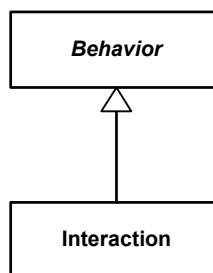


Die folgenden Abschnitte behandeln die Metamodelle zu Notationselementen aus Kapitel 15 (Sequenzdiagramm). Diese Abschnitte sind für die Zertifizierung zum „OMG Certified UML Professional (Fundamental)“ wichtig. Zum schnelleren Auffinden haben wir die Seitenzahlen des dazugehörigen Abschnitts in der 3. Auflage von „UML 2 glasklar“ hinzugefügt (in der 2. Auflage sind diese Abschnitte im Buch abgedruckt)

Interaktion (S.414)

Metamodell



Metamodell `Interaction`

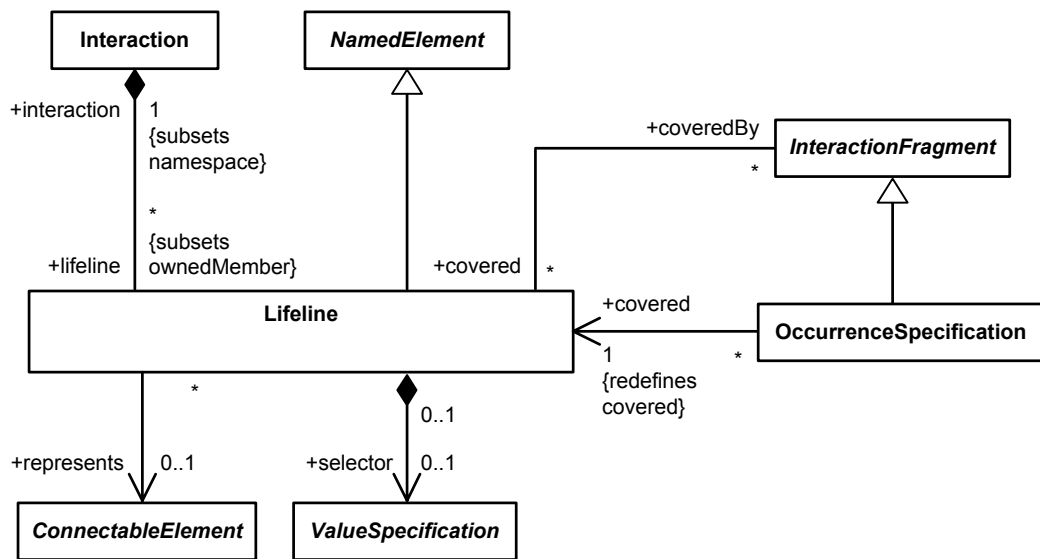
Eine `Interaction` ist ein `Behavior` aus den `CommonBehaviors` Paket des UML Metamodells.

Lebenslinie und Ereignisse (S.417)

Metamodell

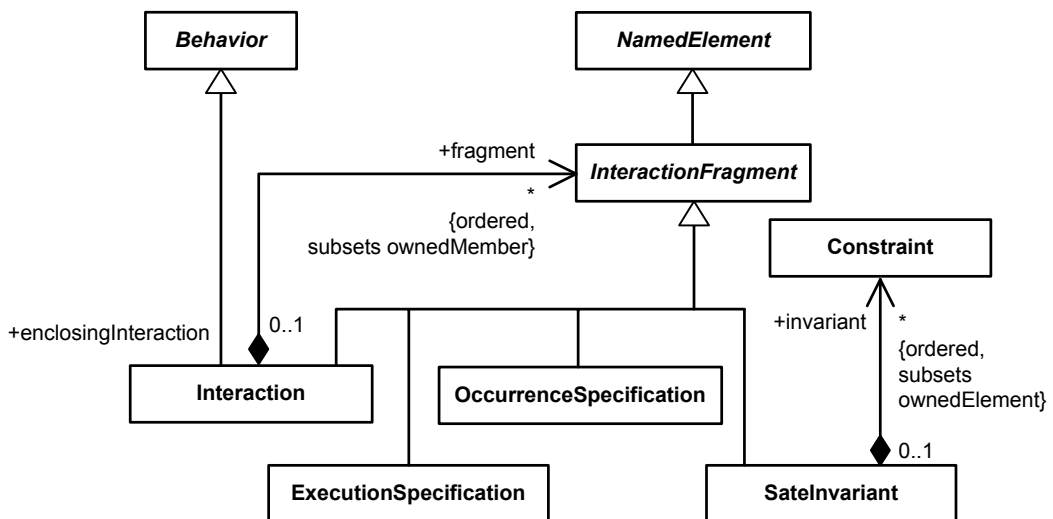
Die Klasse `Lifeline` ist eine Unterklasse der abstrakten Klasse `NamedElement` und stellt damit zunächst nur eine benannte Box dar. Besitzer der Lebenslinie und von dort aus mit Namen ansprechbar ist die umschließende Interaktion (`+interaction / +lifeline`).

Falls die Lebenslinie nicht nur einen beliebigen Kommunikationspartner sondern ein verbindbares Element (`ConnectableElement`) darstellt, existiert die `+represents`-Assoziation. Der Selektor bei mehrwertigen Attributen wird durch eine Komposition (`+selector`) zu einer `ValueSpecification` abgebildet.



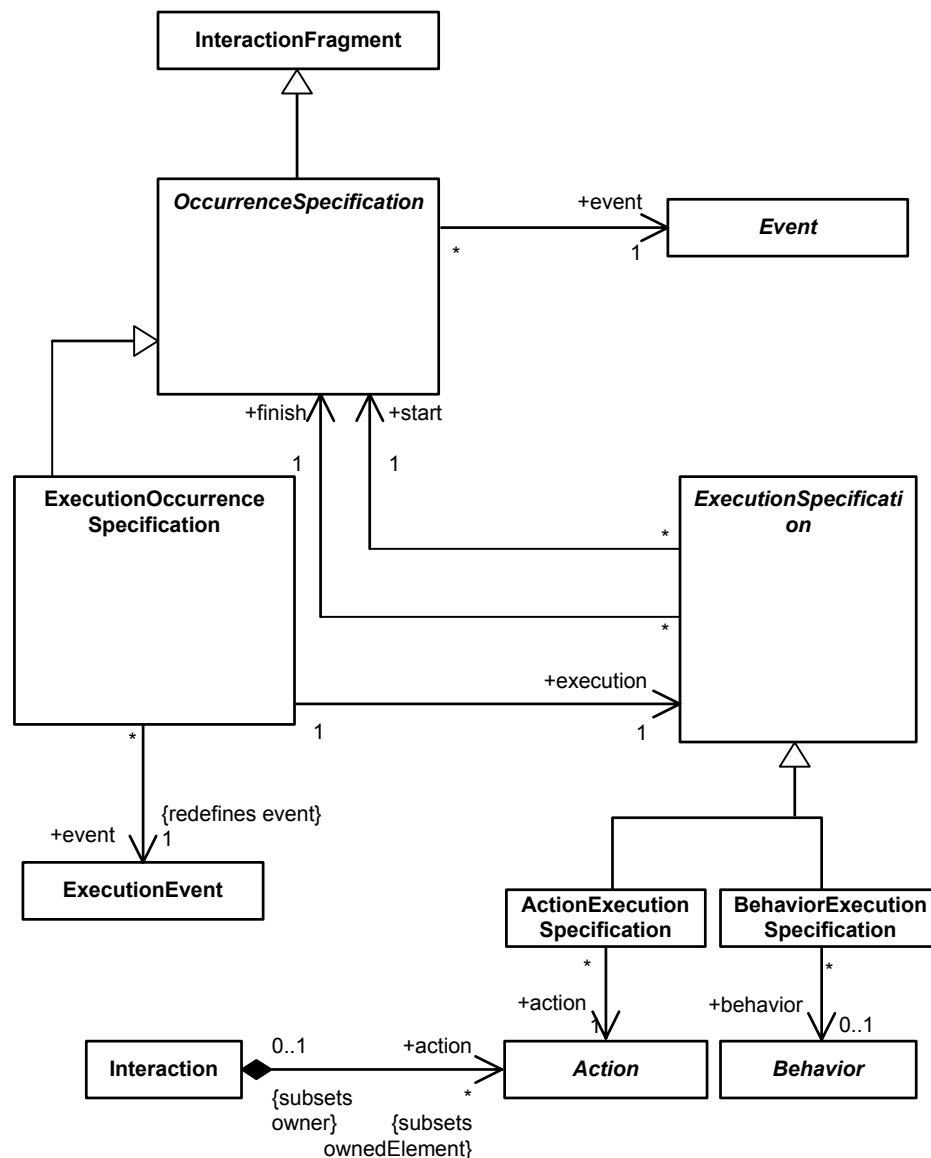
Metamodell Lifeline

Interessant ist nun die Abbildung der Ereignisse, die auf einer Lebenslinie stattfinden können. Derartige Ereignisauftritte werden im Modell als `OccurrenceSpecification` veranschaulicht. Wie wir im weiteren Verlauf des Kapitels noch sehen werden, liegen nicht nur Ereignisse und Ausführungssequenzen auf einer Lebenslinie, sondern auch z.B. Zustände oder Referenzen auf andere Interaktionen. Daher wurde ein Sammelbegriff (abstrakte Oberklasse) für derartige Interaktionsbestandteile oder -fragmente eingeführt: `InteractionFragment`



Metamodelleinbettung InteractionFragment

Beachten Sie dabei, dass die Interaktion an sich selbst wieder ein `InteractionFragment` ist. Hier wurde das Composite Pattern (siehe [GOF01]) genutzt. Jedes `InteractionFragment` ist zudem ein benanntes Element (`NamedElement`).



Metamodell Ereignisauftritte

Eine *OccurrenceSpecification* ist eindeutig mit ihrem Typ einem *Event* (+event) verbunden. Für Ausführungssequenzen (*ExecutionSpecification*) wurden spezielle Ereignisauftrittsbeschreibungen eingeführt (*ExecutionOccurrenceSpecification*), die Start (+start) und Ende (+finish) der Sequenz markieren und deren zugeordnete Events von *Event* auf *ExecutionEvents* ({redefines event}) beschränkt wurden.

Der Auftritt eines durch eine *ExecutionOccurrenceSpecification* beschriebenen Ereignisses kann dann zur Ausführung (+execution) einer Aktion (*Action*) oder allgemein eines Verhaltens führen. Diese müssen dann durch spezielle Ausführungssequenzen (*ActionExecutionSpecification* / *BehaviorExecutionSpecification*) (im Modell durch Balken oder Kästen dargestellt) beschrieben werden.

Hier erfolgt nun der Abstraktionsschritt zwischen `OccurrenceSpecification`, speziellen `MessageOccurrenceSpecifications` für Nachrichten und den Eventtypen (hier `MessageEvent`). `MessageOccurrenceSpecifications` repräsentieren den Auftritt von Ereignissen im Zusammenhang mit Nachrichten, d.h. Ereignisse zum Senden/Empfangen von Signalen und zum Rufen/Empfangen von Operationen. `MessageOccurrenceSpecification` unterscheidet sich von `OccurrenceSpecification` dadurch, dass sie nur spezielle Ereignisauftritte (vom Typ `MessageEvent` oder davon abgeleiteten Typen) repräsentieren (+event {redefines event}).

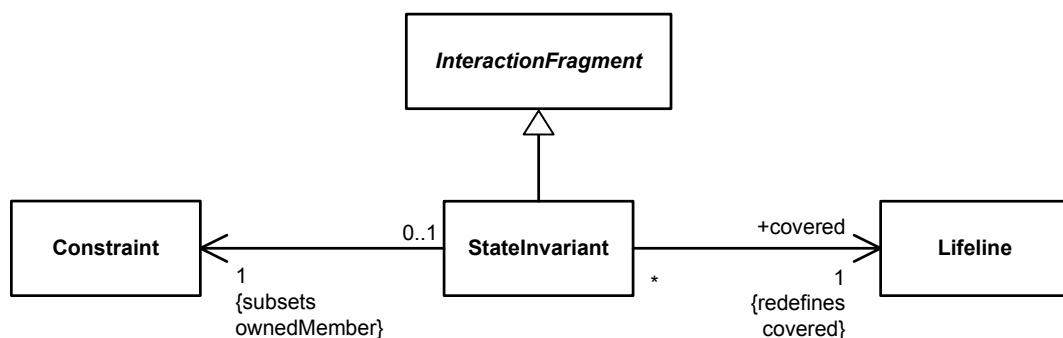
Der Name inklusive Parameter einer Nachricht (Signatur) wurde durch eine Assoziation (+signature) zu einem `NamedElement` modelliert. Er erschließt sich aber auch aus den Beziehungen der `OccurrenceSpecification`. Es existieren Beziehungen zu den gerufenen Aktionen oder Operationen. Da der Name der Operation/Aktion gleich dem Namen der Nachricht sein muss, leitet sich somit die Beziehung (+signature) ab. Die Beziehung zur `ValueSpecification` repräsentiert die Aktualparameter, die den Formalparametern der Nachricht (definiert durch die Signatur) zugewiesen werden.

Falls die Nachricht über einen speziellen Konnektor verschickt wird, kann über die Beziehung +connector eine Verbindung geschlossen werden. Im Modell müssen Sie dazu das Kompositionsstrukturdiagramm heranziehen.

Wie Sie bei genauem Hinsehen feststellen, sind Erzeugungs- bzw. Antwortnachrichten nicht in das Metamodell integriert. Mit viel gutem Willen lässt sich der Bezug herstellen. Die UML geht davon aus, dass jede Nachricht durch eine Aktion erzeugt wird. Ein Operationsaufruf durch eine `CallAction` (synchronous/asynchronous), eine Signalübermittlung durch eine `SendSignalAction` und eben eine Objekterzeugung durch eine `CreateObjectAction`. Während die beiden ersten noch durch `MessageSort` abgebildet werden, wurde dies bei der Erzeugung nicht übernommen.

Zustandsinvariante (S.435)

Metamodell

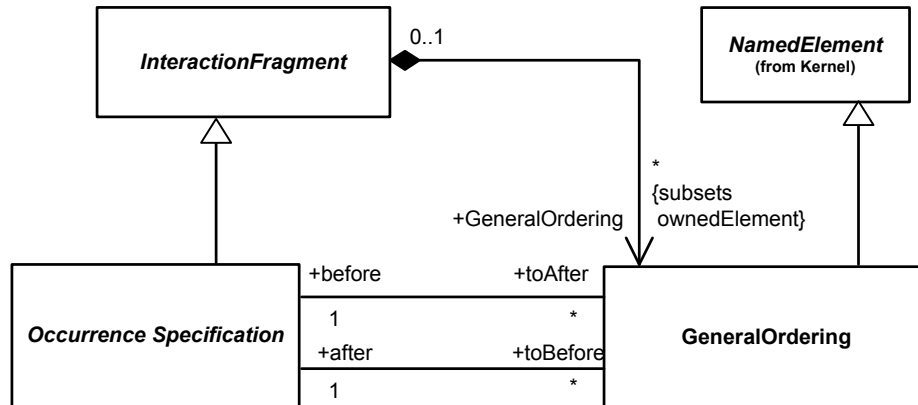


Metamodellausschnitt Klasse `StateInvariant`

Obige Abbildung zeigt die Einbettung der Zustandsinvariante als Klasse `StateInvariant` in das Metamodell der UML 2. Die eigentliche Bedingung wird als `Constraint` definiert. Die Zuordnung zur Lebenslinie (`Lifeline`) erfolgt analog der `OccurrenceSpecification` über ein `InteractionFragment`.

Ordnungsbeziehung (S.458)

Metamodell



Metamodellausschnitt Klasse GeneralOrdering

Obige Abbildung zeigt die Einbindung der Ordnungsbeziehung als Klasse `GeneralOrdering` in das Metamodell der UML 2. Wie bei der `OccurrenceSpecification` erläutert, wird auch die Ordnungsbeziehung als Teil einer Interaktion (`InteractionFragment`) „eingeklinkt“. Der Rest erschließt sich recht schnell: Fuß (`+before`) und Spitze (`+after`) der Ordnungsbeziehung werden über zwei Assoziationen modelliert. Ein Ereignis kann natürlich durch mehrere Ordnungsbeziehungen in die gesamte Ereigniskette „eingeordnet“ (`+toAfter *`, `+toBefore *`) werden. Um eine Ordnungsbeziehung aus der Interaktion (Namensraum) ansprechen zu können, muss sie zudem ein `NamedElement` repräsentieren.