clementini und Di Felice (und somit der Ansatz des OGC, der sich an diesen beiden Ansätzen orientiert) als sehr mächtig. Sie unterstützen jeweils eine große Zahl topologischer Relationen und erlauben dadurch eine feingranulare Unterscheidung von topologischen Beziehungen. Der Ansatz aus dem Bereich des Qualitative Spatial Reasoning (RCC8) ist bzgl. der betrachteten Relationen sehr ähnlich zu diesen Ansätzen. Die Ansätze aus dem Bereich des Ubiquitous Computing dagegen sind auf spezielle Fragestellungen zugeschnitten. Entsprechend einfach und wenig generisch sind die resultierenden Modelle.

Die Mächtigkeit z.B. des 9-Intersection-Modells hat allerdings auch ihren Preis: Ohne eine vorhandene geometrische Modellierung, auf deren Grundlage z.B. die Punkte auf der Begrenzung eines Objekts und die Punkte im Inneren eines Objekts definiert werden können, scheint die Modellierung entsprechender Bereiche und somit die Herleitung der Relationen nur sehr schwer möglich. Das wird auch in den Spezifikationen der OGC deutlich. Obwohl der OGC-Standard es zulässt, topologische Objekte auch ohne die entsprechenden geometrischen Objekte zu definieren, so zeigen doch allein schon die Klassen der topologischen Objekte wie z.B. Node, Edge, usw., dass dieser Ansatz vor allem zusammen mit bekannten Geometrien zum Einsatz kommen sollte. Die Ansätze aus dem Bereich des Ubiquitous Computing dagegen erlauben rein aufgrund der vergleichsweise groben Modellierung topologischer Relationen auch den Einsatz ohne vorhandene Geometrie. So lässt sich die räumliche Inklusions-Beziehung z.B. ohne Probleme explizit auf Objekten definieren, ohne deren Geometrie zu kennen.

Das Nexus-AWM unterstützt sowohl geometrisch modellierte Objekte als auch (zumindest zukünftig) räumliche Objekte ohne explizit definierte Geometrie. Für den ersten Fall erscheint eine Unterstützung der mächtigen Ansätze (z.B. des 9-Intersection-Modells) als sinnvoll, da die entsprechenden Grundmengen (Interior, Exterior, Boundary) aus der Geometrie ableitbar sind. Zur Unterstützung des 9-Intersection-Modells oder dem Modell nach Clementini und De Felice muss das AWM zunächst um entsprechende topologische Objekte erweitert werden. Diese Erweiterungen werden im folgenden Abschnitt beschrieben. Liegt dagegen keine Geometrie vor, so muss eine explizite Modellierung u.U. vergleichsweise einfacher topologischer Relationen möglich sein - ähnlich den oben beschriebenen Ansätzen aus dem Bereich des Ubiquitous Computing. Die von Nexus unterstützten topologischen Relationen werden im übernächsten Abschnitt genauer beschrieben.

## 3.3 Topologie in Nexus

Das entwickelte Konzept zur Abbildung von Topologie in Nexus beschreibt zunächst die topologischen Objekte und danach deren Relationen. Zum besseren Verständnis steht am Ende ein Beispiel.

### 3.3.1 Topologische Objekte in Nexus

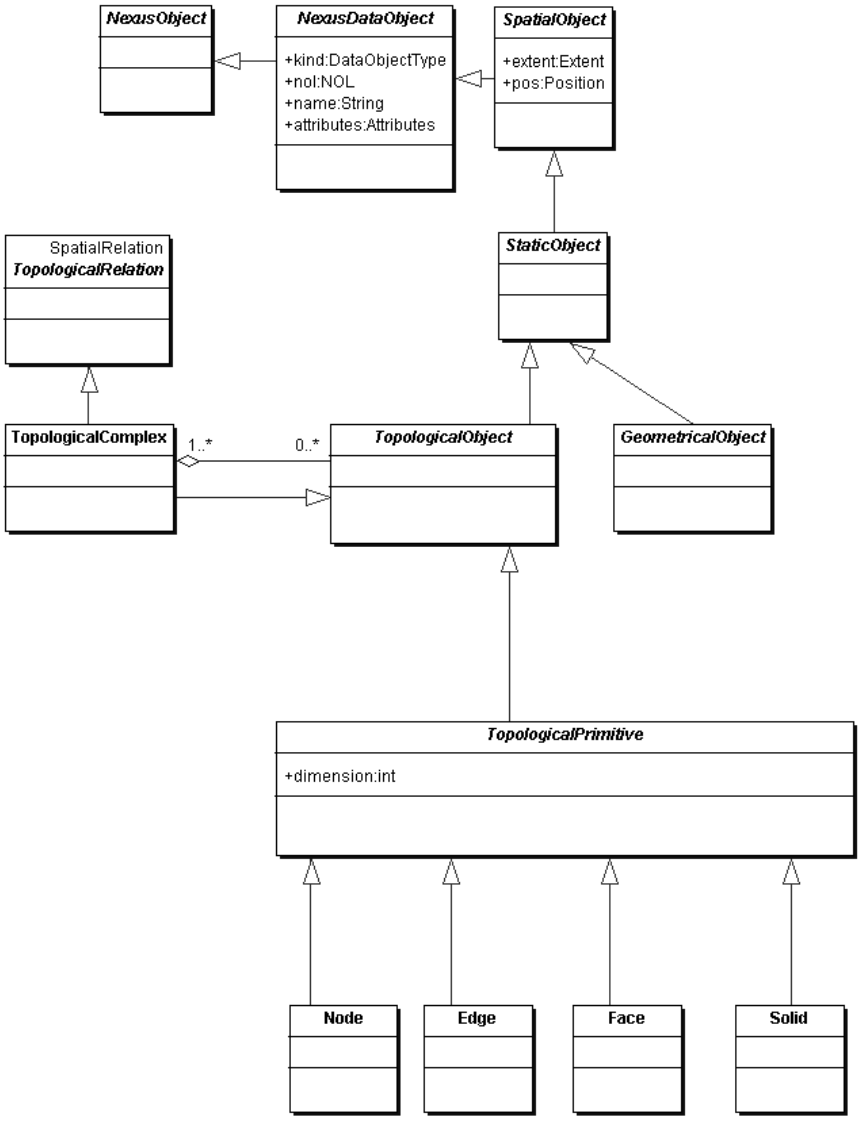


**Abb. 22: Topologische Objekte in Nexus.**

Die in Nexus modellierten topologischen Objekte orientieren sich an denen vom OGC vorgeschlagenen (siehe oben):

- TopologicalObject: Basisklasse für alle topologischen Objekte
- TopologicalPrimitive: 0- bis 3-dimensionales topologische Primitive
  - dimension: Dimension (0-3)
- TopologicalComplex: Eine Menge von TopologicalObjects ohne gemeinsame Interiors.
- Node: Knoten (0-dimensionales topologisches Primitiv)
- Edge: Kante (1-dimensionales topologisches Primitiv)
- Face: Fläche (2-dimensionales topologisches Primitiv)
- Solid: 3d Körper (3-dimensionales topologisches Primitiv)

### 3.3.2 Topologische Relationen in Nexus

Die vom AWM unterstützten topologischen Relationen sind untenstehend abgebildet.

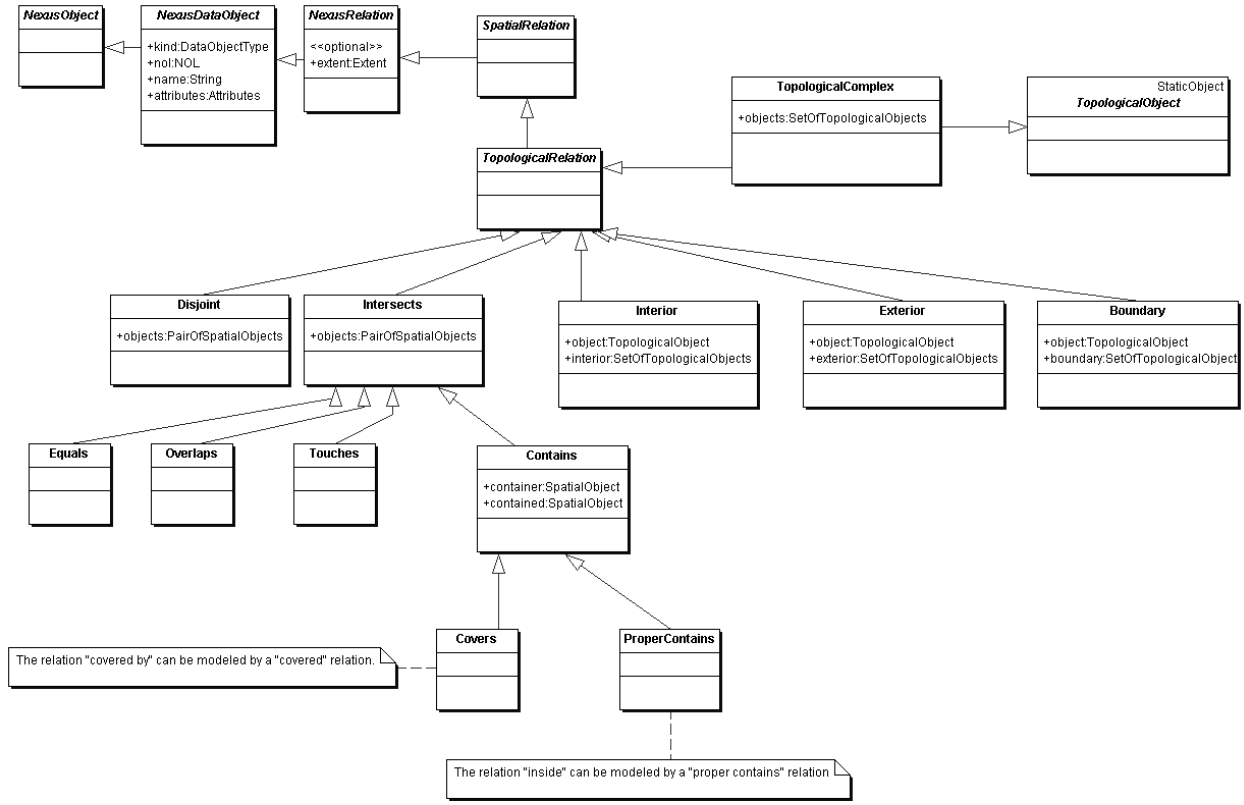


**Abb. 23: Topologische Relationen in Nexus.**

Wie bereits angedeutet, war es bei der Entwicklung des Topologie-Modells das Ziel, sowohl eine feingranulare Topologiebeschreibung vergleichbar dem 9-Intersection-Modell zu ermöglichen, als auch einfache Relationen anzubieten, die sich problemlos ohne bekannte Geometrie explizit modellieren lassen. Wir erreichen das durch eine Hierarchie topologischer Relationen. Die Relation "Contains" lässt sich beispielsweise sehr leicht explizit modellieren. Ihre Ableitungen "Covers" und "ProperContains" stellen dabei Spezialfälle dieser allgemeinen Enthaltensein-Beziehung dar, die sich typischerweise aus einer vorhandenen Geometriebeschreibung ableiten lassen.

- TopologicalRelation: Basisklasse für alle topologischen Relationen

- Interior, Exterior, Boundary: Basismengen bzw. Relationen des 9-Intersection-Modells, auf deren Grundlage die topologischen Relationen des 9-Intersection-Modells abgeleitet werden.

- TopologicalAggregate: Relation zur Definition von topologischen Aggregaten. Durch die Verwendung von Mehrfachvererbung ist es möglich, dass ein TopologicalAggregate-Objekt direkt andere TopologicalAggregate-Objekte über das Attribut „objects“ referenziert. Das heißt, ein TopologicalAggregate-Objekt kann sich aus anderen TopologicalAggregate-Objekten zusammensetzen. Eine alternative Modellierung, die ohne Mehrfachvererbung auskommt, könnte durch die Einführung einer zusätzlichen Relation „TopologicalAggregateRel“ erzielt werden. Relationen dieses Typs könnten dazu verwendet werden, von einem TopologicalAggregate-Objekte aus andere TopologicalAggregate-Objekten zu referenzieren.

    - objects: Topologische Objekte, aus denen sich das aggregierte Objekt zusammensetzt.

- TopologicalComplex: Relation zur Definition von komplexen topologischen Objekten. Wie für die Klasse TopologicalAggregate gilt auch hier, dass sich ein Topological-Complex-Objekt selbst aus weiteren TopologicalComplex-Objekten zusammensetzen kann.
  - objects: Topologische Objekte, aus denen sich das komplexe Objekt zusammensetzt.
- Disjoint: Diese Relation beschreibt zwei disjunkte Objekte
  - objects: die disjunkten Objekte
- Intersects: Diese Relation beschreibt sich schneidende Objekte. Sonderfälle dieser Beziehung können durch Ableitungen dieser Klasse unterschieden werden.
  - objects: die sich schneidenden Objekte
- Equals: Diese Relation beschreibt zwei deckungsgleiche Objekte
- Touches: Diese Relation beschreibt zwei sich berührende Objekte
- Overlaps: Diese Relation beschreibt zwei Objekte, die sich überlappen, bei denen aber weder eines der Objekte im anderen enthalten ist, noch eine Deckungsgleichheit oder Berührung vorliegt.
- Contains: Diese Relation beschreibt zwei Objekte, bei denen das eine im anderen enthalten ist.
  - container: Das Objekt, das das andere enthält
  - contained: Das Objekte, das im ersten enthalten ist
  - objects (*ererbt*): Um die Kompatibilität zur (Super-)Relation „Intersects“ zu wahren, enthält dieses Attribut genau die Referenzen der Attribute „container“ und „contained“. Somit muss nur die „Contains“-Relation modelliert werden und nicht sowohl eine „Contains“- als auch eine „Intersects“-Relation.
- Covers: Das erste Objekt enthält das zweite, und das zweite Objekt berührt das erste am Rand.
- ProperContains: Das zweite Objekt ist im ersten enthalten ohne dessen Begrenzung zu berühren.

### 3.3.3 Modellierungsbeispiel

Im folgenden Beispiel wird die Topologie von Flurstücken (engl. Lot) beispielhaft modelliert. Auf einem dieser Flurstücke steht ein Gebäude, dessen Grundriss durch eine Fläche modelliert wird (siehe Abb. 24).

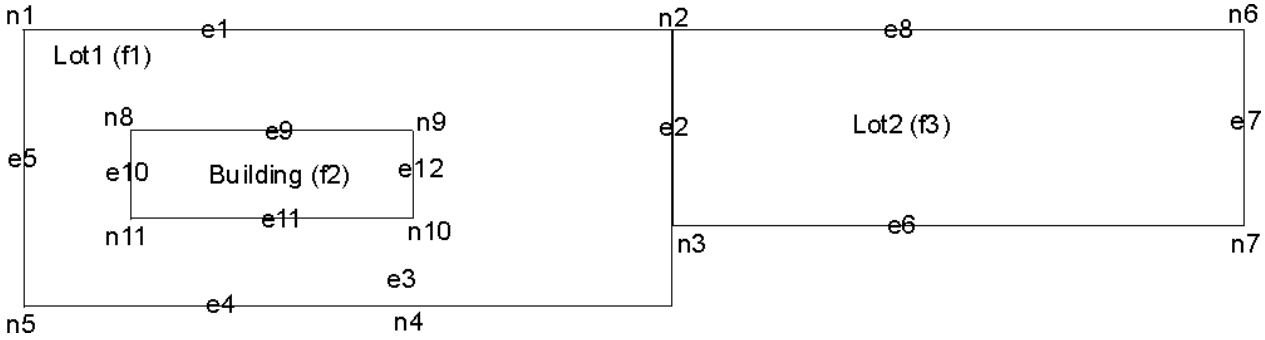


**Abb. 24: Flurstücke**

Betrachtet man die Flurstücke und den Gebäudegrundriss als Flächen f1—f3, die durch Kanten e1—e12 begrenzt werden, die ihrerseits wieder durch entsprechende Knoten n1—n11 begrenzt sind, dann lässt sich dieses Beispiel wie folgt durch die Verwendung der topologischen Objekte aus Abb. 24 modellieren:

```
<awml:awml>
  <awml:nexusobject>
    <nsas:type><nsas:value>Node</nsas:value></nsas:type>
    <nsas:nol><nsas:value>...n1NOL...</nsas:value></nsas:nol>
    <nsas:pos><nsas:value>...n1Pos...</nsas:value></nsas:pos>
    <nsas:dimension><nsas:value>0</nsas:value></nsas:dimension>
  </awml:nexusobject>
  ...Nodes n2-n11...
  <awml:nexusobject>
    <nsas:type><nsas:value>Edge</nsas:value></nsas:type>
    <nsas:nol><nsas:value>...e1NOL...</nsas:value></nsas:nol>
    <nsas:pos><nsas:value>...e1Pos...</nsas:value></nsas:pos>
    <nsas:extent>
        <nsas:value>...e1Extent...</nsas:value>
    </nsas:extent>
    <nsas:dimension><nsas:value>1</nsas:value></nsas:dimension>
  </awml:nexusobject>
  ...Edges e2-e12...
  <awml:nexusobject>
    <nsas:type><nsas:value>Face</nsas:value></nsas:type>
    <nsas:nol><nsas:value>...f1NOL...</nsas:value></nsas:nol>
    <nsas:pos><nsas:value>...f1Pos...</nsas:value></nsas:pos>
    <nsas:extent>
      <nsas:value>...f1Extent...</nsas:value>
    </nsas:extent>
    <nsas:dimension><nsas:value>2</nsas:value></nsas:dimension>
  </awml:nexusobject>
  ...Faces f2-f3...
```

```
<!-- Topologische Relationen zwischen den topologischen
     Objekten nach OGC -->
<awml:nexusobject>
  <nsas:type><nsas:value>Boundary</nsas:value></nsas:type>
  <nsas:nol>
    <nsas:value>...Boundary1NOL...</nsas:value>
  </nsas:nol>
  <nsas:bounded>
    <nsas:value>...e1NOL...</nsas:value>
  </nsas:bounded>
  <nsas:boundary>
    <nsas:value>...n1NOL...</nsas:value>
  </nsas:boundary>
</awml:nexusobject>
<awml:nexusobject>
  <nsas:type><nsas:value>Boundary</nsas:value></nsas:type>
  <nsas:nol>
    <nsas:value>...Boundary2NOL...</nsas:value>
  </nsas:nol>
  <nsas:bounded>
    <nsas:value>...e1NOL...</nsas:value>
  </nsas:bounded>
  <nsas:boundary>
    <nsas:value>...n2NOL...</nsas:value>
  </nsas:boundary>
</awml:nexusobject>
...Boundaries der Edges e2–e12...
<awml:nexusobject>
  <nsas:type><nsas:value>Boundary</nsas:value></nsas:type>
  <nsas:nol>
    <nsas:value>...Boundary3NOL...</nsas:value>
  </nsas:nol>
  <nsas:bounded>
    <nsas:value>...f1NOL...</nsas:value>
  </nsas:bounded>
  <nsas:boundary>
    <nsas:value>...e1NOL...</nsas:value>
  </nsas:boundary>
</awml:nexusobject>
...weitere Boundaries der Face 1 und der Faces f2–f3...
<awml:nexusobject>
  <nsas:type><nsas:value>Interior</nsas:value></nsas:type>
  <nsas:nol>
    <nsas:value>...Interior1NOL...</nsas:value>
  </nsas:nol>
```

```
    <nsas:surrounding>
      <nsas:value>...f1NOL...</nsas:value>
    </nsas:surrounding
    <nsas:interior>
      <nsas:value>...f2NOL...</nsas:value>
    </nsas:interior>
  </awml:nexusobject>
</awml:awml>
```

Alternativ können die Parzellen Lot 1 und Lot 2 durch entsprechende Lot-Objekte und das Gebäude durch ein Building-Objekt modelliert werden. Unter Verwendung der in Abb. 23 dargestellten topologischen Relationen ergibt sich folgendes Modell:

```
<awml:awml>
  <awml:nexusobject>
    <nsas:type><nsas:value>Lot</nsas:value></nsas:type>
    <nsas:nol><nsas:value>...Lot1NOL...</nsas:value></nsas:nol>
    <nsas:pos><nsas:value>...Lot1Pos...</nsas:value></nsas:pos>
    <nsas:extent>
      <nsas:value>...Lot1Extent...</nsas:value>
    </nsas:extent>
  </awml:nexusobject>
  ...Lot2 und Building...

  <!-- Topologische Beziehungen zwischen den Objekten -->
  <awml:nexusobject>
    <nsas:type>
      <nsas:value>Touches</nsas:value>
    </nsas:type>
    <nsas:nol>
      <nsas:value>...Touches1NOL...</nsas:value>
    </nsas:nol>
    <nsas:objects>
      <nsas:value>Lot1NOL,Lot3NOL</nsas:value>
    </nsas:objects>
  </awml:nexusobject>
  ...weitere Touches-Relationen...
  <awml:nexusobject>
    <nsas:type>
      <nsas:value>ProperContains</nsas:value>
    </nsas:type>
    <nsas:nol>
      <nsas:value>...Contains1NOL...</nsas:value>
    </nsas:nol>
```

```
    <nsas:container>
      <nsas:value>...Lot1NOL...</nsas:value>
    </nsas:container>
    <nsas:contained>
      <nsas:value>...BuildingNOL...</nsas:value>
    </nsas:contained>
  </awml:nexusobject>
</awml:awml>
```

# 4 Zusammenfassung

Der hier vorgelegte Bericht der Taskforce Topologie beschreibt zunächst das Konzept zur Verwendung von Relationen in Nexus. Es werden die verschiedenen Eigenschaften von Relationen und ein konkreter Ansatz zu deren Verwaltung in Nexus präsentiert. Schließlich wird eine Klassenhierarchie für verschiedene Teilbereiche wie z.B. thematische oder zeitliche Relationen vorgestellt und erläutert. Das hier entwickelte Konzept zur Behandlung von Relationen in Nexus macht zwei Infrastrukturerweiterungen erforderlich: Zum einen müssen Mengen abgebildet werden können, zum anderen wird es notwendig, dass Context Server einen schreibenden Zugriff erlauben, d.h. es muss ermöglicht werden, Referenzen auf Relationsobjekte innerhalb eines Context Server Objektes von außen speichern bzw. ändern zu dürfen.

Aufbauend auf den Ausführungen zu Relationen im Allgemeinen wird im zweiten Teil des Technischen Berichts das Topologie-Konzept für die Nexus-Plattform erarbeitet. Am Anfang steht hier eine Zusammenstellung verwandter Arbeiten aus verschiedenen relevanten Forschungsbereichen zum Thema Topologie. Auf der Grundlage einer Diskussion der verschiedenen Ansätze wird die Nexus-spezifische Umsetzung erläutert und anhand eines Beispiels illustriert.

# 5 Glossar

| | |
|---|---|
| AWM: | Augmented World Model |
| AWML: | Augmented World Modeling Language |
| AWQL: | Augmented World Query Language |
| AWS: | Augmented World Schema |
| NOL: | Nexus Object Locator |
| OGC: | OpenGIS Consortium |

# 6 Literatur


Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26, No. 11, Nov. 1983, pp. 832-843.

Clementini, E. & Di Felice, P., 1996: An Algebraic Model for Spatial Objects with Indeterminate Boundaries. In: P. A. Burrough and A. U. Frank, Eds.: Geographic Objects with Indeterminate Boundaries. GISDATA Series vol. chapter 11. London: Taylor & Francis, 1996, pp. 155-169.,

Dürr F., Rothermel K., 2003, On a Location Model for Fine-Grained Geocast, In Proceedings of the Fifth International Conference on Ubiquitous Computing (UbiComp 2003), Seattle, WA, USA, pp. 18-35

Egenhofer, Max J. and Herring, J., 1991, Categorizing binary topological relationships between regions, lines, and points in geographic databases, Technical Report, Department of Surveying Engineering, University of Maine, Orono.

Jiang C., Steenkiste P., 2002, A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing, In Proceedings of the Fourth International Conference on Ubiquitous Computing (UbiComp 2002), Göteborg, Sweden, pp. 246-263

OpenGIS Consortium, 1999, OGC Abstract Specification, Topic 8: Relationships between Features, Version 4, Project Document Number: 99-108r2, Datum: 1999-03-26. http://www.opengis.org/techno/abstract/99-108r2.pdf

OpenGIS Consortium, 2003, Geography Markup Language (GML) Implementation Specification, Document Reference Number OGC 02-023r4, Datum: 2003-01-29.

Randell D.A., Cohn A. G., 1989, Modelling Topological and Metrical Properties in Physical Processes, In Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning, Los Altos, pp. 55-66.