Uit: Sander Hoogendoorn Pragmatisch modelleren met UML 2.0 ISBN 9043006521

Het domein van de applicatie

There is no safety in numbers, or in anything else.

James Thurber

Wat was er nu eigenlijk eerder: de kip of het ei? Er zijn projecten die starten met het beschrijven van de requirements. Daaruit volgen de klassen in het domein van de applicatie, zeg maar de business classes. Er zijn ook projecten die starten met het identificeren van de business classes en vervolgens de requirements ophangen aan deze klassen. De kip of het ei?
Ik prefereer het om allebei te doen. Ik start met het modelleren van de requirements en start tegelijk met het in kaart brengen van het domein van de applicatie. Al tijdens het modelleren van de requirements in procesdiagrammen en use cases komt er allerlei kennis over het domein aan de oppervlakte. Ik gebruik deze kennis om gaandeweg een eerste schets van de klassen in het domein te maken. Ik noem deze schets het business class diagram. In agile software development wordt gedurende het project meer en meer kennis aan het business class diagram toegevoegd.

Tip 139. Schets het domein van de applicatie gedurende het modelleren van de requirements.

Beide gezichtspunten convergeren naarmate de user interface wordt gemodelleerd, en in de samenwerking van de klassen in sequence diagrammen en interactiepatronen. Maar hoe krijgt de eerste schets van het domein van de applicatie vorm?

Tien voor taal

De eerste ruwe schets van het domein beschrijft de 'concepten' die waardevol zijn in dit domein. Deze concepten zijn relatief eenvoudig op te sporen. Ze komen voor in het dagelijks taalgebruik van de gebruikers. Er is een eenvoudige techniek om de concepten te vinden. Een eenvoudige taalkundige analyse is vaak al voldoende. Loop hiertoe alle zelfstandige naamwoorden langs die in de procesdiagrammen en use cases voorkomen.

Tip 140. Identificeer concepten uit het domein van de applicatie in de requirements.

Ik werp even een blik op de use cases van Dare2Date, zoals **Bewerken Profiel** en **Versturen Bericht**. Deze use cases beschrijven concepten als **Profiel** en **Bericht**. Het is niet toevallig dat ik de namen van deze concepten met hoofdletters noteer. Zo zijn ze makkelijk te herkennen. **Profiel** en **Bericht** maken zeker deel uit van het domein van de applicatie. En wellicht horen ook **Bezoeker**, **Creditcard** en **Voorkeuren** tot het domein. Dit puur op basis van de namen en beschrijvingen van de procesdiagrammen en de use cases. Als deze zijn geformuleerd in het taalgebruik van de gebruikers, zijn de concepten in het domein snel te vinden.

Concepten identificeren

Maak vervolgens een lijst van deze zelfstandige naamwoorden. Scan hiervoor de eigenschappen van de use cases zoals naam, doelstelling en stappenplan. Beschouw ieder van de hierin voorkomende zelfstandige naamwoorden als een potentieel concept in het domein. Voeg ook de actoren toe aan de lijst met kandidaten, zowel de primaire als de secundaire. Onderzoek ook de de procesdiagrammen op zelfstandige naamwoorden.

Tip 141. Identificeer zelfstandige naamwoorden in procesdiagrammen en use cases als potentieel concept in het domein. Beschouw ook actoren als potentieel concept.

Neem als voorbeeld use case **Versturen Bericht**.

Use case
Versturen Bericht

Preconditities
Profiel is bekend.
Actor is ingelogd.

Postcondities
Bericht is verstuurd aan Profiel of
Actor heeft geannuleerd.

Stappenplan
1. Haal Profiel op
2. Toon webpagina.
3. Actor typt nieuw Bericht in.
4. Actor bevestigt versturen Bericht.

5. Applicatie verstuurt Bericht aan Profiel.
6. Actor krijgt bevestiging van versturen Bericht.

Een quick scan van **Versturen Bericht** levert onderstaande concepten op.

Bericht
Profiel
Actor
Webpagina
Applicatie
Bevestiging

Niet alle zelfstandige naamwoorden in deze lijst zijn echter geschikt. **Actor**, **Bevestiging** en **Webpagina** liggen niet heel erg voor de hand. Wanneer is een potentieel concept nu eigenlijk geschikt als business class?

Van concept naar business class

Wanneer maakt een potentieel concept deel uit van het domein van een applicatie? Hier komt de referentiearchitectuur van pas. De potentiële concepten uit de vorige paragraaf horen thuis in verschillende lagen van de referentiearchitectuur. **Bericht** bevat zeer waarschijnlijk bedrijfslogica. **Bericht** maakt deel uit van de laag *Business*. **Webpagina** kent geen bedrijfslogica. **Webpagina** hoort in *Presentation*. **Bevestiging** heeft te maken met het tonen van boodschappen. Deze kandidaat is waarschijnlijk een utility. Om een lang verhaal kort te maken: alleen potentiële concepten die in de laag *Business* zijn te plaatsen zijn geschikt. Alleen deze concepten maken potentieel deel uit van het domein van de applicatie. Laat de andere voorlopig vervallen.

Tip 142. Beschouw alleen potentiële concepten die deel uitmaken van *Business*.

Bericht is zinvol. **Webpagina** niet. **Bevestiging** is wellicht een twijfelgeval. Bedenk in zo'n geval of een concept bedrijfslogica kent of niet. Maak er niet te veel woorden aan vuil. Een concept meer of minder is in deze vroege fase geen probleem. In het vervolg van het project vallen ongebruikte concepten vanzelf af. En naar alle waarschijnlijkheid duiken er ook nog nieuwe concepten op.
Actor lijkt geschikt. **Actor** is echter wel erg abstract. Vervang abstracties als **Actor** liever door de daadwerkelijk actor, in dit voorbeeld **Abonnee**. Use case **Versturen Bericht** levert zo de volgende concepten op.

Abonnee
Bericht
Profiel

De collectie geschikte concepten vormt de laag *Business* van de applicatie. Deze concepten verworden al snel tot de factories en business classes van de applicatie. Laat de naam van een klasse altijd met een hoofdletter beginnen. Noteer instanties van klassen met een kleine eerste letter. Dit houdt onderscheid tussen beide.

Tip 143. Definieer één business class per geschikt concept.

Zo zijn business classes als **Profiel**, **Bericht**, **Abonnee**, **Bezoeker** en de factory **Voorkeuren** al spoedig geïdentificeerd.

Spraakverwarring

Het is in projecten eerder regel dan uitzondering dat een concept onder diverse namen voorkomt. Andersom komt trouwens ook voor. Verschillende concepten die onder dezelfde naam worden geduid. Dit levert spraakverwarring in uw project op. Een projectrisico! Deze spraakverwarring neemt bovendien toe naarmate het domein complexer is. Vooral in domeinen waarin nauw verwante concepten voorkomen, vormt deze spraakverwarring een serieus probleem.

Leken
Ooit werkte ik aan een project met ruim honderd medewerkers. Brrr. Het domein van de applicatie – hypotheken – kende concepten als voorstel, offerte en contract. Omdat bij de gebruikers deze verschillen zeer vanzelfsprekend waren, duurde het ruim drie maanden voordat de ontwikkelaars überhaupt ontdekten dat er verschillen waren.

De spraakverwarring is voor een groot deel te voorkomen door de business classes te voorzien van een korte beschrijving. Liefst in één enkele zin. Stel deze beschrijvingen altijd samen met de gebruikers en in hun taalgebruik.

Tip 144. Voorzie alle business classes van een korte, eenduidige beschrijving.

Deze vingeroefening blijkt in de praktijk een goede graadmeter voor de kwaliteit van de gekozen business classes. Als gebruikers niet in staat zijn snel een zinvolle beschrijving te geven, is een kandidaat waarschijnlijk niet geschikt. Zo valt zo'n kandidaat alsnog door de mand.

My first business class diagram

Nu de business classes zijn geïdentificeerd is het allereerste business class diagram voor de applicatie te modelleren. Het business class diagram modelleert de structuur van het domein van de applicatie. In eerste instantie geeft het een globaal beeld van de factories en de busines classes. In de loop van een project

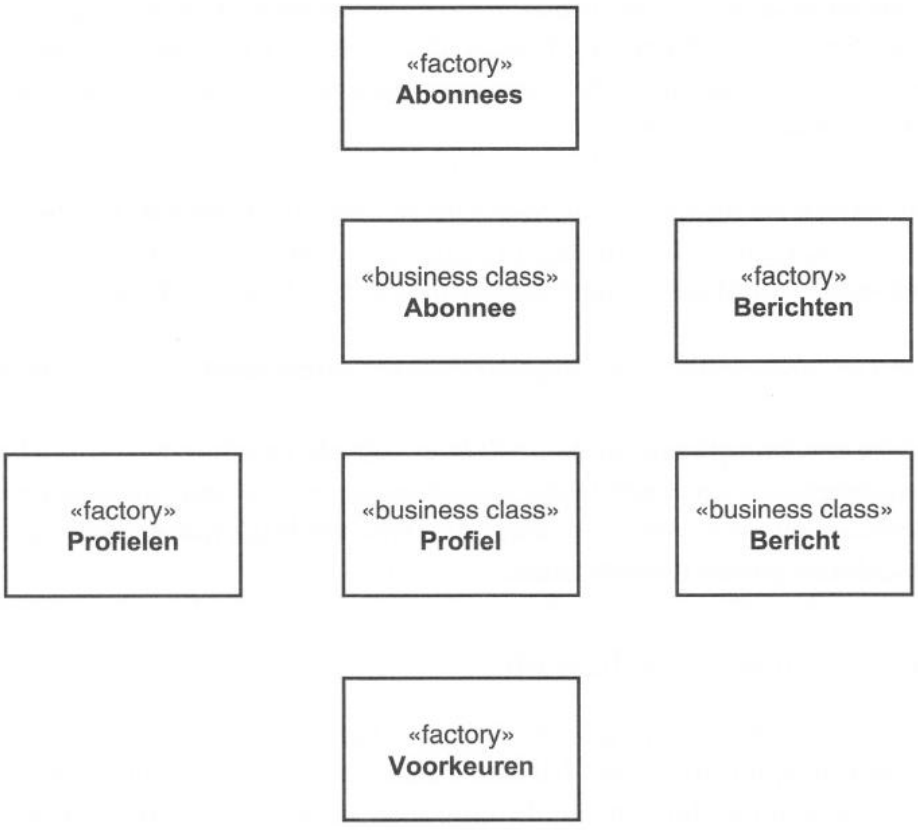
wordt dit aangevuld met hun methoden, attributen en de exacte relaties tussen de klassen.
Neem alle factories en business classes op in het business class diagram. Meervoud is een factory, enkelvoud een business class. Geef ze weer als rechthoeken. Plaats de naam van de klasse midden in deze rechthoek. Modelleer nog geen attributen of methoden.

Tip 145. Creëer het business class diagram. Geef alle factories en business classes een plaats in dit diagram.

Voorzie uw factories en business classes van de stereotypen **«factory»** en **«business class»**. Dit verduidelijkt het business class diagram, en geeft de verschillende verantwoordelijkheden van de klassen goed weer.

Tip 146. Voorzie factories en business classes van de stereotypen **«factory»** en **«business class»**.

Het eerste business class diagram voor Dare2Date is gemodelleerd in afbeelding 53. Het bevat de tot nu toe geïdentificeerde factories en business classes. Over de relaties tussen deze klassen gaat de komende paragraaf. Later in het project komen de gedetailleerde relaties aan de orde. Deze zijn beschreven in hoofdstuk *Relaties tussen klassen.*



afbeelding 53 – Eerste business class diagram

Dit eerste business class diagram is nog verre van indrukwekkend. Maar zoals gezegd, het groeit en bloeit.

Associaties

Zodra de eerste factories en business classes zijn onderscheiden, valt ontegenzeggelijk op dat er relaties bestaan tussen deze klassen. Een abonnee kent een profiel en een profiel beheert zijn berichten. En dus is er een relatie tussen **Abonnee** en **Profiel** en is er een relatie tussen **Profiel** en **Bericht**. Modelleer relaties tussen business classes in het business class diagram zodra ze onderkend worden. In afbeelding 54 zijn deze beide relaties reeds onderkend.

Tip 147. Modelleer relaties tussen business classes zodra ze ter sprake komen.

In paragraaf *Het smart factory pattern* in hoofdstuk *De kloof tussen ontwerp en code* is de relatie tussen factory en business classes aangestipt. Alhoewel ik de factories altijd opneem in het business class diagram, modelleer ik maar heel zelden de relaties tussen een factory en de bijbehorende business class. Een business class diagram raakt al snel verstopt met associaties en de relatie tussen factory en business class is toch evident. Ik modelleer factories bovendien vooral om hun methoden, niet om hun relaties.

Tip 148. Laat de relatie tussen een factory en business class achterwege.

In het business class diagram concentreer ik me zodoende op de onderlinge relaties van de business classes. Deze zijn bepalend voor de structuur van de applicatie. UML kent diverse typen onderlinge relaties tussen klassen, zoals associaties, dependencies en generalisaties.
De associatie is de meest algemene. Een associatie modelleert niet veel meer dan dat er een betekenisvolle, structurele relatie tussen klassen bestaat. Of liever gezegd: tussen instanties van klassen. In de praktijk betekent dit dat er een associatie tussen twee klassen geldt, als een instantie van tenminste één van deze klassen instanties van de andere klasse kent. Tenzij over een relatie hele specifieke kennis boven tafel komt, volstaat het in deze fase van een project om relaties tussen klassen als associaties te modelleren. Wees zuinig met het toekennen van associaties. Bij het uitwerken van associaties in code, wordt voor iedere associatie code ontwikkeld.

Tip 149. Modelleer een associatie tussen twee business classes als instanties van de ene business class instanties van de andere business class kennen.

Een associatie is in UML weergegeven als een lijn tussen de betrokken klassen. Dit lukt zelfs op een whiteboard. Voorzie de associatie van een zinvolle naam die de relatie afdoende beschrijft. Doe dit alleen als niet direct duidelijk is wat

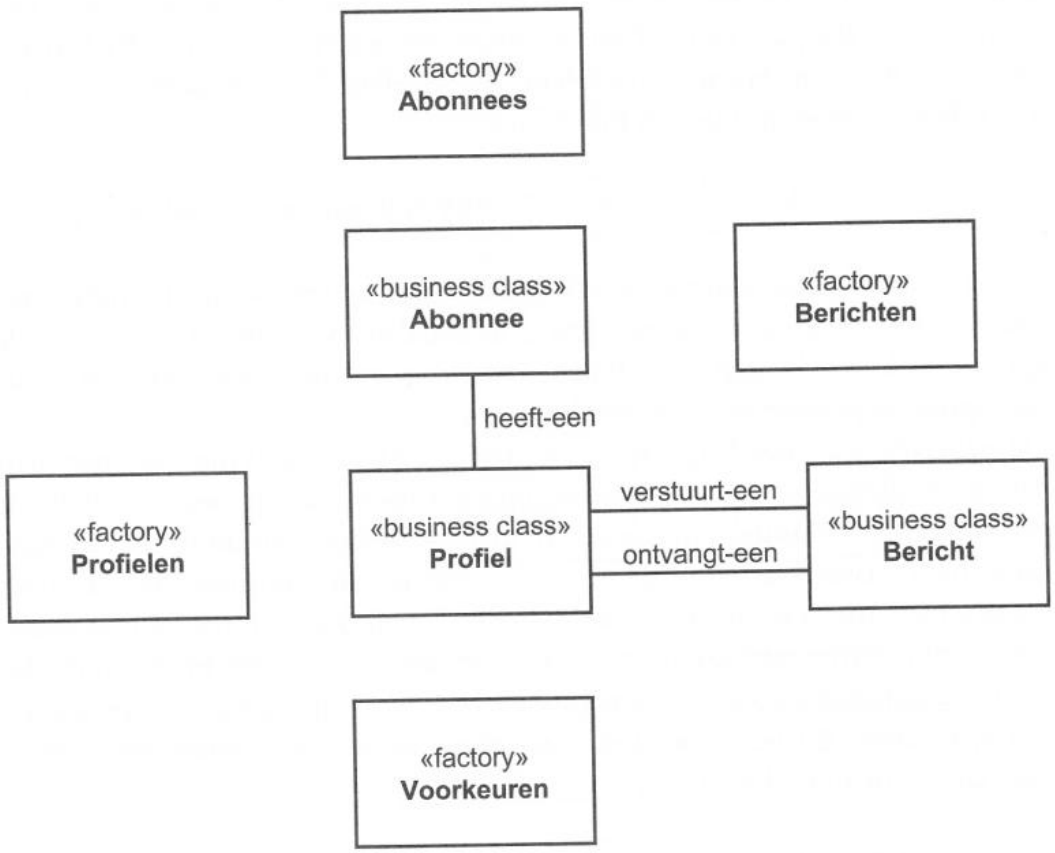
de associatie doet. Noteer deze naam langs de lijn, midden tussen de betrokken klassen. De naam van een associatie is te lezen volgens onderstaande notatie.

klasse + associatie + klasse

De naam van een associatie wordt gevat in een werkwoordszin: een werkwoord in onvoltooid tegenwoordige tijd, plus eventuele voor- en bijvoegsels. Misschien is de term werkwoordszin taalkundig niet heel fraai, maar het geeft de strekking aardig weer.

Tip 150. Voorzie een associatie van een naam als de aard van de associatie niet duidelijk is. Noteer deze naam in een werkwoordzin.

Een **Abonnee** heeft een **Profiel**. De bijbehorende associatie heet dan ook **heeft-een**. Associaties krijgen vaak namen als **is-onderdeel-van**, **woont-samen-met**, **bestaat-uit** of **is-verbonden-met**. Het business class diagram voor Dare2Date ziet er nu uit als in afbeelding 54.



afbeelding 54 – Business class diagram met relaties

Er is in dit diagram te lezen dat **Abonnee heeft-een Profiel**, **Profiel verstuurt-een Bericht** en **Profiel ontvangt-een Bericht**. Althans, dat laatste hoopt hij. Behalve een naam kent een associatie nog diverse andere eigenschappen, zoals de multipliciteit, de rollen van de betrokken klassen en de navigeerbaarheid van de associatie. In vroege fasen van project volstaat het business class diagram van afbeelding 54. Het is te vroeg om alle betekenisvolle structurele relaties te onderzoeken. Er is daarvoor nog onvoldoende kennis aanwezig. Het vinden van de relaties kost daardoor onevenredig veel tijd. Later in het project, als meer kennis is opgedaan, zijn de onderlinge relaties van de business classes veel makkelijker te modelleren. Noteer steeds alleen die eigenschappen van associaties die onderkend worden.

Relaties tussen klassen

Ik heb een hekel aan generaliseren. Surinamers doen dat ook altijd zo.

Theo Maassen

Nu de individuele klassen zijn vormgegeven, is het moment aangebroken om ook de relaties tussen klassen in het business class diagram te beschouwen. Gedurende de iteraties in een systeemontwikkelproject krijgen deze relaties steeds meer vorm. Er wordt immers meer en meer bekend over het domein van de applicatie. UML definieert diverse soorten relaties tussen klassen. James Rumbaugh noemt al dertien verschillende typen dependencies [Rumbaugh-99]. De makkelijkste en bekendste van deze relaties is de associatie. Deze is al eerder gebruikt om de relaties in het business class diagram initieel vast te leggen. Maar wanneer is het zinvol zo'n associatie te modelleren? En wanneer de andere relaties als een compositie of een generalisatie? Mijn strategie is: start alleen met associaties en vervang deze in de loop van het project door specifiekere relaties. Dit hoofdstuk zet de belangrijkste op een rijtje.

Associaties

Iedere relatie tussen twee of meer klassen, maar ook tussen instanties van dezelfde klasse heet in UML een *associatie*. Wanneer er een associatie geldt tussen klassen, is er een verbinding mogelijk tussen instanties van de betrokken klassen. Zo'n verbinding tussen twee instanties heet trouwens een *link*. Een voorbeeld. Er is een associatie tussen de business classes **Abonnee** en **Profiel**. Iedere abonnee heeft ten hoogste één profiel. Ieder profiel hoort bij exact één abonnee. Tussen instanties van **Abonnee** en **Profiel** bestaan links.
Een associatie geeft een structurele relatie aan tussen klassen. Een instantie van de ene klasse houdt kennis vast over nul, een of meer instanties van de andere klassen. In runtime worden tussen instanties voortdurend links aangemaakt en ook weer opgeheven. Associaties zijn de lijm van de applicatie. Modelleer een associatie tussen twee of meer klassen als er zo'n structurele relatie tussen de klassen bestaat. Geef een associatie tussen twee klassen weer door een lijn tussen beide betrokken klassen te trekken. Een associatie tussen twee klassen heet een *binaire associatie*. Geef een associatie tussen meer dan twee klassen weer als een diamant. Trek een lijn van elke betrokken klasse naar de diamant. Een associatie tussen drie klassen heet een *ternaire associatie*. Een associatie tussen n klassen heet een *n-aire associatie*.

Tip 261. Modelleer een associatie tussen klassen als tenminste één van de klassen kennis over de anderen vasthoudt.

Een aantal eigenschappen zijn van belang in een associatie:

- *Naam.* Iedere associatie heeft een naam.
- *Rollen.* De betrokken klassen vervullen rollen in de associatie.
- *Multipliciteit.* Een associatie kent multipliciteit, analoog aan een gegevensdiagram.
- *Navigeerbaar.* En tenslotte zijn associaties navigeerbaar. Soms in één, soms in allebei de richtingen.

De komende paragrafen beschrijven deze eigenschappen voor (binaire) associaties.

Namen van associaties

Iedere associatie heeft een naam. Geef een associatie een zodanige naam dat onderstaande notatie als natuurlijke taal te lezen is.

klasse + associatie + klasse

De associatie in deze notatie bestaat vrijwel altijd uit een werkwoord gevolgd door een of meer bijvoegsels. Dit noem ik een *werkwoordszin.* Beschrijf zo'n werkwoordszin in onvoltooid tegenwoordige tijd. Taalkundig misschien niet fraai, maar wel correct. Het geheel vormt zo een leesbare zin, in telegramstijl. Om met Alistair Cockburn te spreken: "Who beats who?"
Het is gebruikelijk de naam van een associatie te beschrijven zonder hoofdletters, met minstreepjes tussen de gebruikte woorden. Centreer de naam van een associatie tussen de betrokken klassen, op of onder de lijn of bij de diamant.

Tip 262. Beschrijf de naam van een associatie in een werkwoordszin, zonder hoofdletters en met streepjes tussen de woorden.

Een **Abonnee** heeft een **Profiel**. Als associatie geldt nu **heeft-een**. Associaties worden wel benoemd als **is-gelieerd-aan**, **woont-in** of **doceert-aan**. De binaire associatie tussen **Abonnee** en **Profiel** is gemodelleerd in afbeelding 104.
De naam van een associatie speelt een rol in het identificeren van associaties. Soms organiseer ik hiervoor een workshop, waaraan ook gebruikers deelnemen. Gebruikers kunnen vaak goed aangeven hoe de concepten uit het domein van de applicatie samenhangen. Druk naast de namen van klassen, ook de namen van associaties uit in de taal van de gebruikers.

afbeelding 104 – Associatie

Tip 263. Beschrijf namen van associaties in gebruikerstaal.

Er is een stilzwijgende afspraak dat een associatie van links naar rechts wordt gelezen. In afbeelding 104 is zo **Abonnee heeft-een Profiel** te lezen. Modelleer de klassen in het business class diagram bij voorkeur zo dat de associaties inderdaad van links naar rechts te lezen zijn.

Tip 264. Modelleer associaties van links naar rechts in een klassendiagram.

Plaats als het niet anders kan de associatie van boven naar beneden. Plaats in bovenstaand voorbeeld dan **Abonnee** boven **Profiel**. Soms is een associatie niet prettig leesbaar te modelleren, maar alleen van rechts naar links, of erger nog, alleen van onder naar boven. Om te voorkomen dat dergelijke associaties verkeerd geïnterpreteerd worden, voorziet UML in een *naampijl* (name arrow). Dit is een dichte driehoek die naast de naam van de associatie wordt geplaatst, en met een punt die in de richting wijst waarin de associatie gelezen moet worden.

Tip 265. Als een associatie niet van links naar rechts, of van onder naar boven leest, plaats dan een naampijl in de gewenste leesrichting.

Veronderstel om wat voor reden dan ook dat **Profiel** links van **Abonee** is afgebeeld, dan is de binaire associatie **heeft-een** te modelleren als in afbeelding 105.

afbeelding 105 – Associatie met naampijl

De associatie in afbeelding 105 leest nu alsnog **Abonnee heeft-een Profiel**. Zonder naampijl had hier nu **Profiel heeft-een Abonnee** gestaan. Maar het plaatsen van **Profiel** onder **Abonnee** was een beter alternatief geweest.

Alhoewel de naam van een associatie belangrijke kennis huisvest, neemt het opnemen van alle namen van associaties veel ruimte in beslag in het overbevolkte business class diagram. Laat, om ruimte te besparen, namen van associaties zoveel mogelijk weg. Dit kan als de associatie tussen de betrokken klassen evident en eenduidig is.

Tip 266. Laat namen van evidente associaties zoveel mogelijk weg.

De relatie tussen **Abonnee** en **Profiel** is evident en eenduidig. Laat **heeft-een** gerust weg.

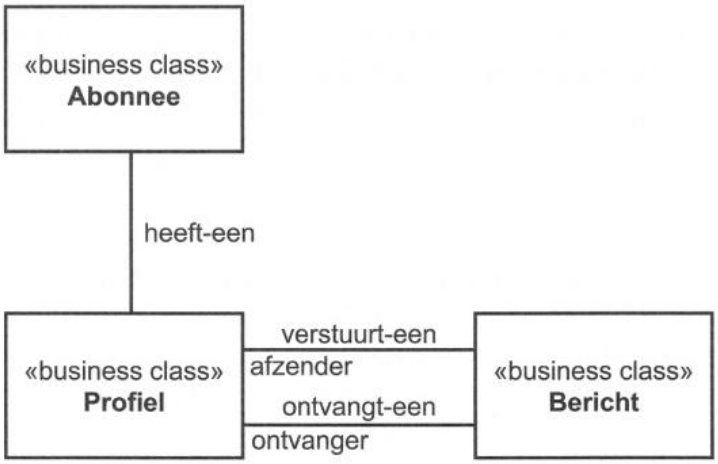
Rollen in associaties

Iedere klasse heeft een specifieke rol in een associatie. In verschillende associaties speelt een klasse verschillende rollen. De rol van een klasse kan in UML worden beschreven bij de associatie. Meestal beschrijft de naam van de betrokken klasse de rol al afdoende, zoals in **Abonnee heeft-een Profiel**. Als een klasse echter een rol speelt die niet overeenkomstig zijn naam is, beschrijf deze rol dan expliciet bij de associatie. Neem de rol bij de klasse op, aan het uiteinde van de associatie. Beschrijf een rol als een zelfstandig naamwoord, in termen van het domein van het project.

Tip 267. Beschrijf de rol van een klasse in een associatie als de naam van de klasse deze rol onvoldoende benoemt.

Zoals gezegd, in de associatie tussen **Abonnee** en **Profiel** zijn de rollen duidelijk. Tussen **Profiel** en **Bericht** gelden echter twee associaties. Een profiel ontvangt een bericht en een profiel verstuurt een bericht. **Profiel** vervult in deze associaties twee verschillende rollen, die van **afzender** en die van **ontvanger**. Benoem deze rollen in het business class diagram zoals in afbeelding 106.
Plaats, om ruimte te besparen, de naam van de associatie boven de lijn en de rollen van de klassen eronder. Een andere mogelijkheid is om nooit naam en rollen tegelijk te modelleren. Eén van beide is in de meeste gevallen afdoende specifiek.

Tip 268. Modelleer niet tegelijk de naam van een associatie én de rollen van de betrokken klassen.



afbeelding 106 – Rollen in associaties

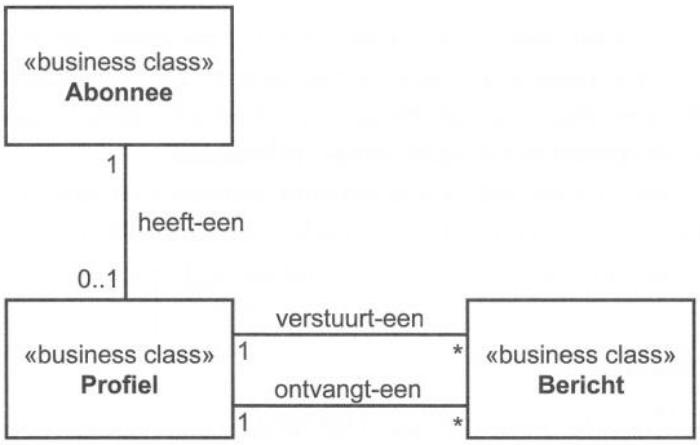
Multipliciteit

De multipliciteit van een associatie geeft aan hoeveel instanties van een klasse tegelijk een relatie kunnen hebben met één instantie van de andere klasse. Met de nadruk op tegelijk. Multipliciteit modelleert dus niet hoeveel instanties er gelinkt zijn tijdens de levensduur van een instantie, maar hoeveel er op hetzelfde moment gelieerd *kunnen* zijn.
In een associatie is van beide rollen de multipliciteit aan te geven. Dit gebeurt middels de bekende expressies, zoals *, **0..1**, **0..*** en **1..***. Zie hiervoor paragraaf *Multipliciteit* in hoofdstuk *Het user interface diagram*. Soms levert de multipliciteit in associaties verrassende resultaten op. Vrijwel altijd zijn deze te wijten aan bedrijfsregels, die het snelst aan de oppervlakte komen in een workshop met gebruikers. Zo heeft een abonnee ten hoogste één profiel. Da's waar ook! Als een abonnee zich net heeft aangemeld, is er nog geen profiel. Een abonnee verstuurt en ontvangt daarnaast nul, één of meer berichten. De multipliciteit van een klasse in een associatie is weergegeven aan het eigen uiteinde van de associatie.

Tip 269. Onderzoek de multipliciteit in een associatie zodra deze wordt gemodelleerd.

In afbeelding 107 is de multipliciteit van de associaties tussen **Abonnee**, **Profiel** en **Bericht** gemodelleerd.
Zodra bekend is welke multipliciteit geldt, modelleer ik deze. Ook als de multipliciteit exact één is. Hiermee is in een oogopslag te zien welke multipliciteit geldt in het diagram, en waar deze nog niet is gedefinieerd.

Tip 270. Modelleer de multipliciteit in een associatie expliciet; ook als deze exact één is.

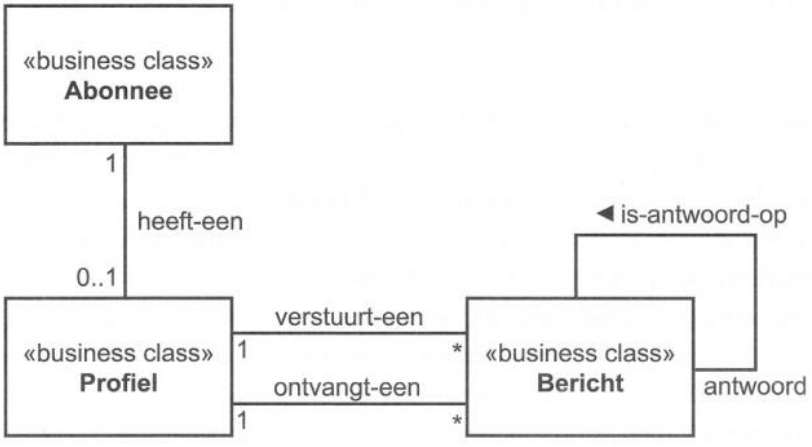


afbeelding 107 – Associaties met multipliciteit

Reflexieve associaties

Een opvallende associatie is die van een klasse met zichzelf. Dit heet een *reflexieve associatie.* Beide uiteinden van de associatie komen uit bij de bewuste klasse. Bij een reflexieve associatie zijn de naam en de rollen essentieel. Zonder deze eigenschappen is de associatie lastig te interpreteren.
Ook hier geldt dat wanneer de naam van de klasse een van de rollen afdoende beschrijft, deze niet wordt benoemd. Beschrijf in een reflexieve associatie in elk geval één van de rollen. Immers, de klasse speelt nooit twee keer dezelfde rol. Het toevoegen van een naampijl is in een reflexieve associatie geen overbodige luxe.

Tip 271. Voorzie iedere reflexieve associatie van een eenduidige naam. Modelleer tenminste één van de rollen en plaats een naampijl.



afbeelding 108 – Reflexieve associatie

Op de website van Dare2Date kan op een bericht altijd een antwoord worden verstuurd. Dit antwoord is ook weer een bericht. Deze situatie is gemodelleerd als een reflexieve associatie van **Bericht** in afbeelding 108. Deze reflexieve associatie leest als **Antwoord is-antwoord-op Bericht**.
Beperk het gebruik van reflexieve associaties zoveel mogelijk. Helaas zijn ze in het dagelijkse leven niet altijd te vermijden. Het uitwerken van reflexieve associaties in code ligt niet altijd voor de hand en heeft vaak een recursief karakter.

Navigeerbaarheid

Associaties zoals die in afbeelding 108 drukken uit dat de betrokken klassen kennis van elkaar hebben. **Abonnee** kent **Profiel** en vice versa. Niet in alle gevallen is het gewenst, en ook niet nodig, dat beide klassen elkaar kennen. Meestal is het voldoende dat een van beide de ander kent en niet andersom. Van iedere **Voorkeur** is bekend bij welk **Profiel** deze hoort. UML duidt dit aan door te zeggen dat de associatie tussen **Profiel** en **Voorkeur** navigeerbaar is in de richting van **Profiel**. Andersom is vooralsnog niet nodig. Ieder profiel heeft immers mogelijk meerdere voorkeuren. Hiervoor is **Profiel** afhankelijk van de factory **Voorkeuren**. Navigeerbaarheid wordt gemodelleerd door een pijlpunt te plaatsen aan het eind van de associatie in de richting van de navigatie. Een associatie die in één richting navigeerbaar is, heet een *uni-directionele associatie*. Een associatie die in beide richtingen navigeerbaar is, wordt een *bi-directionele associatie* genoemd.

Tip 272. Geef aan dat een associatie navigeerbaar is in één richting als de ene klasse kennis heeft van de andere. Geef aan dat een associatie navigeerbaar is in beide richtingen als beide klassen kennis hebben van de andere.

Er zijn drie variaties in het modelleren van navigeerbaarheid. Ik hanteer ze als volgt:

- *Niet nader gespecificeerd.* De navigatie kent geen pijlpunten. Gebruik deze notatie zolang onbekend is welke klasse kennis heeft van de ander, zoals in afbeelding 108 tussen **Abonnee** en **Profiel**.
- *Uni-directioneel.* Eén van de klassen heeft kennis van de ander nodig. **Voorkeur** kent een bijbehorend **Profiel**.
- *Bi-directioneel.* Beide klassen hebben kennis over de ander nodig, zoals in de associatie tussen **Abonnee** en **Profiel** in afbeelding 109.

Door de navigeerbaarheid op deze manier te modelleren is aan het business class diagram direct te zien welke associaties reeds bekend zijn en naar welke nog moet worden gekeken. Immers, een associatie zonder pijlpunten is nog ongedefinieerd.