Rijp voor discussie en verdere ontwikkeling

Aggregatie een veelbelovende constructie

Henk Jan Pels

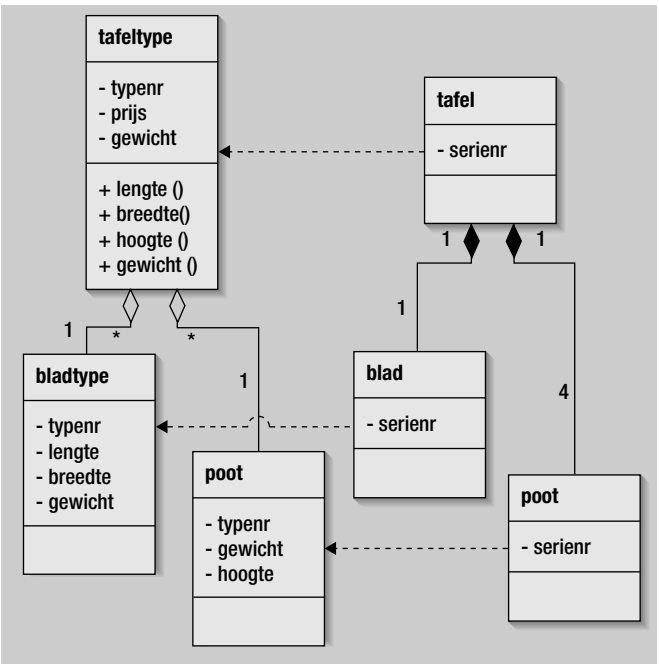
Aggregatie is een abstractie waarbij we een samengesteld ding beschouwen en dan afzien van de details van de onderdelen, zodat we ons kunnen concentreren op de eigenschappen van het geheel. Het meest typische voorbeeld van aggregatie, waarin het woord ook als zodanig gebruikt wordt, is het aggregaat. Een aggregaat is de samenstelling van een motor en een generator. Men spreekt dan bijvoorbeeld over een "dieselaggregaat met een vermogen van 50 kilowatt en een gewicht van 120 kilogram voor een prijs van 2000 Euro". In deze omschrijving hoort de eigenschap 'diesel' eigenlijk bij de motor, terwijl het 'vermogen van 50 kilowatt' afkomstig is van de generator. Het 'gewicht van 120 kilogram' is het resultaat van de optelling van de gewichten der delen. De 'prijs van 2000 Euro' is een eigenschap die alleen bij het geheel hoort en niet kan worden herleid uit de prijzen van de onderdelen.

AGGREGATIE

In dit voorbeeld van het aggregaat komen de voornaamste kenmerken van de aggregatie-abstractie naar voren. Een aantal onderdelen wordt samengevoegd tot een geheel. We spreken daarna over de eigenschappen van het geheel en zien af van eigenschappen van afzonderlijke delen, zoals de gewichten van de delen. Aan het geheel kunnen we specifieke eigenschappen toekennen, zoals de prijs, maar het geheel erft ook eigenschappen van de delen. Dat kan op verschillende manieren: het vermogen wordt rechtstreeks geërfd van de generator, maar het gewicht wordt geërfd als de som van de gewichten van de delen.

In het 'klassieke' datamodelleren uit de jaren tachtig werd aggregatie al als abstractie onderkend, zij het op een wat vreemde, en wat mij betreft oneigenlijke manier: het object werd beschouwd als aggregatie van zijn attributen. Op zich is daar niets mis mee, maar het is niet wat we zoeken. De term wordt namelijk toegepast op de elementen van het datamodel, en niet op die van de te modelleren wereld. Later werd in UML wel de aggregatie als constructie gebruikt, maar dan louter als een bijzondere vorm van associatie.

In afbeelding 1 is het voorbeeld van het aggregaat in UML weergegeven, niet voor een dieselaggregaat, maar voor een tafel, bestaande uit een blad en vier poten.
In het model van afbeelding 1 wordt onderscheid gemaakt tussen het type en exemplaar. Daarmee kan het onderscheid worden gemodelleerd dat UML maakt tussen twee vormen van aggregatie: *shared* en *composite*. Het tafeltype is gemodelleerd als de shared aggregatie van een blad en vier poten. Shared betekent dat hetzelfde bladtype kan worden toegepast in verschillende tafeltypen, bijvoorbeeld met dezelfde lengte en breedte, maar een verschillende hoogte. Evenzo kan hetzelfde poottype in verschillende tafeltypen met verschillende groottes van het blad worden toegepast. Rechts van de typen zijn de exemplaren tafel, blad en poot gemodelleerd. Deze objectklassen staan voor de individuele exemplaren die geproduceerd worden. Ze hebben alle een serienummer, want het betreft zeer exclusieve producten. Hier is een composite aggregatie gebruikt, om aan te geven dat een specifiek blad- of pootexemplaar in niet meer dan één tafel kan worden gebruikt. Dit in



AFBEELDING 1: AGGREGATIE IN UML.

tegenstelling tot de typen, die wel op meerdere plaatsen kunnen worden toegepast.

In plaats van de termen shared en composite wordt in sommige publicaties ook wel gesproken over 'catalogus' en 'fysieke' aggregatie. Ik vind dat sprekender benamingen, omdat inderdaad het linker deel van afbeelding 1 overeenkomt met de catalogus van de tafelfabrikant, terwijl het rechterdeel de verzameling feitelijk geproduceerde fysieke producten voorstelt.

Ik heb gebruik gemaakt van de classificatie-abstractie zoals die in de voorgaande artikelen is beschreven, om elegant te kunnen modelleren dat tafel-exemplaren de prijs en het typenummer, maar ook de waarden voor gewicht, lengte, breedte en hoogte van hun tafeltype erven. Evenzo erven poten en bladen de attribuutwaarden van hun typen.

Modellering van scratch af aan (5)

In het vorige artikel liet Henk Jan Pels zien hoe classificatie gebruikt kan worden om de impliciete abstracties in de wereld van document management bespreekbaar te maken. Er is toen beloofd dat het volgende artikel over productfamilies zou gaan. Daar houdt Henk Jan Pels zich niet helemaal aan, want er hoort nog een stapje tussen. Een belangrijke eigenschap van producten is dat ze zijn samengesteld uit componenten. Dat is typisch een voorbeeld van aggregatie. Hij neemt de lezer eerst mee naar de abstractie die aggregatie wordt genoemd. Die passen we toe in een gewone productstructuur. In het volgende nummer komt hij dan aan bij de productfamilie.

DE SEMANTIEK VAN AGGREGATIE

In UML is aggregatie niet meer dan een tekensymbool dat aangeeft dat het blad en de poten niet als afzonderlijke dingen moeten worden gezien, maar als een geheel dat tafel heet. UML geeft geen expliciete betekenis aan, maar biedt de mogelijkheid om desgewenst programmeursfaciliteiten als een cascaded delete te bie-

Een toepassing is hier dus een instantie van een producttype

den. Met alleen deze betekenis kun je echter nauwelijks van een abstractie spreken. Daarom heb ik er een paar dingen aan toegevoegd die voor het conceptueel modelleren van aggregatie handig kunnen zijn.

Om te beginnen hebben we weer een tekstuele notatie nodig als equivalent voor de UML-notatie. UML ziet aggregatie als een bijzondere vorm van associatie, waarin de delen gekoppeld worden tot een geheel. Door nu het sleutelwoord 'aggregatie' te gebruiken in plaats van 'associatie' wordt aangegeven dat het hier om de aggregatie-abstractie gaat:

```
<<bladtype, aggregatie, tafeltype, 1>>
```

Net als bij associaties gaan we weer uit van de 'heeft'-betekenis: "bladtype 'heeft' aggregatie tafeltype". Het verschil tussen aggregatie en associatie stoppen we nu in extra semantiek voor de aggregatie. Om te beginnen heeft een aggregatie impliciet de rolnamen aggregaat en component:

```
<<bladtype, aggregatie, aggregaat: tafeltype
omgekeerd component, 0>>
```

De vereiste cardinaliteiten kunnen net als bij associaties worden gespecificeerd. Dat een tafeltype als componenten één bladttype en één poottype heeft (de vier poten zijn van hetzelfde type) blijkt uit:

```
<<bladtype, aggregatie, tafeltype omgekeerd[1],
1>>,
<<poottype, aggregatie, tafeltype omgekeerd[1],
1>>
```

Dat een tafel één blad en vier poten heeft en dat een blad of poot in ten hoogste één tafel kan voorkomen, geven we aan met:

```
<<blad, aggregatie, tafel[0..1] omgekeerd[1], 1>>,
<<poot, aggregatie, tafel[0..1] omgekeerd[4], 1>>
```

Een typische eigenschap van een aggregaat is dat het uit meerdere componenten bestaat. Daarom is het eerste stukje semantiek dat we aan de aggregatie toedichten de regel dat elke aggregatie impliciet de rolnamen aggregaat en component meekrijgt. In meer formele termen volgt uit bovenstaande aggregatie van blad naar tafel:

```
<<blad, aggregatie, aggregaat: tafel [0..1]
omgekeerd component[1], 0>>
```

Voor bijvoorbeeld de aggregatie van bladdtype naar tafeltype betekent dit dat:
als b = een bladtype,
dan b.aggregaat = de verzameling van alle tafeltypen waarin b toegepast wordt,
en
als t = een tafeltype,
dan t.component = de verzameling bestaande uit het bladttype en het poottype waaruit t bestaat.

Er bestaat een theorie over aggregaties met gehelen en delen, die bekend is onder de naam 'Mereologie'. In deze theorie worden de

operaties deel (*part*) en geheel (*whole*) gedefinieerd in de betekenis dat de operatie part, toegepast op een element in een aggregatie-hiërarchie, de verzameling oplevert van zichzelf en alle elementen onder hem. De operatie whole levert de verzameling bestaande uit het element en alle elementen daarboven. Het ligt voor de hand

Het mekka voor aggregaties moet wel de toepassing van stuklijsten zijn

om deze begrippen ook in de semantiek van de aggregatie op te nemen: als een objectklasse genoemd wordt als doel van een aggregatie, dan heeft hij een afgeleid attribuut-'deel' dat voor elk object de verzameling oplevert van al zijn delen in mereologische zin, dus van al zijn componenten en de delen van die componenten. Elke objectklasse die een aggregatie heeft, krijgt een afgeleid attribuut-'geheel' dat voor elk object de verzameling oplevert van al zijn aggregaten verenigd met de gehelen daarvan. Formeel kunnen we dat, voor de lezers die bovenstaande zinnen niet helemaal eenduidig vinden, als volgt noteren;

```
<<cc, aggregatie, ca>>
```
impliceert de operaties
```
<<ca, operatie, deel() set post: result = {self}
∪ self.component ∪ self.component.deel, 0>>
and
<<cc, operatie, geheel() set post: result = {self}
∪ self.aggregaat ∪ self.aggregaat.geheel, 0>>
```

Als u geen zin heeft zich in deze formulebrij te verdiepen, dan is het toch goed om te onthouden dat component de verzameling van de directe onderdelen oplevert, terwijl deel de totale explosie omvat, inclusief het betreffende object zelf.

Net als bij generalisatie en classificatie hoort natuurlijk bij aggregatie ook een beetje erfelijkheid. In afbeelding 1 is die eigenlijk al min of meer aangegeven: het gewicht, de lengte, de breedte en de hoogte van het tafeltype zijn als procedures gedefinieerd. Het ligt voor de hand om te stellen dat het aggregaat die eigenschappen van zijn delen erft. Het zal alleen niet zo eenvoudig gaan als bij generalisatie, waar de subklasse alle beperkingen van de superklasse erft, of bij de classificatie, waar de instantiatie alle attribuutwaarden van zijn klasse erft. Bij aggregatie is er sprake van een principieel verschil: de hogere in de hiërarchie erft van de lagere, terwijl dat bij generalisatie en classificatie dat juist andersom is. Anders gezegd: generalisatie- en classificatiehiërarchieën erven top-down, terwijl aggregatiehiërarchieën juist bottom-up erven.

AGGREGAAT-ATTRIBUTEN

Als je bottom-up erft, wil je zeker niet alle eigenschappen erven, want de bedoeling van de aggregatie is juist om details te verbergen. Dat betekent dat bij aggregaties expliciet zal moeten worden aangegeven welke eigenschappen geërfd worden. Zo worden lengte, breedte en hoogte geërfd als kopie van overeenkomstige attributen van de componenten. We noemen dit *aggregaat-attributen*. De hoogte van het tafeltype specificeren we bijvoorbeeld als:

```
<<tafeltype, aggrattr, hoogte: kopie, 1>>
```

De precieze betekenis hiervan is dat voor elk tafeltype, dit attribuut als waarde krijgt: de verzameling van alle waarden die als waarde van een attribuuthoogte van de diverse componenten voorkomen. In dit geval heeft alleen het poottype een eenwaardige attribuuthoogte en er is maar één poottype per tafeltype. Een tafeltype heeft dus precies één waarde voor hoogte. Het gewicht van een tafeltype moet worden berekend als de som van de gewichten van de componenten:

```
<<tafeltype, aggrattr, gewicht: som, 1>>
```

De betekenis hiervan is dat de waarde van gewicht voor een tafeltype berekend wordt als de som van het gewicht van de componenten. Formeel:

```
<<tafeltype, OCL, gewicht =
sum(c.gewicht: c ∈ self.component), 0>>
```

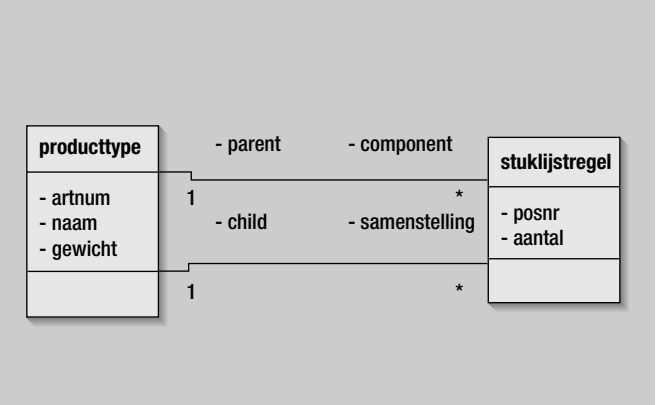
Op analoge manier kan nog een hele reeks van aggregaatattributen gedefinieerd worden zoals gemiddelde, kleinste, grootste enzovoort. Het heeft weinig zin om die hier allemaal op te sommen. Laten we zeggen dat u, mocht u er toe komen de hier gedefinieerde aggregatie-abstractie ooit toe te passen, vrij bent om zulke aggregaatattributen naar behoefte toe te voegen. Teneinde aggregaatattributen ook te kunnen toepassen in recursieve aggregaties (objectklasse is aggregatie van zichzelf) spreken we af dat aan een aggregaatattribuut een waarde mag worden toegekend als er geen componenten zijn die de waarde kunnen bepalen.

STUKLIJSTEN

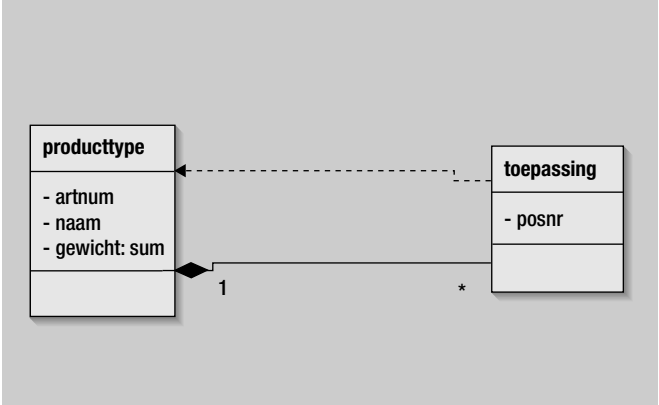
Het mekka voor aggregaties moet wel de toepassing van stuklijsten zijn. Als afsluiting van dit artikel laten we daarom nog even zien welke rol aggregatie hierin speelt. Het klassieke model van een stuklijst is weergegeven in afbeelding 2.

In de klassieke stuklijst wordt, zoals afgedwongen door het relationele model, de n:m associatie tussen een samenstelling en een component gemodelleerd met de stuklijstregel, die met twee associaties verbonden is met de *parent* respectievelijk het *child* (de Engelse termen zijn hier niet elegant te vermijden). In de stuklijstregel wordt met posnr (positienummer) aangegeven waar de component op de tekening te vinden is. Het aantal geeft aan in welke hoeveelheid het gebruikt wordt.

Afbeelding 3 laat zien hoe een stuklijst eruit ziet met de aggregatieabstractie. De term stuklijstregel is vervangen door de



AFBEELDING 2: KLASSIEKE STUKLIJST.



AFBEELDING 3: STUKLIJST MET AGGREGATIE.

term toepassing. In plaats van twee associaties zijn een classificatie en een aggregatie gebruikt. Een toepassing is hier dus een instantie van een producttype: het is letterlijk de toepassing van dat producttype als component van een samenstelling. Een producttype kan meermalen worden toegepast (catalogusaggregatie). Het attribuut aantal is verdwenen: het is eleganter om te modelleren dat een producttype meerdere malen wordt toegepast. De aggregatie beschrijft dat een samengesteld producttype gezien

Als je bottom-up erft wil je zeker niet alle eigenschappen erven

moet worden als de aggregatie van een aantal toepassingen. Door de classificatie erft de toepassing de attribuutwaarden van zijn producttype. Door de aggregatie krijgt vervolgens het gewicht van een samenstelling als waarde de som van de gewichten van zijn componenten. Ter illustratie specificeren we een bevolking voor de structuur van afbeelding 3:

```
<p1, artnum, 101, 1>, <p1, naam, eettafel, 1>,
<(p1, producttype, 1)>
<p2, artnum, 102, 1>, <p2, naam, langepoot, 1>,
<p2, gewicht, 5, 1>, <(p2, producttype, 1)>
<P3, artnum, 103, 1>, <p3, naam, eetblad, 1>,
<p3, gewicht, 15, 1>, <(p3, producttype, 1)>
<t1, posnr, 1, 1>, <(t1, p2, 1)>,
<(t1, toepassing, 1)>
<t2, posnr, 2, 1>, <(t2, p2, 1)>,
<(t2, toepassing, 1)>
<t3, posnr, 3, 1>, <(t3, p2, 1)>,
<(t3, toepassing, 1)>
<t4, posnr, 4, 1>, <(t4, p2, 1)>,
<(t4, toepassing, 1)>
<t5, posnr, 2, 1>,  <(t5, p3, 1)>,
<(t5, toepassing, 1)>
```

Op basis van de semantiek van het model volgen nu onder andere de volgende afgeleide gegevens:

```
<t1, artnum, 102, 0>, <t1, naam, langepoot, 0>,
<t1, gewicht, 5, 0>,
<t5, artnum, 103, 0>, <t5, naam, eetblad, 0>,
<t5, gewicht, 15, 0>,
<p1, gewicht, 45, 0>
```

Het gewicht van de samenstelling is dus impliciet bepaald. De artikelnummers van de componenten van de tafel worden eenvoudige gevonden met:

```
p1.component.artnum
```

CONCLUSIE

Van aggregatie is een interessante abstractie te maken door er de juiste semantiek aan toe te kennen. Het lijkt een veelbelovende constructie voor het modelleren van productstructuren. Tegelijk is het idee nog nieuw en de semantiek van aggregatie zoals hier gedefinieerd is minder elegant dan die van classificatie en generalisatie, omdat hij minder eenvoudig is en eigenlijk een beetje ad hoc. Aggregatie in deze vorm is dus zeker voor discussie en voor verdere ontwikkeling vatbaar. Ik zie dus reacties met belangstelling tegemoet. In de volgende aflevering gaan we verder het idee toe te passen in een analyse van productfamilies. ●

Henk Jan Pels (H.J.Pels@tm.tue.nl) is werkzaam aan de Faculteit Technologie Management, Capaciteitsgroep Informatie & Technologie van de Technische Universiteit Eindhoven.