

Three small gray squares are positioned at the top of the page: one on the left, one in the center, and one on the right. The central square is slightly higher than the other two.

Informatieanalyse




Informatie- analyse

Gerlof Donga

Bert Pinkster





Meer informatie over deze
en andere uitgaven kunt u
verkrijgen bij:

Sdu Klantenservice
Postbus 20014
2500 EA Den Haag
tel.: (070) 378 98 80
fax: (070) 378 97 83

© 2007 Sdu Uitgevers bv,
Den Haag
Academic Service is een
imprint van Sdu Uitgevers bv

1e druk, 1e oplage december 2003
2e oplage september 2004
2e druk, 1e oplage mei 2007

Vormgeving en omslag: Studio
Bassa, Culemborg
Zetwerk: Redactie bureau Heijer,
Markelo

ISBN: 978 90 395 2546 3
NUR: 124

Alle rechten voorbehouden. Alle auteursrechten en databank-
rechten ten aanzien van deze uitgave worden uitdrukkelijk
voorbehouden. Deze rechten berusten bij Sdu Uitgevers bv.

Behoudens de in of krachtens de Auteurswet 1912 gestelde
uitzonderingen, mag niets uit deze uitgave worden verveel-
voudigd, opgeslagen in een geautomatiseerd gegevensbestand
of openbaar gemaakt in enige vorm of op enige wijze, hetzij
elektronisch, mechanisch, door fotokopieën, opnamen of enige
andere manier, zonder voorafgaande schriftelijke toestemming
van de uitgever.

Voorzover het maken van reprografische verveelvoudigin-
gen uit deze uitgave is toegestaan op grond van artikel 16 h
Auteurswet 1912, dient men de daarvoor wettelijk verschul-
digde vergoedingen te voldoen aan de Stichting Reprorecht
(postbus 3051, 2130 KB Hoofddorp, www.reprorecht.nl).
Voor het overnemen van gedeelte(n) uit deze uitgave in
bloemlezingen, readers en andere compilatiewerken (artikel
16 Auteurswet 1912) dient men zich te wenden tot de Stichting
PRO (Stichting Publicatie- en Reproductierechten Organisatie,
Postbus 3060, 2130 KB Hoofddorp, www.cedar.nl/pro). Voor
het overnemen van een gedeelte van deze uitgave ten behoeve
van commerciële doeleinden dient men zich te wenden tot de
uitgever.

Hoewel aan de totstandkoming van deze uitgave de uiter-
ste zorg is besteed, kan voor de afwezigheid van eventuele
(druk)fouten en onvolledigheden niet worden ingestaan en
aanvaarden de auteur(s), redacteur(en) en uitgever deswege
geen aansprakelijkheid voor de gevolgen van eventueel voor-
komende fouten en onvolledigheden.

All rights reserved. No part of this publication may be repro-
duced, stored in a retrieval system, or transmitted in any form
or by any means, electronic, mechanical, photocopying, recor-
ding or otherwise, without the publisher's prior consent.

While every effort has been made to ensure the reliability of
the information presented in this publication, Sdu Uitgevers
neither guarantees the accuracy of the data contained herein
nor accepts responsibility for errors or omissions or their con-
sequences.

Woord vooraf

Bij de opleiding voor ICT-beheerder op niveau 4 verwachten we dat deze ook begrip heeft voor de eisen die aan de ontwikkeling van informatiesystemen gesteld worden. Dit is een breed en diep vakgebied waarin men snel de weg kwijt raakt. Schematisch wordt een basis gelegd voor de systeemontwikkeling van databases, applicaties, administratieve systemen, infrastructuren, productiesystemen, enzovoort. Naar opvatting van de auteurs ligt de nadruk op systeemontwikkeling en niet op het ontwerpen van de infrastructuur. Dit laatste komt in andere onderdelen van de opleiding uitgebreid aan de orde.

Dit boek geeft een voorzichtige inleiding in systeemontwikkelingsmethodieken waarbij de objectgeoriënteerde aanpak centraal staat. Daarbij wordt uitgebreid gebruik gemaakt van een subset van UML-diagrammen. In dit boek laten we de theoretische kant van elk onderdeel steeds volgen door een uitwerking in een 'case study'. Zo wordt in de loop van het boek steeds verder gewerkt aan de bouw van een customer relation management system. Centraal in het boek staat de behandeling van klassen. Deze klassen spelen een belangrijke rol bij het objectgeoriënteerd denken, maar kunnen ook een centrale rol spelen bij het databaseontwerp als vervanging van de oudere ERD-diagrammen. Hiermee legt dit boek ook een verband met databases en de toepassing van SQL. We behandelen hier niet een volledige set UML-diagrammen. Het implementatie-deel van de UML-diagrammen staat ver af van de toepassing door MBO'ers in de beroepspraktijk. De diagrammen die in dit boek zijn afgebeeld, zijn gemaakt met het programma Microsoft Visio Professional 2002.

Dit boek betekent een uitbreiding van de serie voor de MBO-opleiding tot ICT-beheerder; wij zijn erg benieuwd hoe het in de praktijk bevalt. Elke op- en aanmerking wordt door de auteurs zeer

gewaardeerd. Daarmee ontstaan in de toekomst immers alleen maar betere boeken.

Veel succes met het werken met dit boek!

September 2003

Gerlof Donga
Bert Pinkster

Bij de tweede druk

In deze druk is een aantal fouten verbeterd. Tevens is op verzoek van diverse gebruikers een hoofdstuk over toestandsdiagrammen toegevoegd.

Mei 2007

Gerlof Donga
Bert Pinkster

Inhoud

| | | |
|----------|--|-----------|
| 1 | Ontwikkelmethoden | 1 |
| 1.1 | Inleiding | 1 |
| 1.2 | Lineaire methodes | 1 |
| 1.3 | Incrementele modellen | 7 |
| 1.4 | Het spiraalmodel | 9 |
| 1.5 | OO-levenscyclusmodellen | 10 |
| 1.6 | Extreme programming | 11 |
| 1.7 | Het synchroniseer-en-stabiliseermodel | 12 |
| 1.8 | Samenvatting | 12 |
| 2 | De case PartyCups – functionele requirements in use-cases | 15 |
| 2.1 | Inleiding | 15 |
| 2.2 | De case PartyCups – introductie | 15 |
| 2.3 | De scope | 16 |
| 2.4 | Aanpassingen aan bestaande systemen | 17 |
| 2.5 | Nieuwe applicaties | 19 |
| 2.6 | Het interview | 19 |
| 2.7 | De Unified Modeling Language | 20 |
| 2.8 | Niet-functionele requirements | 22 |
| 2.9 | Pseudo-requirements | 23 |
| 2.10 | User stories | 23 |
| 2.11 | Van user story naar use-case diagram | 25 |
| 2.12 | Use-case templates | 26 |
| 2.13 | Samenvatting | 29 |
| 3 | De case PartyCups – activiteitendiagrammen | 37 |
| 3.1 | Inleiding | 37 |
| 3.2 | Het activiteitendiagram | 37 |
| 3.3 | De case ‘PartyCups’ – de organisatie | 42 |
| 3.4 | De orderadministratie in schema | 46 |
| 3.5 | Bijzondere bestellingen | 50 |
| 3.6 | Samenvatting | 51 |

| | | |
|----------|--|------------|
| 4 | Functioneel ontwerp – het bouwen van klassen- diagrammen | 61 |
| 4.1 | Inleiding | 61 |
| 4.2 | Het klassendiagram | 61 |
| 4.3 | Overerving | 62 |
| 4.4 | Van kandidaat-klassen tot klassen | 63 |
| 4.5 | De case PartyCups | 65 |
| 4.6 | Attributen en operaties | 66 |
| 4.7 | Klasse en object | 68 |
| 4.8 | Samenvatting | 69 |
| 5 | Functioneel ontwerp – specificaties voor een database | 73 |
| 5.1 | Inleiding | 73 |
| 5.2 | Multipliciteit | 73 |
| 5.3 | Attributen en operaties | 74 |
| 5.4 | Aggregatie en compositie: gekwalificeerde associaties | 75 |
| 5.5 | Toepassing op databases | 76 |
| 5.6 | De database bij PartyCups | 78 |
| 5.7 | Samenvatting | 83 |
| 6 | Een methodische aanpak van softwarebouw | 85 |
| 6.1 | Inleiding | 85 |
| 6.2 | Unified Process | 86 |
| 6.3 | Startfase (inception phase) | 87 |
| 6.4 | Uitwerkingsfase (elaboration phase) | 90 |
| 6.5 | Bouwfase (construction phase) | 92 |
| 6.6 | Opleverfase (transition phase) | 93 |
| 6.7 | De case PartyCups – het inlog subsysteem als voorbeeld | 94 |
| 6.8 | Samenvatting | 100 |
| 7 | Sequentiediagrammen – Het gedrag in modellen en de communicatie tussen objecten | 103 |
| 7.1 | Inleiding | 103 |
| 7.2 | Het doel van een sequentiediagram | 103 |
| 7.3 | De notatie van onderdelen van het sequentiediagram | 104 |
| 7.4 | De berichten | 105 |

| | | |
|--------------|---|------------|
| 7.5 | Tijdgebonden overgangen | 106 |
| 7.6 | Iteraties | 107 |
| 7.7 | Voorbeelden van sequentiediagrammen | 108 |
| 7.8 | Sequentiediagrammen bij PartyCups | 112 |
| 7.9 | Samenvatting | 113 |
| 8 | Toestandsdiagrammen – het veranderende gedrag van één object | 115 |
| 8.1 | Inleiding | 115 |
| 8.2 | Het doel van een toestandsdiagram | 115 |
| 8.3 | De notatie van onderdelen van een toestandsdiagram | 116 |
| 8.4 | Voorbeelden van toestandsdiagrammen | 117 |
| 8.5 | Samenvatting | 121 |
| Index | | 125 |

1.1 Inleiding

Het bouwen van een boomhut is niet zo moeilijk. Je zorgt voor wat planken, spijkers en touw. Je zoekt een stevige tak uit en gaat bouwen. Misschien lukt het niet in één keer, maar met wat doorzettingsvermogen, misschien wat afkijken en advies van anderen krijg je het heus wel voor elkaar. Vol trots denk je dan aan een groter project. Je zult een huis gaan bouwen. Al gauw loop je tegen onoverkomelijke problemen aan. Een boomhut is blijkbaar toch iets heel anders dan een huis. Bij softwareontwikkeling is het net zo. Een programma maken of een website bouwen lukt wel, maar als je echt een applicatie wilt bouwen die in een informatiesysteem gebruikt kan worden, is er wel meer nodig. In dit hoofdstuk gaan we kijken naar enkele methoden voor softwareontwikkeling.

1.2 Lineaire methodes

De oudste methodes voor softwareontwikkeling zijn de **lineaire methodes**. Stap voor stap wordt het ontwikkelingsproces doorlopen. Een volgende stap kan pas starten als de vorige is afgerond. Het is te vergelijken met het bakken van een brood. Het deeg kan pas bewerkt worden als het gerezen is en het bakken is de laatste stap. Dergelijke methoden van softwareontwikkeling zijn tijdrovend. Een doorlooptijd van meer dan een jaar is heel gebruikelijk.

Terzijde

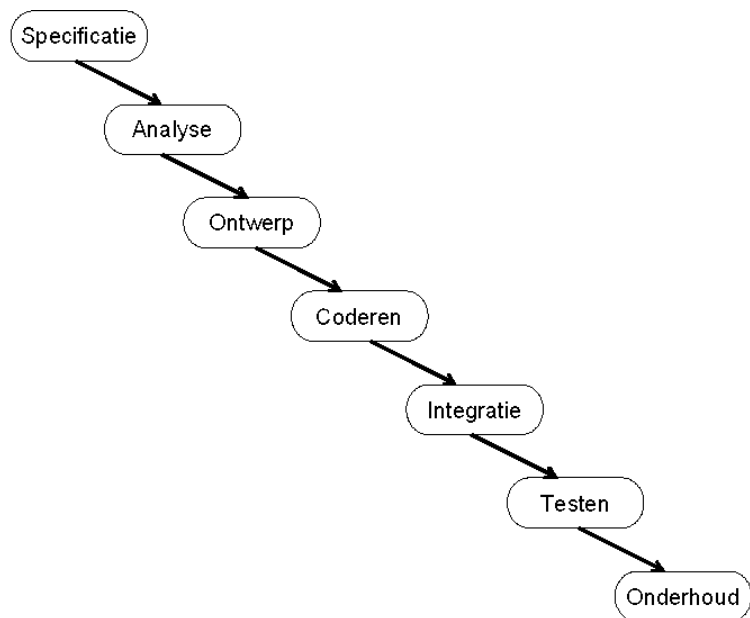
De **9-9-9 regel**

Was het vroeger gebruikelijk dat een softwareontwikkeltraject negen maanden duurde, tegenwoordig moet een dergelijk project in negen weken zijn afgerond. Over een tijdje kunnen we zelfs een ontwikkeltijd van negen dagen verwachten.

1.2.1 De watervalmethode

De **watervalmethode** is al ruim dertig jaar geleden ontwikkeld als lineaire methode.

Figuur 1.1
De stappen bij de
watervalmethode



Terzijde

Steeds vaker ziet men bij de ontwikkeling van software dat ook het moment waarop de software niet meer gebruikt zal worden al in het ontwikkelproces wordt opgenomen. Slechts weinig software wordt voor de eeuwigheid gebouwd. Als van tevoren bepaald wordt dat de ontwikkelde software een verwachte levensduur heeft van bijvoorbeeld vijf jaar, dan kan

daar bij de ontwikkeling rekening mee gehouden worden. De financiële kant van de opdracht wordt direct helder. We weten hoeveel er geïnvesteerd wordt en over welke periode de investering afgeschreven moet worden.

De eerste stap bij de watervalmethode is de specificatie. Daarbij legt men vast waaraan het uiteindelijke softwareprogramma moet voldoen. De eisen die de opdrachtgever stelt, worden helder en eenduidig beschreven. Als de specificatie voltooid is, moet de opdrachtgever zijn handtekening hieronder zetten. Na het plaatsen van de handtekening is de specificatie definitief. Er kan niet meer van afgeweken worden. Voor de opdrachtgever betekent dit, dat de specificatie volledig moet zijn. Ook de makers van de specificatie moeten zorgvuldig werken. Immers, de opdracht moet ook nog uitgevoerd kunnen worden. De handtekening geeft het einde van deze fase aan. De specificatie gaat door naar de volgende groep, de analisten.

De analisten gaan met behulp van de specificatie onderzoeken hoe de toepassing het best gebouwd kan worden. De functionaliteit van de toepassing wordt bekeken. Wat zijn logische opeenvolgende stappen? Kortom, de analisten maken een schets van de toepassing. Ook hier wordt de fase weer afgesloten door een handtekening van de opdrachtgever.

De volgende stap wordt gedaan door de ontwerpers. Zij zullen erop letten dat veel voorkomende handelingen op eenvoudige wijze verricht kunnen worden. Verder geven zij aan welke onderdelen op welke manier gebouwd moeten worden. Daarmee geven zij al een taakverdeling aan voor de volgende groep, de programmeurs. Die groep kan pas starten als er weer een handtekening door de opdrachtgever gezet is. De programmeurs zorgen voor de code van de programmaonderdelen. Uiteraard hebben de ontwerpers al aangegeven wat er aan invoer en aan uitvoer van elk programma-onderdeel verwacht wordt. Na de integratie van de verschillende programmaonderdelen horen er eigenlijk vrijwel geen aanpassingen meer nodig te zijn, zodat het product aan verschillende testen blootgesteld kan worden. Na de oplevering (handtekening!) moet het product probleemloos volgens de specificatie werken. Pas dan blijkt of de specificatie goed was. In het ergste geval werkt het

product probleemloos volgens specificatie, maar levert het niet de gewenste resultaten.

De methode wordt de watervalmethode genoemd, omdat men na afronding van een stap nooit meer terug kan. Het is net zo als bij een waterval; water dat van de cascade naar beneden gaat, kan nooit meer terug.

De watervalmethode heeft een aantal voordelen:

- De eisen aan het product worden scherp vastgelegd. Er is dus geen discussie achteraf over wat het product wel of niet kan.
- Er kan scherp gepland worden. De opleveringsdata van de verschillende stappen kunnen contractueel vastgelegd worden.
- Elke stap is in feite een project. Op projecten kunnen projectmanagementtechnieken toegepast worden

Nadelen zijn er echter ook:

- De eisen aan het product worden scherp vastgelegd. Er is dus geen ruimte voor aanpassingen onderweg. Van tevoren moet volkomen helder zijn wat het product gaat bieden.
- Pas na integratie van de programmaonderdelen blijkt wat het product doet. Men noemt dit wel de 'Big Bang'-aanpak.
- Het testen vindt pas na afloop plaats. Ernstige fouten leiden tot het overdoen van alle voorgaande stappen.
- Hierdoor wordt het hele proces onbestuurbaar. Elke herstart heeft gigantische gevolgen.
- Vaak wordt daarom aan het einde een reparatieslag uitgevoerd. Hierdoor wordt de architectuur ondoorzichtig en wankel. Soms spreekt men zelfs van een kaartenhuisconstructie.

Terzijde

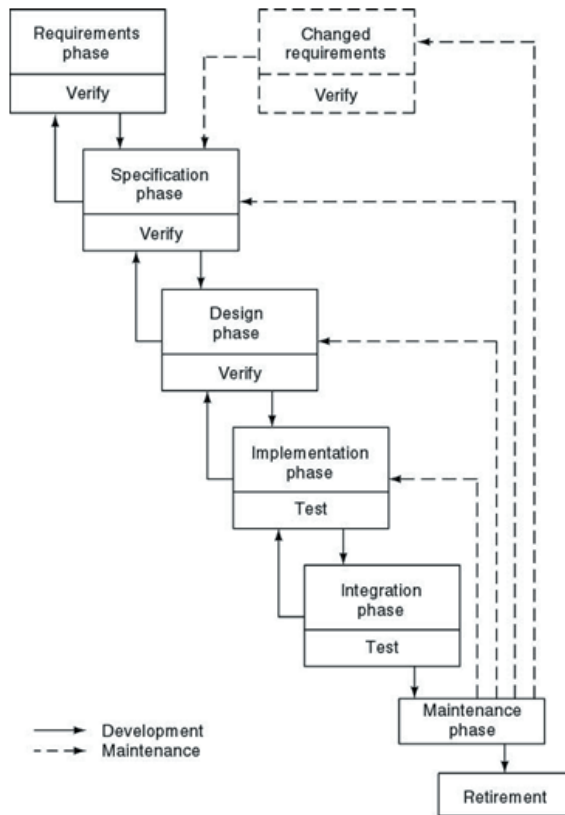
De watervalmethode is goed te vergelijken met de wijze waarop huizen gebouwd worden. Ook daar werkt men in lineaire stappen. Eerst wordt een programma van eisen opgesteld waaraan het huis moet voldoen. Daarin staat bijvoorbeeld het aantal kamers, de woonoppervlakte, de hoogte van de plafonds en nog veel meer. De volgende groep maakt schetsen van de woningen, hoe ze passen in de omgeving en hoe men denkt aan de eisen te voldoen. Als deze goedgekeurd zijn, worden de bouwtekeningen gemaakt. Daarop staan de exacte

maten, het materiaalgebruik en de wijze waarop de materialen verwerkt moeten worden. Vervolgens gaan de bouwvakkers aan het werk. Elke bouwvakker heeft daarbij zijn specialiteit, bijvoorbeeld elektricien, timmerman, metselaar of dakdekker. Als iedereen volgens voorschrift werkt, past alles en kan iedereen gewoon doorwerken. Na de integratie van het geheel wordt gecontroleerd of alles volgens specificatie gebouwd is en kan de oplevering plaatsvinden. Ook bij een woning blijft er daarna de noodzaak voor onderhoud. Daarbij hoort ook de aanpassing aan nieuwe woonsituaties.

In de praktijk wordt de watervalmethode ook wel de ‘gooi-het-over-de-muur-methode’ genoemd. Elke fase kent immers zijn eigen specialisten. Deze leveren hun product op en vertrekken naar een volgend project. De volgende groep heeft vaak zelfs geen kans om nog navraag te doen bij de vorige groep. Zij zijn dus volledig op zichzelf aangewezen. Erg vervelend wordt het als de vorige groep slecht of onbegrijpelijk (maar wel goedgekeurd) werk heeft geleverd. De nieuwe groep zal dan bijzondere inspanningen moeten leveren om toch nog tot een goed resultaat te komen. Dit komt vaker voor dan je zou verwachten. Een complicerende factor bij ieder project is, dat kort voor de oplevering nog erg veel werk gedaan moet worden. Dit lukt alleen door allerlei extra hulpkrachten in te schakelen. In de automatisering zijn er volop detacheringbureaus die last-minute personeel kunnen leveren. Deze mensen hebben geen tijd om zich in te werken, maar beginnen direct aan de afronding van de klus. Het laat zich raden dat het product daar vaak niet helderder van wordt.

Er is veel te verbeteren aan de watervalmethode. Daarom zijn er ook andere ontwikkelmethoden ontstaan. Het grootste probleem bij elke softwareontwikkeling is de hoeveelheid onderhoud. De **onderhoudskosten** bedragen in de looptijd van een product vaak 70 tot 80 procent van de totale gebruikskosten van het programma. Een programma dat ontwikkeld werd voor € 200.000 blijkt na afloop van het gebruik dus ongeveer € 1.000.000 gekost te hebben.

Figuur 1.2
Onderhoud bij het
watervalmodel



Terzijde

De in Nederland nog zeer populaire ontwikkelmethode **SDM** (System Development Method) is een voorbeeld van de toepassing van het watervalmodel. Ooit, in een tijd dat er nauwelijks ontwikkelmethodes waren, werden veel ontwerpers met deze methode opgeleid. Inmiddels zijn er nieuwere methodes, maar de mensen die opgeleid zijn met vroegere methodes blijven deze nog erg lang gebruiken.

1.2.2 Rapid prototyping model

Er is nauwelijks een verschil tussen het **rapid prototyping model** en het watervalmodel te ontdekken. Er wordt tegemoet gekomen aan één van de belangrijkste bezwaren van het watervalmodel door al in de eerste fases een prototype te bouwen. Hiermee kan de klant

eerder zien of wat hij verwacht, overeenkomt met wat de maker denkt op te leveren. De communicatie met de klant wordt hierbij bevorderd. Een prototype toont de gebruikersinterface en de belangrijkste handelingen. We zien dus eigenlijk de ‘buitenkant’ van het product. De vereisten worden nu verkregen door dit prototype te demonstreren aan de klant. De klant krijgt de kans zijn op- en aanmerkingen te plaatsen. De ontwikkelaar blijft het prototype verbeteren en aanpassen tot het voldoet aan alle eisen van de klant. Na goedkeuring begint de ontwikkelaar weer helemaal opnieuw. Het prototype mag nooit gebruikt worden om het verder te ontwikkelen tot het uiteindelijke product!

Ook deze methode heeft voor- en nadelen. Als belangrijk voordeel geldt, dat de klant al in een vroegtijdig stadium een beeld heeft van het product. Daarmee wordt het ontwikkelproces beter afgestemd op de klant en heel veel problemen bij oplevering worden voorkomen. Een probleem kan zijn dat de klant aan de hand van het prototype hogere verwachtingen gaat koesteren over het product. De aanvullende eisen kunnen leiden tot kostenverhoging en latere levering.

Terzijde

Ook hier is weer een vergelijking te treffen met de bouw-wereld. Aannemers zijn heel vertrouwd met ‘meerwerk’. Het gevraagde bouwwerk wordt scherp gecalculeerd, maar elke aanpassing door de klant kost dan opeens veel geld.

Andere nadelen zijn dat deze werkvorm relatief duur is. Het maken van een goed prototype kost veel tijd en dus ook veel geld. Het product is dan wel beter afgestemd op de eisen van de klant, maar ook duurder en het wordt later opgeleverd.

1.3 Incrementele modellen

In **incrementele modellen** pakken we de zaken fundamenteel anders aan dan in lineaire modellen zoals het watervalmodel. We splitsen de opdracht in een aantal deelopdrachten en spreken van tevoren af hoe de onderdelen op elkaar aansluiten. Elk deel volgt wel hetzelfde traject als in het watervalmodel. Na specificatie en

analyse maken we het ontwerp van dit deel. Na het coderen wordt eerst getest of het deel werkt en vervolgens wordt het toegevoegd aan de andere gereed zijnde onderdelen. Dan wordt gekeken of de onderdelen probleemloos samenwerken. Het grote voordeel is dat we al heel snel een deelresultaat aan de klant kunnen opleveren.

Terzijde

Het is een goed gebruik dat een programmeur niet alleen code produceert, maar daarbij ook uitleg geeft over wat met de code bereikt wordt. Ook een versienummer en de naam van de programmeur hoort in de code thuis. Het komt helaas nog wel eens voor dat jaren later stad en land afgebeld wordt om de programmeur die de code gemaakt heeft, te vinden. Hij is waarschijnlijk de enige die nog kan nagaan hoe hij de code heeft geconstrueerd. Uiteraard is dit heel triest, maar helaas komt deze situatie nog steeds voor.

De klant kan tussentijds feedback geven over de opgeleverde onderdelen. Die feedback kan dan verwerkt worden in de ontwikkeling van de volgende modules. Ook economisch heeft dit model voordelen. In plaats van in één keer de hele opbrengst van het product te ontvangen na oplevering, kunnen we nu betaling van het product per opgeleverd onderdeel vragen. Telkens als er een module opgeleverd wordt, betaalt de klant die module. Ook kan de klant veel sneller onderdelen van zijn product beginnen te gebruiken waardoor ook hij sneller opbrengsten heeft. Het nadeel van dit systeem is dat bij onvolledige afspraken aan het begin of bij veel tussentijdse wijzigingen de eenheid binnen de toepassing ontbreekt. Iedere module heeft dan zijn eigen eigenaardigheden. De grote problemen worden vooruitgeschoven en komen dus pas aan het einde van het project aan bod. Een ander gevaar is een te grote verkaveling. Als de losse modules erg klein zijn, willen we wel de verschillende stappen overslaan en direct met coderen beginnen. Later blijkt de systeemdokumentatie onvolledig en zijn de problemen niet te overzien.

Terzijde

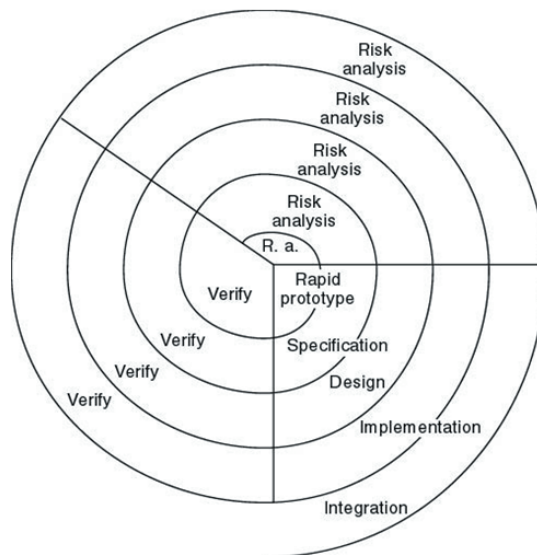
De ontwikkelsnelheid is van steeds groter belang. Niet alleen omdat de klant het product liefst gisteren al wil hebben, maar

ook omdat de ontwikkelkosten beter gecontroleerd kunnen worden. Daarom is er een sterke voorkeur voor methoden als **RAD** (Rapid Application Development)

1.4 Het spiraalmodel

Het **spiraalmodel** is in feite een variant van het watervalmodel. Het enige verschil is dat er een risicoanalyse wordt gemaakt voordat een nieuwe fase begonnen wordt. Als de risico's te groot blijken, wordt er meteen gestopt met de verdere ontwikkeling van het product. Er wordt bijvoorbeeld gekeken of er men nog steeds op het tijdschema zit, of de nodige expertise in huis is, of de economische situatie niet veranderd is, of het product ondertussen niet achterhaald is door een product van de concurrentie, enzovoort. Het mag duidelijk zijn dat het beoordelen van deze risico's erg moeilijk is. Het vraagt veel ervaring en een grondige kennis van het domein waarin de software geschreven wordt. Daarom wordt dit model meestal gebruikt in projecten die 'in-house' ontwikkeld worden. Alleen in die situatie heeft men de juiste experts ter beschikking die de risicoanalyse kunnen doen.

Figuur 1.3
Het spiraalmodel



*Terzijde***Incrementeel en iteratief**

Zoals bij elk vakgebied worden ook bij systeemontwikkeling moeilijke woorden graag gebruikt. Dit leidt nogal eens tot spraakverwarring omdat niet iedereen de juiste betekenis van die woorden kent. Zo worden **incrementeel en iteratief** vaak verward. Iteratief staat voor herhaling. Bij systeemontwikkeling worden vaak stappen herhaald. Omdat we vaak met modules werken, geldt dit voor elk onderdeel, dus ontwerpen, coderen, testen, aanpassen, ontwerpen, coderen, testen, enzovoort tot de module goed werkt. Incrementeel wil zeggen dat er elke keer werkende modules worden toegevoegd worden aan de bestaande applicatie.

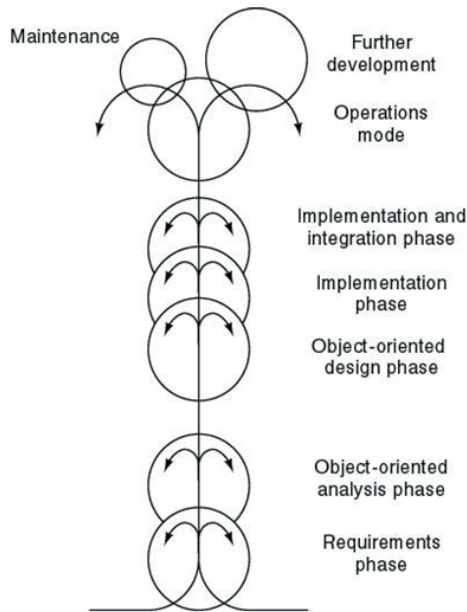
1.5 OO-levenscyclusmodellen

De vorige modellen zijn klassieke **levenscyclusmodellen** die zowel voor gestructureerde als **objectgeoriënteerde methoden** gebruikt worden. Als je echter met een objectgeoriënteerde methode werkt, zijn er modellen die dit beter ondersteunen. Uit ervaring heeft men geleerd dat er bij een OO-methode behoefte is aan meer terugkoppeling tussen de verschillende fases en ook aan meerdere iteraties binnen elke fase. Hiervoor zijn heel specifieke levenscyclusmodellen ontwikkeld. Een voorbeeld hiervan is het **fonteinmodel**.

In het **fonteinmodel** werkt men dikwijls al tegelijk aan een volgende fase. Dit wordt in figuur 1.4 aangegeven door de overlapping van de bollen. Elke bol staat voor een fase in de software levenscyclus en de doorsnede staat voor een soort overgangsperiode waar er intense feedback is tussen die verschillende fases.

De pijlen binnen elke bol geven aan dat er ook binnen elke fase veel iteraties zijn. Men maakt bijvoorbeeld een eerste versie van de vereisten, gaat hiermee naar de klant, past ze aan, toont deze weer, enzovoort.

Figuur 1.4
Het fonteinmodel



1.6 Extreme programming

Dit is een nieuwe benadering van systeemontwikkeling. Het is een reactie op de andere modellen die door de meeste ontwikkelaars als erg saai worden ervaren. Het idee van **extreme programming** is dat we zo snel mogelijk beginnen met het implementeren van het product, zonder een uitgebreide specificatie of ontwerp. Dit kan al snel leiden tot niet-onderhoudbare software. Om dit probleem te omzeilen, wordt continu testen centraal gesteld. Programmeren gebeurt in groepjes van twee. Terwijl de ene ontwikkelaar de gewenste functie schrijft, schrijft de andere de code die deze functie test. Op die manier ontstaat parallel met het programma een testprogramma dat na elke verandering wordt uitgevoerd om na te kijken of alles nog steeds werkt. Andere aspecten van extreme programming zijn:

- De paren worden steeds gewisseld om zo kennis van de verschillende modules aan elkaar door te geven. Let op dat dit niet werkt als we gebruik maken van specialisten: men kan bijvoorbeeld niet zomaar iemand die specialist is in databanktechnologie verwisselen met een specialist in 3D grafische software.
- Iedereen werkt samen in een grote ruimte om de communicatie te bevorderen.

- Er is *altijd* een afgevaardigde van de klant aanwezig, aan wie de programmeurs uitleg kunnen vragen over de details van de requirements (vereisten).

Verder werkt het model grotendeels zoals de incrementele modellen. Er wordt een scenario afgesproken met de klant. De testen voor dit scenario worden geschreven en deze submodule wordt geïmplementeerd.

Het is nog te vroeg om deze aanpak echt te kunnen evalueren, maar ze is wel het aangenaamste voor de ontwikkelaars en er zijn al enkele belangrijke successen mee geboekt.

1.7 Het synchroniseer-en-stabiliseermodel

Dit is het model dat bij Microsoft gebruikt wordt. De bedoeling is dat net als in het incrementele model per module wordt gebouwd, maar in dit model worden alle modules tegelijkertijd gebouwd in plaats van na elkaar. Na elke dag worden de modules geïntegreerd. Men laat dus de verschillende teams elk hun gang gaan tijdens de ontwikkeling van hun module, maar elke avond moeten ze een werkende versie in de databank ingeven. Hierna worden alle modules geïntegreerd. Als de integratie niet lukt door het niet functioneren van één van de modules, moet dat team doorwerken tot hun module geaccepteerd wordt, zodat de volgende dag iedereen op de nieuwe versie kan verder werken. Dit model is ook in andere bedrijven geprobeerd, maar nergens was het zo succesvol als bij Microsoft.

1.8 Samenvatting

Er zijn verschillende types ontwikkelmethodes. De oudste zijn lineair opgebouwd: een volgende stap kan pas genomen worden na voltooiing van de vorige. Deze methodes zijn heel controleerbaar en langzaam. Voor kleine projecten voldoen ze beter dan voor grote. Bij incrementele methodes worden de diverse onderdelen van een ontwikkelproject naast elkaar gestart. Regelmatig zal getest moeten worden of de onderdelen (nog) op elkaar aansluiten. We gebruiken de term iteratief voor zich steeds weer herhalende,

opeenvolgende, gelijksoortige handelingen tijdens de ontwikkeling. Het spiraalmodel is daar een mooi voorbeeld van. Bij extreme programming is de voortdurende aandacht van de klant nodig. Eigenlijk wordt ontworpen in het bijzijn van de klant. Het synchrooniseer-en-stabiliseermodel trekt een zware wissel op de ontwikkelaars, maar kent een erg harde deadline. Bij elk ontwikkeltraject geldt, dat de betrokkenheid en het voorstellingsvermogen van de klant een belangrijke bijdrage levert aan het succes.

Opgaven

1. Zoek (op Internet) naast SDM naar een tweede lineaire ontwikkelmethodiek.
2. Welk groot Nederlands softwarehuis maakt nog steeds gebruik van SDM?
3. Bij het watervalmodel staan voor- en nadelen uitgebreid vermeld. Welke zijn ook van toepassing op het spiraalmodel?
4. Als je zelf gaat ontwikkelen, welke ontwikkelmethodiek zou dan je voorkeur hebben? Waarom?
5. Een kleine ondernemer met weinig ervaring met software wil jou een ontwikkelopdracht geven voor een magazijnbeheersysteem. Welke ontwikkelmethodiek zou het beste bij hem passen? (Hij kent natuurlijk geen ontwikkelmethodieken, maar wat klinkt voor hem het beste?)
6. Een bank wil een deel van zijn effectenverkeer opnieuw laten automatiseren. Vroeger is dit al eens ontwikkeld met SDM. Doordat de beurswereld over drie jaar een nieuwe afrekenmethode in gebruik neemt, moet deze systeemontwikkeling binnenkort uitgevoerd worden. Welke methode zouden we kunnen toepassen?
7. Er is een onderscheid tussen software die als product wordt verkocht via winkels (bijvoorbeeld Microsoft Office, diverse computerspellen, of Davilex Account) en software die als maatproduct aan klanten wordt geleverd. Zijn alle ontwikkelmethoden even geschikt voor deze soorten software?

8. Als software inderdaad bijvoorbeeld binnen negen dagen gerealiseerd zou moeten worden, welke methode raad jij dan aan bij het ontwikkelen?
9. Ontwikkelmethoden zijn tot stand gekomen dankzij het soort applicaties dat er mee wordt ontwikkeld. Als we een indeling van het soort applicaties maken, welke ontwikkelmethodiek is dan voor de volgende applicaties het meest geschikt:
 - a. Berekeningsprogramma voor de massa van hemellichamen.
Een reeks gegevens worden per ster ingevoerd en het programma berekent in een keer de massa's en drukt deze in een lijst af.
 - b. Programma om giroafschriften af te drukken.
 - c. Vluchtreserveringsprogramma bij een reisbureau.
 - d. Reisplannerprogramma van de website www.ov9292.nl.
10. Waarom is er bij de moderne ontwikkelmethoden zoveel aandacht voor testen?

De case PartyCups – functionele requirements in use- cases

2.1 Inleiding

In dit hoofdstuk gaan we kijken naar een bedrijf en gaan we op zoek naar functionele eisen (*requirements*). Na afloop van dit hoofdstuk kunnen we requirements indelen in een drietal categorieën. We passen deskresearch toe en gaan een interview opstellen, afnemen en vastleggen. Daarna verwerken we de resultaten van de vastgelegde interviews tot use-cases. Indien nodig wordt een gebruikersobservatie uitgevoerd. Functionele requirements kunnen we vertalen naar use-cases. Ten slotte kunnen we met behulp van een use-case diagram laten zien welke functionaliteit een (toekomstig) softwaresysteem voor een gebruiker heeft. Voor het maken van de diagrammen gebruiken we het softwarepakket Microsoft Visio Professional 2002.

2.2 De case PartyCups – introductie

PartyCups is een jong bedrijf, gevestigd in het midden van Nederland. Het bedrijf verzorgt de uitrusting voor grote partijen en feesten. Het heeft daartoe contacten met leveranciers van bakers, glazen, servies en bestek. Ook plastic bakers, borden en bestek behoren tot het assortiment. Verder is het mogelijk de bakers, glazen, servies en bestek te huren. De contacten met klanten en leveranciers verlopen tot nu toe via een ‘kaartenbak’ in een Access-database die ooit is gerealiseerd door een medewerker van het bedrijf. Door de groei van

de onderneming dreigt men het overzicht kwijt te raken over bestellingen en oude leveringen. Er zijn al klachten binnen gekomen van klanten over verkeerde leveringen of verkeerde facturen. Afspraken uit het verleden worden soms niet nagekomen omdat ze niet zijn vastgelegd en dus zijn vergeten. Het bedrijf wil graag een eenvoudig klantrelatiebeheersysteem waarin afspraken met klanten kunnen worden bijgehouden. Het pakket moet ook de orderhistorie kunnen laten zien. Een dergelijk pakket wordt vaak een **CRM-systeem** (Customer Relation Management System) genoemd.

De directeur van de onderneming heeft ons gevraagd de ontwikkeling van deze applicatie te verzorgen. Om deze applicatie op te kunnen leveren, gebruiken we een methode voor het ontwikkelen daarvan. Deze methode beschrijft de stappen die ondernomen worden om de ontwikkeling met succes af te ronden. De methode die wordt beschreven is gebaseerd op Rapid Action Development, vaak afgekort tot RAD. Met deze methode kunnen we snel een applicatie opleveren.

2.3 De scope

Het bouwen van een nieuwe applicatie begint in het algemeen met het vaststellen van de **scope**, de grenzen aan het te ontwikkelen systeem. Daarbij brengen we in kaart wie er allemaal bij het te ontwikkelen systeem betrokken zijn. We onderzoeken hoe deze betrokkenen willen werken met het te ontwikkelen systeem. Vooral aan het begin van een ontwikkelingstraject zal dit onduidelijk zijn. Daarom kan er aan het begin van een project geen definitieve begroting worden opgesteld. De functionaliteit die we in een systeem willen realiseren moet uitvoerbaar blijven binnen het budget en de mogelijke hulpmiddelen.

Bij het aanpassen van bestaande systemen gaat het om het verbeteren en uitbreiden van de bestaande functionaliteit. De gewenste aanpassingen zijn meestal wel duidelijk aangegeven, maar de opdrachten zijn soms ook op verschillende manieren uit te leggen.

Het is voor het resultaat heel belangrijk om voortdurend te communiceren met de opdrachtgever. De scope bepaalt immers welke functionaliteit wordt opgenomen in het te realiseren systeem.

In beide gevallen is het belangrijk de scope van het systeem te bewaken. Het is de taak van de softwareontwikkelaar om aan de opdrachtgever aan te geven wat de realisatie van een systeem kost. Alle betrokken partijen weten dan min of meer waar zij aan toe zijn.

2.4 Aanpassingen aan bestaande systemen

Als een systeem uitgebreid of verbeterd moet worden, kan worden afgeleid wie er betrokken zijn bij een nieuwe applicatie door de situatie te bestuderen waarin deze wordt toegepast. Het bestuderen van het gebruik van de oude applicatie kan op verschillende manieren. Het is handig om te beginnen met het bestuderen van de documentatie die er van het bestaande systeem is. Het gaat dan om eventuele systeembeschrijvingen zoals functioneel en technisch ontwerp, gebruikershandleiding en de uitvoer die het systeem realiseert in de vorm van formulieren en rapporten. Het bestuderen van zulk materiaal wordt ook wel *deskresearch* genoemd. We kunnen dit onderzoek nog uitbreiden door het gevonden materiaal uit te breiden met wat er beschreven is over vergelijkbare systemen in vaktijdschriften en boeken.

Na het bestuderen van dit materiaal beginnen we met het opstellen van te houden *interviews* om eventuele opheldering te vragen en om zicht te krijgen op de problemen die er met het oude systeem zijn. Om vast te stellen wie we willen interviewen, moeten we al een beeld hebben van de te ontwikkelen applicatie. Het heeft alleen zin om mensen te interviewen die het systeem gebruiken of profijt moeten hebben van de uitkomsten van het systeem.

Terzijde

Een oud gezegde luidt: de beste stuurder staan aan wal. Iedere vrachtwagenchauffeur zal kunnen vertellen dat magazijnmedewerkers onhandig werken. De chauffeur zal altijd kritiek hebben op de wijze waarop zijn vracht bij de wagen aangeleverd

wordt. Het komt in de verkeerde volgorde, de gewichten van de diverse stukken variëren te veel, de lading is incompleet en ga zo maar door. Het is altijd eenvoudig om kritiek op andermans werk te hebben. Moet de vrachtwagenchauffeur geïnterviewd worden voor een magazijnapplicatie? Waarschijnlijk niet. In het magazijn zijn namelijk heel andere criteria van belang dan bij het transport. Het kan echter ook zo zijn, dat de vrachtwagenchauffeur een wezenlijke bijdrage levert aan het magazijnproces. Hoe dan ook, de softwareontwikkelaar moet vaststellen wie er betrokken zijn bij de functionaliteit die beschreven moet worden. Alle betrokkenen worden actoren genoemd.

Het vaststellen van deze functionele requirements voor elke actor zal niet in een keer gebeuren. Het komt regelmatig voor dat gedurende de ontwikkeling van het systeem nog functionaliteiten worden ontdekt. Voor het vaststellen van de huidige functionaliteit is dit niet erg, integendeel. Juist door ons niet vast te leggen op de huidige functionaliteit houden we vanaf het begin rekening met eventuele veranderingen in de applicatie. De functionaliteiten van een systeem stellen we in overleg met de betrokken actoren vast.

Een lijst met te interviewen personen moet worden opgesteld. Daarbij is het noodzakelijk om ook aan te geven waarom het gewenst is iemand te interviewen. Een aantal actoren kunnen we gemakkelijk benaderen voor een interview, bij andere actoren gaat dit moeilijker. De oorzaak daarvoor kan zijn dat door het grote aantal mensen in de groep het onmogelijk wordt om bij iedereen een interview af te nemen. Soms is het juist een enkel iemand die vanwege zijn of haar specialisme weinig tijd vrij kan maken voor een interview. Het is dan belangrijk om aan te kunnen geven wat het belang van het interview of de interviews is. We kunnen dus het beste al in het kort aangeven welk resultaat we uit elk interview verwachten. Bij het houden van de interviews is een goede voorbereiding een voorwaarde om bruikbare gegevens te verkrijgen. Vooral bij mensen die weinig tijd hebben, kan het storend overkomen om te vragen naar informatie die voor de hand ligt. Voor het houden van de interviews en het selecteren van de geïnterviewden moeten we ons inleven in het op te leveren systeem. Dat kan door de al eerder beschreven deskresearch.

Een andere mogelijkheid voor het vinden van informatie over een bestaand systeem is om een aantal observaties uit te voeren. Daarbij wordt de betrokken gebruiker bekeken tijdens het werken met het te verbeteren systeem. We hebben net als bij het interview wel instemming nodig van de betrokken gebruiker. Daarbij moeten we voor de vergelijkbaarheid tussen verschillende observaties natuurlijk wel een observatieschema opstellen. In een dergelijk schema leggen we vast hoe de **observatie** wordt uitgevoerd en waarop gelet gaat worden. Dus ook hier loont het, ons goed voor te bereiden. We willen immers weten welke functionaliteiten het informatiesysteem aan de gebruiker biedt. Hoe de gebruiker omgaat met het systeem is voor het functioneel ontwerp niet direct van belang. Het kan wel informatie opleveren over irritaties bij het gebruik van het huidige systeem die door het nieuwe systeem vermeden dienen te worden. Deze dienen dus wel genoteerd te worden, maar het belangrijkste zijn de door de gebruiker gerealiseerde taken die met het systeem worden uitgevoerd.

2.5 Nieuwe applicaties

Bij nieuwe applicaties is het vaststellen van de requirements veelal afhankelijk van het beeld dat de opdrachtgever of initiatiefnemer van de nieuwe applicatie heeft. Met hem of haar zullen we een of meer interviews moeten houden om gezamenlijk vast te stellen wie de gebruikers worden van het **nieuw** te ontwikkelen **systeem**. Zo'n gebruiker kan een persoon in de organisatie zijn, maar het kan ook een ander systeem of een andere applicatie zijn. Daarbij moet ook gedurende het ontwikkeltraject steeds beter worden aangegeven welke functionaliteit elke gebruiker stelt aan het nieuwe systeem. Doordat het hierbij echter gaat om een systeem dat nog niet eerder heeft gewerkt, zal er meestal eerst een prototype worden ontwikkeld. Enerzijds om bijvoorbeeld aan te tonen dat het technisch gerealiseerd kan worden en anderzijds om een beter zicht te krijgen om de wensen die de gebruiker aan het systeem stelt.

2.6 Het interview

Bij het houden van de interviews is van belang dat we zoveel mogelijk over dit systeem te weten komen. Het stellen van open vra-

gen levert ons de meeste informatie op. Een gesloten vraag is een vraag waarbij maar een beperkt soort antwoorden mogelijk is. Een dergelijke vraag kan alleen worden beantwoord met ja of nee, of met een waarde of feit. Voorbeelden van gesloten vragen zijn:

- Verkoopt u in deze winkel ijs?
- Wat kost dit boek?

Een open vraag levert echter een heel scala aan antwoordmogelijkheden op. Het is daarbij wel zaak om de geïnterviewde te blijven focussen op het onderwerp waarover we informatie willen vinden. Soms hebben geïnterviewden namelijk hun eigen verhaal dat behoorlijk kan afwijken van wat wij willen weten. Het gaat dus om het afnemen van een *gestructureerd* interview waarbij we de vragen van tevoren hebben geformuleerd. Die vragen hoeven we natuurlijk niet letterlijk te nemen maar zijn leidraad bij de afname van het interview. Voorbeelden van open vragen zijn:

- Kunt u toelichten hoe een verkoop wordt afgehandeld?
- Welke handelingen moeten worden verricht om iemand lid te maken van uw vereniging?

Die informatie moet wel genoteerd worden. Het is lastig om zowel de vragen te stellen als de antwoorden van de geïnterviewde te noteren. Door met twee personen een interview af te nemen wordt dit eenvoudiger. Een alternatief kan zijn om een memorecorder mee te nemen, maar dit wordt niet door iedereen geaccepteerd.

Vaak levert een set interviews weer nieuwe vragen op die later beantwoord moeten worden. Het is daarom aan te raden in de procesplanning een tweede serie interviews op te nemen ter verfijning van de antwoorden.

2.7 De Unified Modeling Language

De scope van een systeem en de gewenste functionaliteiten leggen we vast met behulp van een diagram in *UML*. UML staat voor Unified Modeling Language. Het is de gemeenschappelijke taal om een systeemontwerp te beschrijven en vast te leggen. De eisen aan een te ontwikkelen systeem, de functionele requirements, worden in deze taal vastgelegd met behulp van het *use-case diagram*. Elke

functionele eis die aan het systeem gesteld wordt, plaatsen we in of onder een ovaal, een use-case.

Elke gebruiker die betrokken is bij zo'n use-case wordt aangegeven door een poppetje. De meeste gebruikers zullen het systeem benaderen vanuit hun functie. Deze functie wordt onder het poppetje geplaatst, als naam. Het poppetje zelf wordt actor genoemd.

Figuur 2.1
Use-case 'inloggen'



Figuur 2.2
De actor 'Administratief medewerker'

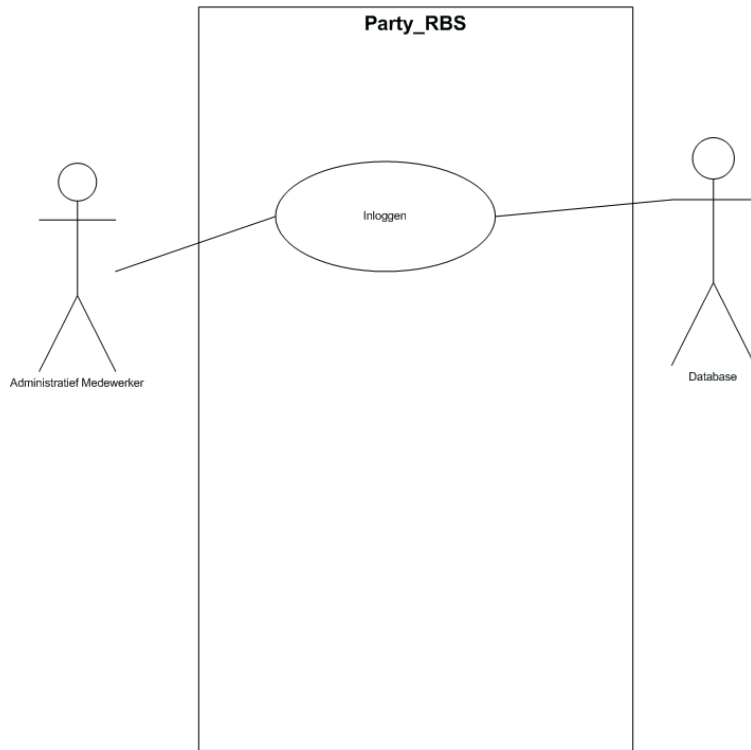


De verzameling van alle **use-cases** en de daarbij behorende **actoren** kunnen in een **diagram** worden getoond. De actoren worden links of rechts geplaatst en de use-cases in het midden. Een rechthoek om de verzameling use-cases heen geeft de systeemgrens van het systeem aan, de scope. De lijnen tussen actoren en use-cases geven aan welke actoren bij welke use-cases betrokken zijn. Het is handig om de actieve actoren, dat wil zeggen de actoren die het systeem in werking stellen, links van de systeemgrens te plaatsen en de passieve actoren (ondersteunende actoren) rechts.

De resultaten vanuit de deskresearch, interviews en observaties worden dus verwerkt tot vast te stellen functionaliteiten, betrokken actoren en een vastgelegde systeemgrens (scope) in de vorm van een use-case diagram met daarbij behorende templates en scenario's.

Figuur 2.3

Het use-case diagram
'Party_RBS (inloggen)'



2.8 Niet-functionele requirements

Naast de functionele requirements komen we tijdens de interviews ook andere requirements tegen. Deze requirements zeggen niets over wat het systeem moet kunnen, maar meer over hoe het systeem dit moet doen of welke beperkingen er voor het systeem gelden (*business constraints*). Bijvoorbeeld in het nieuw te realiseren systeem kan een order pas worden toegevoegd als deze bij een klant hoort. Een dergelijke requirement noemen we een niet-functioneel (*non-functional*) requirement.

Een ander voorbeeld van een niet-functioneel requirement is de tijd die het kost om een functionaliteit te realiseren met het te ontwikkelen systeem. Bijvoorbeeld het toevoegen van de gegevens van een nieuwe klant mag niet meer dan x seconden duren. Gedurende de interviews zullen veel van deze requirements worden genoemd. Het is heel belangrijk om deze goed te noteren, omdat ze eisen

aangeven die de gebruikers stellen aan het te realiseren systeem. Hierbij betekent *goed* noteren vooral *meetbaar* noteren. Zo zullen veel gebruikers het hebben over gebruikersvriendelijk of gemakkelijk. De applicatie moet gebruikersvriendelijk zijn en gemakkelijk in het gebruik. Het is de taak van de interviewer om door te vragen naar wat de gebruiker dan verstaat onder gebruikersvriendelijk of gemakkelijk.

2.9 Pseudo-requirements

Ook een derde vorm van requirements komt voor. Deze hebben zelfs niets te maken met het systeem, maar meer met de organisatie of omgeving waarin het systeem wordt gemaakt. Zo is het niet gebruikelijk om voor elke applicatie een nieuw computersysteem aan te schaffen. Een eis van de organisatie kan bijvoorbeeld zijn dat de applicatie moet kunnen draaien op het door de organisatie gebruikte computersysteem. Ze worden daarom **pseudo-requirements** genoemd. Een andere pseudo-requirement is bijvoorbeeld de eis van de organisatie dat de broncode gerealiseerd moet worden in Java, omdat deze programmeertaal bij de organisatie bekend is.

In het use-case diagram leggen we alleen de functionele requirements vast. In het diagram worden deze zichtbaar in use-cases (een ovaal met een werkwoord). Soms geven we die functionaliteiten aan per actor in aparte use-case diagrammen. Dit om het overzicht te kunnen houden. Functionele eisen leiden we af uit de informatie die de actoren verstrekken. Dat geldt bij al bestaande systemen, waarbij er wel op moet worden gelet dat een onderscheid gemaakt wordt tussen *wat* een systeem doet en *hoe* het dit doet. Het laatste moet immers door het nieuwe systeem worden verbeterd.

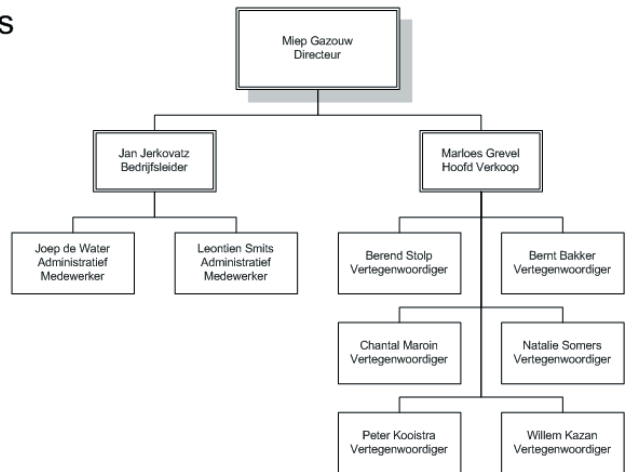
2.10 User stories

Soms is er geen bestaand systeem, of is de functionele eis een uitbreiding op een bestaand systeem en kent de gebruiker de werkwijze nog niet. Dan gaat het erom het beeld dat de gebruiker in gedachten heeft zo nauwkeurig mogelijk te beschrijven en vast te leggen. Ook hier gaat het om de beschrijving die de gebruiker geeft. Deze beschrijving zal vaak concreet zijn, een feitelijke han-

deling beschrijven of het beeld dat de gebruiker in gedachten heeft. Daarom worden dit soms ook gebruikersverhalen genoemd (user stories).

Figuur 2.4
Het organogram van
PartyCups

PartyCups 07/2003



Mevrouw Miep Gazouw van PartyCups heeft in de vier jaar dat haar bedrijf nu bestaat het zien groeien van een eenmanszaak met twee personeelsleden naar een BV met tien personeelsleden. De organisatie van het bedrijf is een stuk ingewikkelder geworden.

Ook al is de boekhouding vanaf het begin uitbesteed, de administratie neemt steeds meer tijd en mankracht in beslag. Vooral het nazoeken van oudere facturen om bepaalde afspraken op te zoeken die in het verleden met klanten zijn gemaakt, bedreigt steeds meer de dagelijkse werkzaamheden. De twee medewerkers op de administratie, Joep en Leontien, werken steeds vaker over en hebben inmiddels aangegeven, dat zij extra hulp nodig hebben. Dat kan overigens ook worden afgeleid uit het feit dat er de laatste tijd meer fouten worden gemaakt. Die moeten dan weer gecorrigeerd worden, wat ook weer extra tijd kost. De vertegenwoordigers in de zes regio's in het land klagen er de laatste tijd over dat de door hen aangeleverde orders niet altijd zorgvuldig worden verwerkt. De orders worden met de hand geschreven aangeleverd, wel

op een vast ingedeeld formulier maar daar wordt soms toch een extra krabbel op gezet. Die is dan niet altijd even leesbaar en door de grote werkdruk op de administratie wordt deze dan of verkeerd, of helemaal niet gelezen.

Miep Gazouw is op de hoogte van de problemen maar denkt dat alleen een extra kracht op de administratie niet de oplossing kan bieden. Ze heeft overlegd met Jan en Marloes, medewerkers die al vanaf het prille begin bij het bedrijf zijn. Ook zij zijn tot de conclusie gekomen dat er voor een andere oplossing moet worden gekozen. Zowel bij Jan als Marloes is het idee opgekomen om door automatisering een oplossing te realiseren voor het huidige probleem. Het administratieve pakket biedt volgens hen onvoldoende mogelijkheden om aan relatiebeheer te doen. Zij willen graag een eigen oplossing gerealiseerd zien. Die oplossing zou dan zowel door de administratie als door de vertegenwoordigers gebruikt moeten kunnen worden. Het systeem dat zij in gedachten hebben wordt ook wel een CRM-systeem genoemd, een pakket waarin de relatie met de klant kan worden beheerd.

2.11 Van **user story** naar **use-case diagram**

Om vast te stellen wie de gebruikers worden van het nieuwe systeem, bestuderen we de opdrachtformulering en de documenten die het bestaande of gewenste systeem beschrijven. Bij PartyCups hebben we te maken met een opdrachtgever, Miep Gazouw, die eigenlijk alleen voor de opdracht zorgt en uiteraard de betaling goedkeurt. Daarnaast is zij slechts zijdelings betrokken bij het gebruik van de applicatie. Vanuit het management van het bedrijf zijn vooral Jan en Marloes betrokken, zij zijn de eigenlijke initiatiefnemers. Ten slotte zijn er twee gebruikersgroepen zichtbaar die voornamelijk de invoer van gegevens verzorgen, de administratieve medewerkers en de vertegenwoordigers. Er is ook een niet menselijke gebruiker, hoewel vaag beschreven, namelijk het al in gebruik zijnde administratieve pakket.

Het komt voor dat tijdens het vaststellen van de requirements van het systeem nog andere gebruikers of (hulp)systemen ontdekt worden. Die worden dan toegevoegd aan het model. Het model is de blauwdruk van het te ontwerpen systeem. Het beschrijft in grote lijnen wat de te realiseren applicatie moet kunnen en wie daar gebruik van maakt. Daarom worden alleen gebruikers of systemen die functionaliteit gebruiken of geleverd krijgen door het systeem in onze beschrijving van het systeem opgenomen.

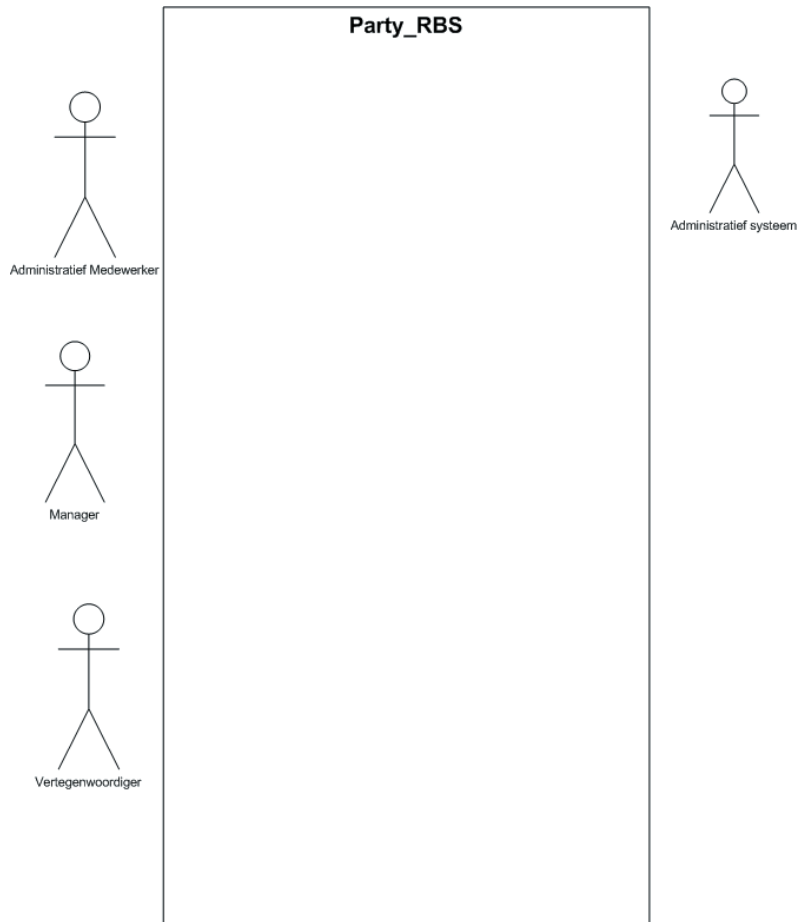
Zo zal Miep Gazouw wel in de documentatie als opdrachtgever voorkomen maar niet meer in de verdere documenten die de te ontwikkelen applicatie beschrijven. Om de grens van een systeem visueel vast te leggen, maken we gebruik van een diagramtechniek uit de Unified Modeling Language, het use-case diagram. Marloes heeft voorgesteld om de te ontwikkelen applicatie Party_RBS te noemen, en dat is door de anderen geaccepteerd. Het voorlopige systeem zonder use-cases ziet er als volgt uit:

Door het vaststellen van de systeemgrens (de rechthoek in het diagram) en de betrokken partijen (de poppetjes) bij het systeem wordt het eenvoudiger om de mogelijkheden van het systeem te beschrijven. We noemen die mogelijkheden de *functies* die het systeem voor elk van de betrokken gebruikers moet hebben. De betrokken gebruikers worden de *actoren* van het systeem genoemd. De functionaliteiten die elk van de actoren door het systeem geleverd krijgt, noemen we *use-cases*. Deze zijn in dit use-case diagram (figuur 2.5) nog niet opgenomen. Met elke use-case wordt een functionele eis (functional requirement) beschreven.

2.12 Use-case templates

Tijdens het interview zei Joep: “Bij het toevoegen van een klant voer ik de naam in en de adresgegevens, bovendien noteer ik de contactmogelijkheden, telefoon, mobiele telefoon, fax en e-mailgegevens. En in het notitieveld de ordergegevens, die voeg ik vooraan in. De datum waarop ik de invoer doe, zet ik voor de ordergegevens, zodat in het notitieveld steeds de laatste gegevens voorop staan, voorzien van datum kenmerk. Als de gegevens niet leesbaar zijn, voer ik de gegevens die ik heb van de klant wel in, maar dan leg ik het

Figuur 2.5
Het use-case diagram
van Party_RBS met
alleen actoren en
systeem



formulier apart om te controleren bij de vertegenwoordiger. Oh ja, de vertegenwoordiger leg ik ook vast bij de klant, elke klant heeft een vaste vertegenwoordiger. Dit leg ik vast door een regiocode toe te kennen aan de klant.”

Een functionele eis die hieruit kan worden gehaald is in ieder geval dat we in staat zijn om een klant toe te voegen in het systeem. De use-case wordt ‘Toevoegen klant’ genoemd en het gebruikersverhaal van Joep vertalen we naar een beschrijving van deze use-case. Zo’n beschrijving van een use-case wordt vastgelegd in een use-case template, een vaste tabel waarin de gegevens van de use-case worden vastgelegd.

| | |
|------------------------|---|
| Naam | Van de use-case |
| Versie | Welke revisie dit is |
| Actor | Betrokken actoren, initiatief nemende actor eerst |
| Preconditie | Beschrijft status van systeem bij aanvang |
| Beschrijving | Succesvol verloop van use-case, beschrijft in stappen het verloop |
| Uitzonderingen | Uitkomsten van een niet succesvol verlopen use-case |
| Niet-functionele eisen | Beperkingen (constraints) en overige eisen aan de use-case |
| Postconditie | Beschrijft status van systeem na afloop |

Tabel 2.1 De use-case template

Met behulp van de use-case template wordt een use-case beschreven. Hierdoor zijn we in staat om aan te geven welke functionaliteit het systeem voor de betrokken actor gaat realiseren. Ook geven we aan aan welke voorwaarden voldaan moet zijn alvorens de use-case kan worden opgestart. Alle handelingen die de betrokken actoren moeten uitvoeren worden erin beschreven en eventuele uitzonderingen met oplossingen worden aangegeven. Ten slotte wordt het resultaat vermeld na afloop van de use-case.

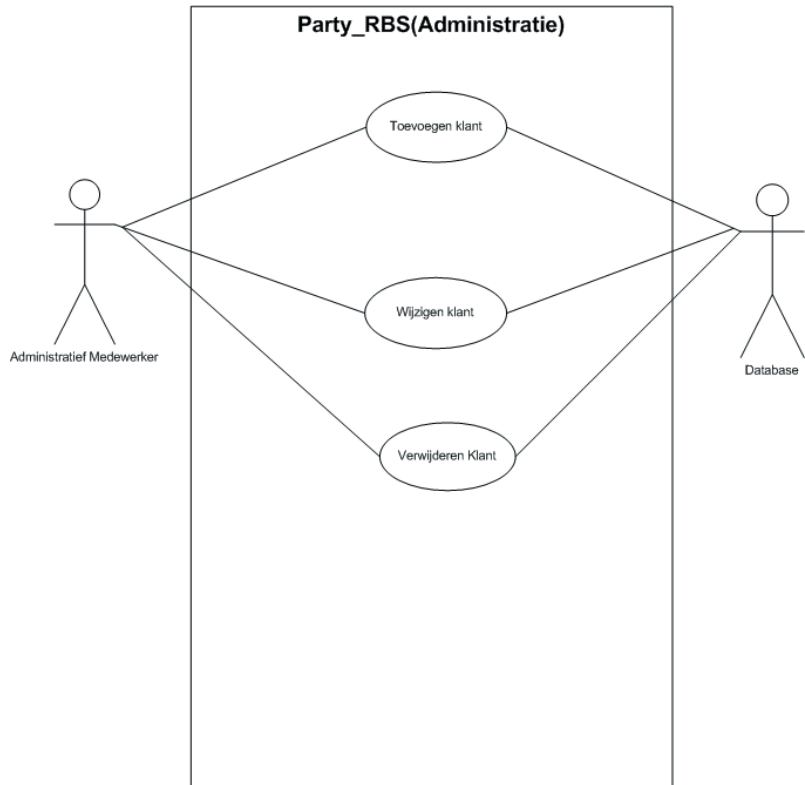
| | |
|------------------------|---|
| Naam | Toevoegen klant |
| Versie | 1.0 |
| Actor | Administratief medewerker, database |
| Preconditie | Medewerker is ingelogd, systeem is beschikbaar voor invoer |
| Beschrijving | <ol style="list-style-type: none"> 1 Medewerker selecteert klant toevoegen 2 Voert naam van klant in 3 Voert adresgegevens klant in 4 Voert regiocode klant in 5 Voert eventueel ordergegevens klant in 6 Drukt op OK-knop om klant toe te voegen, systeem toont melding 'Klant toegevoegd' |
| Uitzonderingen | <p>Klant al bekend in systeem, melding 'Klant komt al voor', terug naar eerste scherm</p> <p>Klantgegevens incompleet, melding gegevens incompleet, aangeven welke gegevens ontbreken, terug naar invoer scherm</p> |
| Niet-functionele eisen | Invoer klantgegevens moet te realiseren zijn in vijf minuten |
| Postconditie | Klant is toegevoegd aan systeem, systeem wacht op actie gebruiker |

Tabel 2.2 De use-case template 'Toevoegen klant'

Door het toevoegen van enkele use-cases en het weglaten van enkele actoren ziet het use-case diagram van het Party_RBS er als volgt uit:

Figuur 2.6

Het use-case diagram
Party_RBS, subsysteem
administratie



Dit use-case diagram geeft aan dat een administratief medewerker met het subsysteem administratie van Party_RBS een klant kan toevoegen, wijzigen of verwijderen uit de database. Meer informatie over elke functionaliteit vinden we terug in de use-case template van de desbetreffende use-case.

2.13 Samenvatting

Voordat we aan de systeemontwikkeling kunnen beginnen, zullen we op onderzoek moeten gaan. Deze zogenaamde deskresearch leidt ertoe, dat we interviews moeten gaan houden met diverse

functionarissen in de organisatie. Eventueel zullen we ook de werkzaamheden moeten observeren. Zo brengen we de huidige en de gewenste bedrijfsprocessen in kaart.

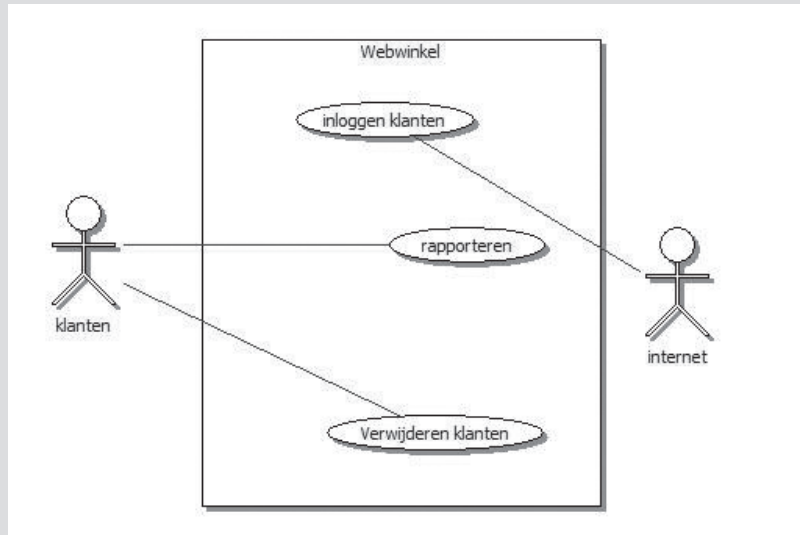
Het use-case diagram dient om functionaliteiten vast te leggen. Binnen de rechthoek (de systeemgrens) worden de use-cases geplaatst. Elke use-case bevat een functionaliteit, er gebeurt dus iets. Wat er precies gebeurt, staat in de use-case template. Daarin zijn ook de voorwaarden voor het gebeuren opgenomen, de uitzonderingen en de speciale gevallen. Elke use-case wordt aangestuurd door minstens één actor. Een actor neemt dus het initiatief om de gebeurtenis te laten beginnen. Deze actor, de actieve actor, plaatsen we bij voorkeur aan de linkerkant van het systeem. Er zijn ook actoren die profijt hebben van of behulpzaam zijn bij de gebeurtenis. Deze actoren, de passieve actoren, bevinden zich aan de rechterkant van het systeem. Sommige actoren zijn soms actief, soms passief en kunnen dus zowel aan de linkerkant als aan de rechterkant van het systeem voorkomen.

Als de use-case diagrammen klaar zijn, moeten we ze met de diverse betrokkenen bij de klant doorspreken. Alleen zo komen we erachter of onze denkbelden overeenkomen met die van de klant.

Opgaven

1. Geef een voorbeeld van een gesloten en open vraag over Party_RBS die we aan Joep kunnen stellen.
2. Vul de use-case templates in voor de use-cases 'Wijzigen klant' en 'Verwijderen klant'.
3. Geef aan hoe een observatie moet worden uitgevoerd bij een van de vertegenwoordigers van PartyCups.
4. Geef aan welke fouten zitten in het diagram van figuur 2.7.

Figuur 2.7
Een use-case diagram
met fouten



5. Geef het type aan van onderstaande requirements:
 - a. Wijzigen van klantgegevens.
 - b. Het systeem moet met behulp van Java gebouwd worden.
 - c. Inloggen.
 - d. Toevoegen van product.
 - e. Van een klant moet altijd de achternaam bekend zijn.
 - f. Het afdrukken van een klantrapport mag niet langer dan 3 minuten duren.
6. Bij een schoolbibliotheek worden boeken aan leerlingen en docenten uitgeleend. De uitleentermijn voor leerlingen is een maand, voor docenten is deze een jaar. Aan het eind van de uitleentermijn ontvangt de lener een herinnering. Elke lener krijgt een pas, zodra deze bij de school is toegelaten of in dienst getreden.
 - a. Geef aan welke actoren er in de tekst van deze opgave staan.
 - b. Welke actor/actoren ontbreken in de tekst van deze opgave?
 - c. Geef aan welke use-cases in de tekst van deze opgave aanwezig zijn.
7. Een diëtiste heeft een programma dat een lijst met producten toont en, na selectie van een product, een aantal gegevens over het product (zoals aanwezige mineralen en vitaminen, de hoeveelheid

calorieën, de hoeveelheid koolhydraten en de hoeveelheid vet). De lijst met producten kan door de diëtiste zelf worden uitgebreid met nieuwe, met de daarbij behorende gegevens. De diëtiste kan ook gegevens bij een product wijzigen of een product verwijderen. Maak het use-case diagram met daarin de verschillende use-cases. Werk de use-case ‘Toevoegen product’ uit in een use-case template.

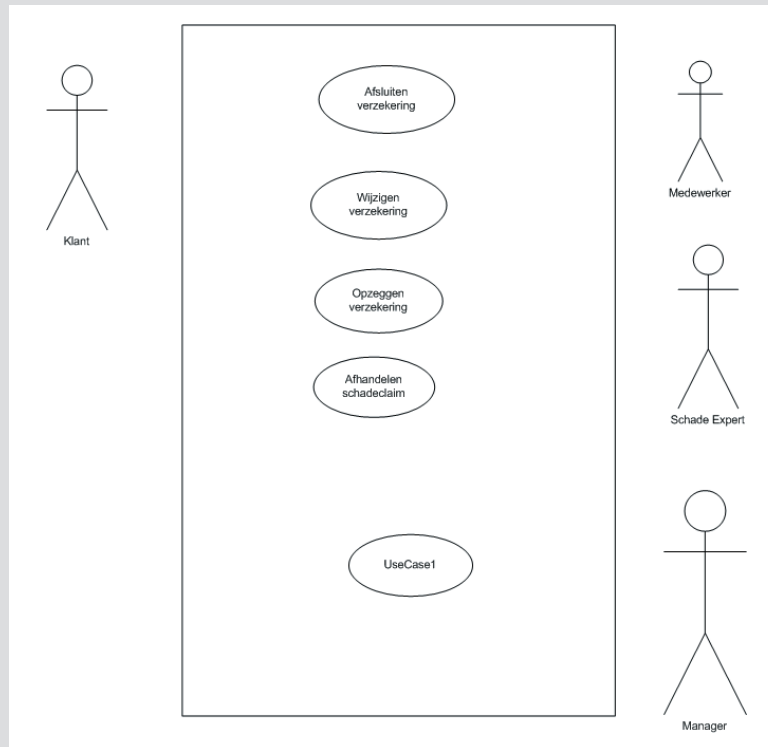
8. Ga naar het dichtstbijzijnde station en controleer of onderstaande use-case template ‘Verkoop reisbiljet’ juist is of dat er nog een aantal zaken zijn vergeten. Voeg indien nodig de ontbrekende zaken toe.

| | |
|-----------------------|--|
| Naam | Verkoop reisbiljet |
| Versie | 0.5 |
| Actor | Klant, Lokale database |
| Preconditie | Systeem wacht op actie van klant |
| Beschrijving | 1 Klant selecteert bestemming 2 Kiest type kaart (enkel of retour) 3 Kiest klasse (1e of 2e) 4 Kiest geldigheid (met of zonder huidige datum) |
| Uitzonderingen | Klant heeft geld te weinig, apparaat reset zichzelf na twee minuten. |
| Non-functionele eisen | Kaartje moet in drie minuten verkocht kunnen worden |
| Postconditie | Klant heeft juiste reisbiljet, systeem wacht op nieuwe actie |

9. Maak, zonder dat je verdere gegevens hebt, het use-case diagram van figuur 2.8 af, zoals dit volgens jou (en volgens de regels van UML) moet.
10. Hotel Amsterdam
 Hotel Amsterdam stelt een aantal kamers beschikbaar aan reisorganisatie Vermeer om verder door te verhuren, uiteraard tegen een vaste jaarlijkse vergoeding. Het hotel heeft daarnaast een aantal kamers waarvoor het zelf rechtstreeks de bezetting regelt. Met Vermeer is afgesproken dat, mochten hotelkamers die Vermeer in de verhuur heeft niet bezet worden, het hotel deze kan gebruiken als de eigen kamers volgeboekt zijn. Deze kamers komen pas een week voor de actuele datum vrij voor eventuele boeking.

Figuur 2.8

Een incompleet use-case diagram voor verzekeringen



Het reserveren van hotelkamers kan door Vermeer gebeuren of bij het hotel zelf. De reserveringen die Vermeer afsluit, worden direct doorgegeven aan het hotel in verband met de bezetting en maaltijden. De reserveringen kunnen binnenkomen per brief, telefoon of e-mail. De volgende gegevens worden daarbij steeds vastgelegd:

- reserveringsnummer,
- naw-gegevens (naam, adres, woonplaats) van de klant,
- klantcode,
- datum van de reservering,
- volgnummer (indien meerdere kamers voor deze klant worden ge-reserveerd anders gevuld met 0),
- datum aankomst,
- datum vertrek,
- kamertype (een/twee/vierpersoons, met bad/douche),
- boekingstype (met ontbijt, met lunch, met diner of combinaties).

Per kamer kunnen afhankelijk van het type kamer de gastnamen worden ingevuld. Deze namen hoeven niet overeen te komen met

de klantnaam. De klanten waarmee het hotel werkt, zijn immers vaak bedrijven of reisorganisaties waaronder reisbureaus of organisaties zoals Vermeer. Indien het om een particuliere klant gaat die nog niet eerder zaken heeft gedaan met het hotel, worden ook creditcardgegevens van de klant vastgelegd.

Alle reserveringen worden in volgorde van binnenkomst door de receptie van het hotel afgehandeld. Overdag wordt elk uur de e-mail geraadpleegd of er reserveringen zijn binnengekomen. De reserveringen worden vastgelegd met de reserveringsadministratie. Het reserveringsoverzicht kan daarbij steeds worden geraadpleegd. Dit overzicht geeft aan welke kamers vrij zijn in een bepaalde periode van een bepaald kamertype. Als de reservering akkoord is, wordt een reserveringsbevestiging naar de klant gestuurd. Als er geen kamers van het gewenste type in een periode beschikbaar zijn, wordt dit aan de klant medegedeeld (dit kan zowel telefonisch, fax of per e-mail).

Op de dag van aankomst meldt de gast zich bij de receptie van het hotel om in te boeken. Op dat moment wordt aan de gast een kamernummer toegewezen, tenzij dit al bij de reservering heeft plaats gevonden. De inboeking wordt vastgelegd in de reserveringsadministratie.

Door het inboeken wordt er voor de gast ook een rekening geopend waarmee hij of zij in het hotel allerlei voorzieningen kan gebruiken (telefoon, restaurant, minibar, consumpties in de bars van het hotel). Het bijwerken van de gastenrekening gebeurt aan het eind van elke dag, op basis van de diverse uitgeschreven nota's per kamernummer die ondertekend dienen te zijn door de klant (uitgezonderd telefoon en minibar).

Komen gasten niet op de afgesproken aankomstdag aan, dan vervalt de reservering, tenzij op de dag van aankomst bericht van vertraging is ontvangen. Als er geen bericht is binnengekomen en de reservering is opgeheven, wordt de kamer opnieuw in de verhuur opgenomen. De klant die de reservering heeft afgesloten krijgt een boetebedrag berekend dat afhankelijk is van de reserveringslengte (in dagen) en de mogelijke verhuur van de kamer(s) aan anderen.

Op de dag van vertrek meldt de gast zich weer bij de receptie om uit te boeken. Na betaling van de factuur (contant of per creditcard), wordt de gastenrekening afgesloten en overgebracht naar de historische gegevens. Aan het einde van elke maand worden rapportages afgedrukt op basis van deze historische gegevens.

De volgende activiteiten binnen het hotel zijn te onderscheiden:

- *Afhandelen reservering.* Het ontvangen van de reserveringsaanvraag. Het nagaan of de reservering kan worden vastgelegd. Indien dit kan, dan een bevestiging sturen, anders een mededeling doen. Verwerken van klantgegevens, eventuele controle creditcard.
- *Wijzigen van reserveringen.* Het ontvangen van een reserveringswijziging. Het nagaan of de reservering bestaat en of de wijziging kan worden doorgevoerd. Als dit kan, wordt een herziene reserveringsbevestiging opgestuurd. Als het niet kan, wordt dit aan de klant medegedeeld.
- *Inboeken gasten.* Bijwerken van de gegevens, controle of gastgegevens juist zijn vastgelegd. Visuele controle creditcard. Toewijzing van kamers en overhandiging sleutel(s). Verwerking van vervallen reserveringen.
- *Bijwerken gastenrekening.* Bij boeken van de gastenrekeningen met behulp van de overzichten van de telefoon- en minibarlijsten en de ondertekende consumptienota's van het restaurant en de bars.
- *Uitboeken van de gast en financiële afhandeling.* Een vervallen reservering moet worden omgezet in een boetefactuur aan de klant die de reservering heeft geregeld. De gastenrekening die wordt afgesloten moet worden overgeheveld naar het historische bestand.
- *Overzichten opstellen.* Aan het einde van iedere maand worden rapporten uitgedraaid met daarin overzichten

Werk het use-case diagram uit van het reserveringssysteem voor Hotel Amsterdam.

3.1 Inleiding

In hoofdstuk 2 is de onderneming PartyCups geïntroduceerd. Daar hebben we geleerd hoe we functionele requirements kunnen vinden en vastleggen in use-cases en een use-case diagram. Door een systeem *incrementeel* op te leveren kan de ene use-case al gerealiseerd zijn, terwijl de andere nog in een documentatiestaadium verkeert. Zo'n incrementele oplevering van een systeem is een vorm van *iteratief* ontwikkelen. Door iteratief te ontwikkelen kunnen we gedurende het ontwikkeltraject steeds beter de wensen van de klant realiseren en een beter systeem opleveren. Gedurende de ontwikkeling van het systeem is de inbreng van de klant dan ook een vereiste, omdat deze de ontwikkelaar van feedback over de applicatie tot nu toe dient te voorzien. Als deze feedback door gebrekkige communicatie niet goed verloopt, is dit een ernstige bedreiging voor de realisatie van het te ontwikkelen systeem.

3.2 Het activiteitendiagram

Door de systeemontwikkeling iteratief aan te pakken moet bij de uitwerking van de use-cases ook worden bijgehouden aan welke versie van een use-case wordt gewerkt. Alle documenten worden daarom voorzien van een versieaanduiding, zodat zichtbaar is wat de laatste versie van een document is. Een risico bij het iteratief ontwikkelen is dat we te snel use-cases beschrijven die we achteraf moeten wijzigen of zelfs verwijderen. Daarom is het goed om het gehele systeem en de plaats daarvan in de organisatie te beschrijven. Daartoe maken we gebruik van een andere diagramtechniek uit UML (Unified Modeling Language), het **activiteitendiagram**.

Een activiteitendiagram is een diagram waarmee we opeenvolgende activiteiten van een proces in een organisatie kunnen weergeven. De eerste handeling die we moeten uitvoeren bij het realiseren van een activiteitendiagram is het vaststellen van het te beschrijven (deel)proces van de te bestuderen organisatie. Als dat te beschrijven proces is vastgesteld is de volgende stap het vastleggen van de betrokken actoren. Dit zijn inderdaad de gebruikers die met het te beschrijven proces te maken hebben.

Figuur 3.1

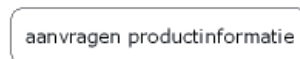
Een swimlane van een activiteitendiagram



Elke actor krijgt in het activiteitendiagram een verticale strook toegewezen, waarin de activiteiten worden geplaatst die de betreffende actor dient uit te voeren of waarvoor deze verantwoordelijk is. Een dergelijke strook wordt in het Engels een *swimlane* genoemd, naar analogie van de zwembaan bij het wedstrijdzwemmen. Boven in de strook staat de naam van de betreffende actor.

Figuur 3.2

Een activiteit in activiteitendiagram



De activiteiten die elke actor uitvoert, zijn aangegeven als een rechthoek met afgeronde zijkanten (zie figuur 3.2). De opeenvolgende activiteiten worden verbonden met pijlen die de overgang van de ene naar de andere activiteit aangeven (de *flow*, zie figuur 3.3). Door het op deze manier aangeven van de activiteiten van een proces, wordt ook de rol die de applicatie moet uitvoeren duidelijk-

ker. De plaats van het te ontwikkelen systeem in het proces kan op deze manier zichtbaar worden gemaakt. Bovendien zien we ook wat de relatie is tussen use-cases en activiteiten van een proces. Ten slotte worden ook de activiteiten zichtbaar die zonder de applicatie moeten worden uitgevoerd. Een proces begint altijd met een startsymbool en kan eindigen met één of meer stopsymbolen.

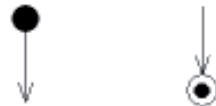
Figuur 3.3

Het flowsymbool in een activiteitendiagram



Figuur 3.4

Het startsymbool (links) en het stopsymbool in een activiteitendiagram



In een activiteitendiagram kan door middel van een zogenaamde **synchronisatiebalk** (*synchronization bar*) worden aangegeven of activiteiten gelijktijdig kunnen worden uitgevoerd.

Figuur 3.5

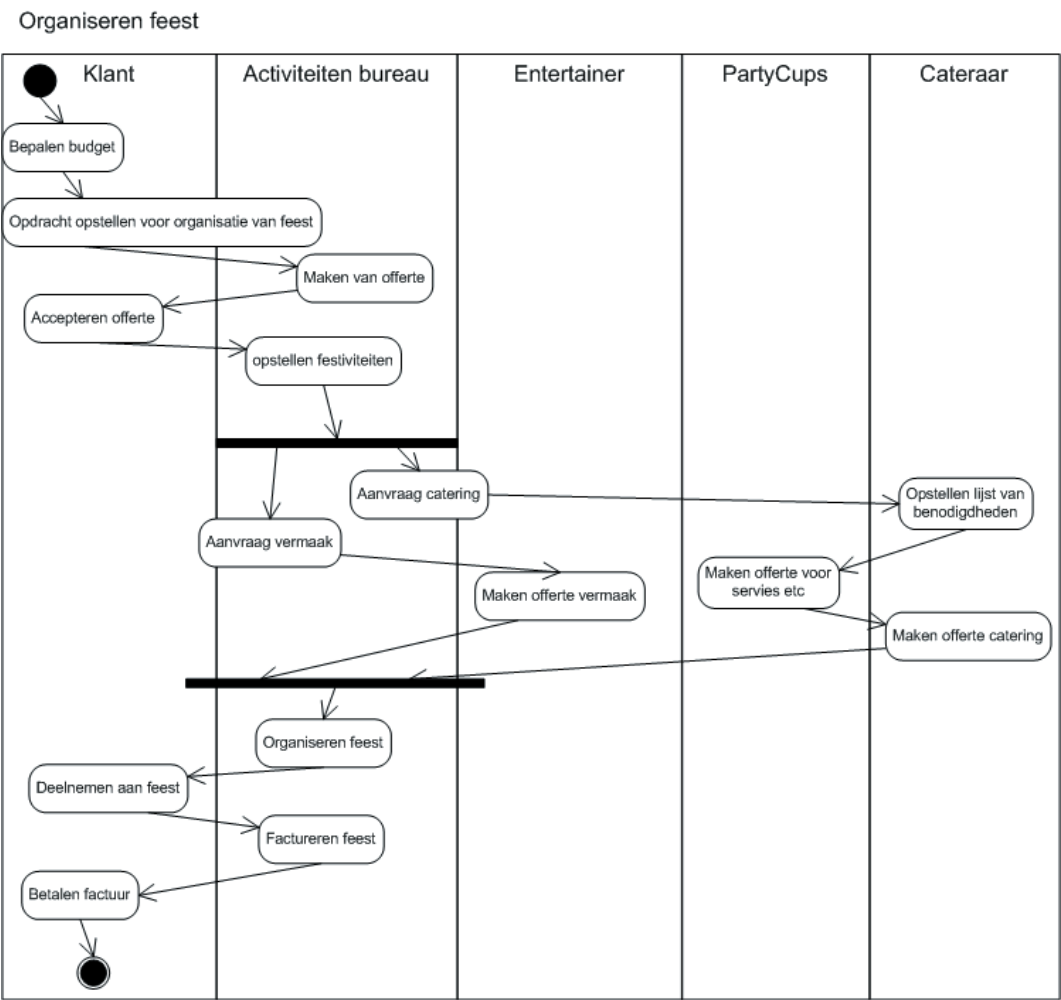
De synchronisatiebalk in een activiteitendiagram



Bij het organiseren van een feest is het gebruikelijk om eerst een budget vast te stellen. Daarnaast is het bepalen van wie wordt uitgenodigd een activiteit. Ook het vaststellen van de catering, het soort vermaak en de locatie horen bij het organiseren van een feest. Soms wordt ook een thema aan een feest gegeven. De locatie is vaak weer afhankelijk van het aantal deelnemers enzovoort. Als we zelf een informeel feest organiseren worden de activiteiten meestal verdeeld over de organisatoren. Bij meer formele feesten zijn verschillende activiteiten uitbesteed aan functionarissen of organisaties. In alle gevallen is het belangrijk om een goede afstemming te krijgen tussen de betrokkenen.

Stel dat een klant een groot feest wil organiseren in verband met een bedrijfslustrum. Zij huurt een activiteitenbureau in om het feest te realiseren. Het activiteitenbureau zal geregeld overleg hebben met de klant, maar ook met allerlei leveranciers. Enkele

daarvan zijn een cateraar (hapjes en drankjes), een leverancier van feestbenodigdheden zoals serviesgoed (bijvoorbeeld PartyCups), een entertainmentbedrijf en/of muziekgroep. Al deze actoren voeren een aantal activiteiten uit in relatie tot het feest. In een activiteitendiagram gaat dat er bijvoorbeeld zo uitzien:



Figuur 3.6 Het activiteitendiagram ‘Organiseren feest’, uitgebreide versie

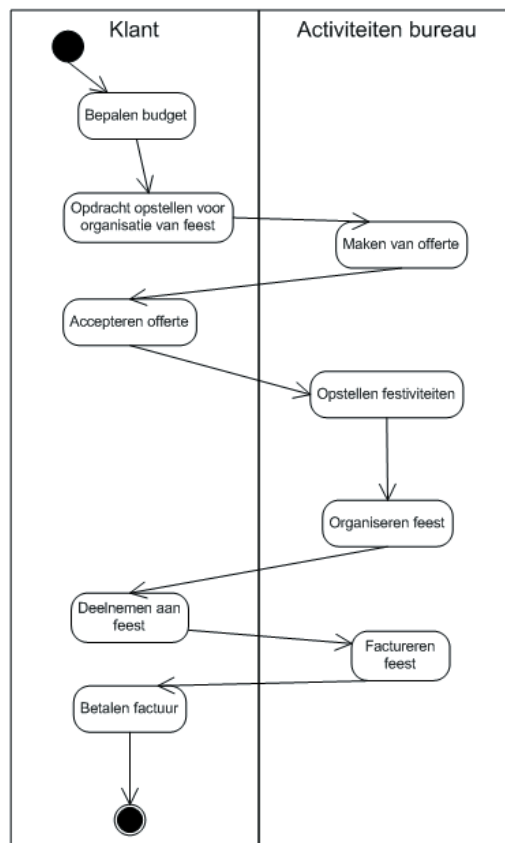
Het valt natuurlijk op dat een activiteitendiagram snel groter wordt, zodra er meerdere actoren bij een proces betrokken zijn. Daarom zullen we vaak het diagram splitsen en de verschillende deelprocessen met elk een eigen activiteitendiagram beschrijven.

Vaak wordt dan een overzichtsdiaagram gemaakt van de hoofdfuncties van de organisatie. De zwarte balk wordt gebruikt om activiteiten die gelijktijdig plaats vinden aan te geven. Uit het diagram kunnen we afleiden dat de activiteiten van entertainer en cateraar parallel kunnen verlopen.

Uit het diagram valt gemakkelijk af te lezen dat bepaalde actoren eigenlijk alleen contact hebben met elkaar. Hierdoor kunnen we een vereenvoudiging aanbrengen. Een veel beknoptere versie van het organiseren van een feest maakt gebruik van het feit dat de klant gebruik maakt van het activiteitenbureau om het feest te organiseren. Door het gebruik van dit bureau wordt het organiseren van een feest voor de klant heel eenvoudig, er is nog maar contact met één organisatie.

Figuur 3.7
Het activiteitendiagram
'Organiseren feest',
beknopte versie

Organiseren feest



Met behulp van een activiteitendiagram kunnen we het proces ‘Organiseren feest’ verder bestuderen. Zo valt op dat het opstellen van festiviteiten na de goedkeuring van de offerte plaats vindt. Mogelijk gebeurt dat inderdaad zo, maar een klant zal toch graag vooraf weten hoe het feest georganiseerd gaat worden. Een terugkoppeling met een extra activiteit bij de klant moet dan in het diagram worden opgenomen. Een activiteitendiagram is dus net als het use-case diagram een hulpmiddel bij de communicatie met de klant.

3.3 De case ‘PartyCups’ – de organisatie

Bij de start van PartyCups werd het werk door drie mensen gedaan. Ook toen was er echter al een taakverdeling. Zo werden de grotere klanten met vaste jaarcontracten door Miep Gazouw geholpen, terwijl de kleinere accounts door Marloes Grevel werden verzorgd. Jan Jerkovatz deed in die tijd de administratie. In die tijd ontstond ook het idee van het halfjaarlijkse PartyCupsfestijn. In het voor- en najaar wordt inmiddels volgens traditie door PartyCups zelf een feest georganiseerd voor haar medewerkers en vaste klanten. Tijdens dit feest worden de meeste productintroducties door leveranciers van PartyCups aan de klanten gepresenteerd. Oorspronkelijk leverde PartyCups alleen de plastic wegwerpbekers, bestek en borden. Inmiddels is het leveringspakket enorm uitgebreid. PartyCups kan alle ‘hardware’-benodigdheden voor een feest leveren.

Het productassortiment bestaat tegenwoordig uit wegwerpartikelen (bekers, bestek, borden, servetten) al of niet met opdruk, verhuur van aardewerkervies, glaswerk, bestek, tafels, stoelen, afdekkleden, opdienschalen, barbecuematerialen, grote barbecue's, partytenten en feestartikelen, waaronder slingers, ballonnen, confetti, maskers. Alle producten kunnen het hele jaar door worden besteld, hoewel een aantal producten duidelijk seizoensgebonden zijn.

In het begin bestond de klantenkring uitsluitend uit grotere evenementenorganisaties die voornamelijk wegwerpbekers

bestelden. In die tijd waren die nog niet voorzien van een opdruk. PartyCups kreeg deze bekervormen geleverd door het productiebedrijf PlasticStans. In overleg met enkele klanten is het idee ontstaan om ook ander wegwerpmateriaal te laten maken en leveren. Zo kunnen feestgangers niet alleen van drank, maar ook van (warme) hapjes worden voorzien. In het assortiment zijn inmiddels naast de bekervormen ook borden en bestek opgenomen. Nog altijd is PlasticStans de huisleverancier voor het plastic wegwerpmateriaal van PartyCups. Naast het plastic wegwerpmateriaal is ook kartonnen materiaal in het assortiment opgenomen.

De klantenkring van PartyCups is inmiddels heel divers. Nog steeds zijn de grotere evenementenorganisaties klant bij PartyCups. Daarnaast zijn de cateraars een belangrijke doelgroep geworden, terwijl ook veel wordt geleverd aan winkels voor feestartikelen. De laatste tijd komen ook veel bars en discotheken spontaan informeren naar de herbruikbare plastic bekervormen. Het land is inmiddels verdeeld in een zestal regio's. Elke regio heeft een eigen vertegenwoordiger met als taak de bestaande klantenkring te onderhouden en nieuwe klanten te werven.

De meeste klantcontacten lopen via de vertegenwoordigers. Ook zijn er klanten die telefonisch of schriftelijk via het kantoor in contact komen met PartyCups. Zodra een klant is opgenomen in het klantenbestand, wordt hij toegewezen aan een van de vertegenwoordigers. Bij bestellingen die spontaan bij PartyCups binnenkomen, wordt geen provisie berekend voor de vertegenwoordiger, in alle andere gevallen krijgt de vertegenwoordiger voor zelf aangeleverde orders 2% provisie en over de andere orders vanuit het klantenbestand van de vertegenwoordiger 1%.

De werkweek van een vertegenwoordiger bestaat grotendeels uit het bezoeken van klanten. Het streven is om bestaande klanten ten minste tweemaal per jaar te bezoeken. Dit wordt op dit moment niet altijd gehaald, omdat het soms onduidelijk is wanneer het laatste contact is geweest, of omdat het inplannen van

het bezoek vergeten wordt. Daarnaast moet de vertegenwoordiger natuurlijk ook nieuwe contacten bezoeken. Die nieuwe contacten kunnen door de vertegenwoordiger zelf worden benaderd of zij zijn op eigen initiatief naar PartyCups gekomen. Ook worden potentiële klanten (leads) via het kantoor doorgegeven naar vertegenwoordigers.

Bij een klantbezoek wordt de catalogus met de producten van PartyCups doorgenomen. De catalogus is ingedeeld in productgroepen: wegwerpartikelen, herbruikbaar plastic, serviesverhuur, meubilair (hieronder vallen ook de tenten en barbecues) en feestartikelen. Zoals al eerder aangegeven, worden de bestellingen genoteerd op een orderformulier. Elke week worden deze orderformulieren afgeleverd op het hoofdkantoor, waarna ze worden verwerkt. Orderformulieren die door de administratie niet verwerkt kunnen worden omdat ze onvolledig of onleesbaar zijn, worden door de administratie terzijde gelegd. Pas na telefonisch contact met de vertegenwoordiger worden deze of aangevuld of verbeterd en alsnog verwerkt.

Na verwerking krijgt de klant een orderbevestiging die ondertekend teruggefaxt moet worden, daarna wordt de order uitgevoerd. Eventuele productiebestellingen worden geplaatst bij de desbetreffende leveranciers. De vertegenwoordiger krijgt elke maand een overzicht van de behaalde provisie en gerealiseerde orders (orders ondertekend door de klant). De looptijd van een order beslaat al gauw een paar weken. Als een klant niet ondertekent, is het de bedoeling dat de vertegenwoordiger de klant telefonisch benadert. Dit wordt soms ook vergeten waardoor een aantal orders niet worden uitgevoerd.

Ten slotte behoort de vertegenwoordiger elk half jaar een overzicht te krijgen van de klanten en hun geplaatste orders. Daarnaast wordt een overzicht gemaakt van klanten die het laatste halfjaar niets hebben besteld. Met behulp van deze lijsten stelt de vertegenwoordiger weer een bezoekschema op voor het komende half jaar. Bovendien geeft de vertegenwoordiger aan, welke klanten uitgenodigd moeten worden voor het halfjaarlijkse PartyCupfestijn.

De inkoop van bestaande producten uit het assortiment wordt door Jan Jerkovatz verzorgd. Nieuwe producten die in het assortiment kunnen worden opgenomen worden gezamenlijk met Miep Gazouw en Marloes Grevel bekeken. De uiteindelijke beslissing voor assortimentsuitbreiding ligt bij Miep Gazouw. Bij de nieuwe productintroductions op het halfjaarlijkse PartyCupfestijn worden ook flyers gemaakt met informatie. Daarna nemen de vertegenwoordigers deze flyers mee daarna naar hun klanten, samen met de nieuwe catalogi.

Het samenstellen van de halfjaarlijkse productcatalogus van het bedrijf is de verantwoordelijkheid van Marloes Grevel. Elk half jaar begint met het aanschrijven van alle leveranciers om nieuwe informatie te verkrijgen. Prijzen worden op aparte inlegvellen vastgelegd zodat deze maandelijks kunnen worden bijgesteld. Naast nieuwe informatie wordt ook het oude materiaal herzien en eventueel vervangen. Bijvoorbeeld nieuw beeldmateriaal van bestaande producten. De catalogus wordt naar grotere klanten toegestuurd, de andere worden verspreid op het PartyCupsfestijn of via de vertegenwoordiger bij de klant afgeleverd.

Bij klachten over de orders wordt de vertegenwoordiger altijd betrokken bij de oplossing. Soms klopt er iets niet met aantallen, andere keren is vergeten een korting door te berekenen. Een heel enkele keer wordt geklaagd over het materiaal zelf. Dan wordt ook de leverancier op de hoogte gebracht van de klacht. Alleen de klant kan de klacht opheffen. In een enkel geval kan het betekenen dat PartyCups de gehele order crediteert. Als dat gebeurt wordt de klacht ook als afgehandeld gezien. Uiteraard moet dan ook eventuele provisie van de vertegenwoordiger worden teruggedraaid.

Een bijzonder traject wordt afgelegd als de klant materiaal met opdruk wil. Dan moet eerst het gehele creatieve proces worden doorlopen. Er wordt door een reclamebureau, dat eventueel door PartyCups wordt ingeschakeld, eerst een ontwerp van de afbeelding gemaakt. Daarna volgt na goedkeuring een proefdruk. Na opnieuw goedkeuring door de klant

volgt de preproductie, waarbij enkele tientallen exemplaren worden voorzien van de opdruk. Als de klant ook nu weer akkoord geeft, wordt de feitelijke productie gedraaid. Een dergelijk traject kan veel langer duren dan de standaard bestellingen uit de catalogus.

3.4 De **orderadministratie** in schema

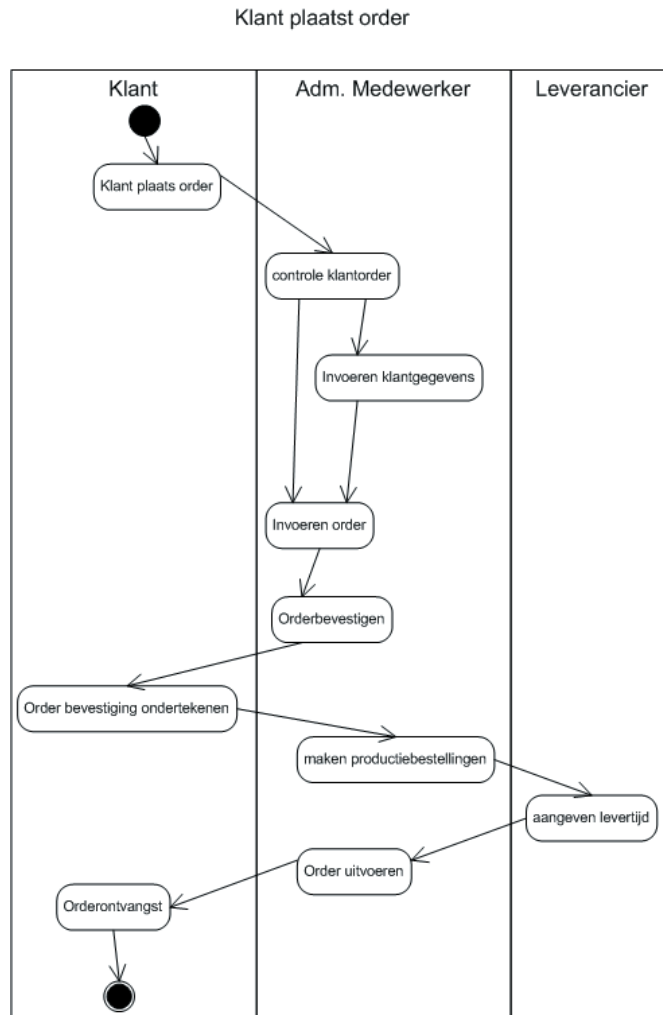
Met de beschrijving uit de case onderscheiden we een aantal processen bij PartyCups. Elk halfjaar wordt door het bedrijf een feest georganiseerd. Er is het **orderproces**, het samenstellen van de productcatalogus en het proces om speciale producten te realiseren. Ook de inkoop kan als een apart proces worden onderscheiden. Niet elk proces is even uitvoerig beschreven. Ook in de praktijk blijkt dat van het ene proces veel meer bekend is dan van een ander proces. Als er onvoldoende informatie over een proces bekend is, zullen we deze informatie alsnog dienen te achterhalen. Bijvoorbeeld door het organiseren van interviews. Het orderproces wordt uitvoerig beschreven in de case, dus kunnen we dit vastleggen in een activiteitendiagram.

In de casebeschrijving zien we dat er in een aantal alinea's wordt gesproken over het orderproces. Het deel van de informatie over dit proces vinden we in de case bij de beschrijving van het contact met de klanten van PartyCups. Een order kan op twee manieren tot stand komen, via een vertegenwoordiger of rechtstreeks door de klant. Een order die rechtstreeks door de klant wordt geplaatst, wordt direct door een administratief medewerker verwerkt. De orders via een vertegenwoordiger worden eenmaal per week ingeleverd bij de administratie en daarna verwerkt. Elke klant krijgt daarna een orderbevestiging die ondertekend teruggefaxt moet worden. Pas als de bevestiging van de klant terug is, wordt de order uitgevoerd. Beide manieren leggen we vast in de vorm van activiteitendiagrammen.

Uit beide diagrammen kunnen we verschillende activiteiten afleiden van de betrokken actoren. Ook kunnen we zien dat in de diagrammen de activiteiten van de leverancier beperkt zijn tot het doorgeven van de levertijd. Er is niet voldoende beschreven over deze actor in de case. Een activiteitendiagram is dan ook niet een document waarvan we kunnen constateren of het juist of onjuist

is. Het gaat er om of de betrokken actoren (en in dit geval natuurlijk vooral de actoren die bereikbaar zijn binnen het bedrijf) het eens zijn met de weergegeven activiteiten of niet. Zij kunnen zelf aangeven of een activiteit overbodig is, dan wel dat er een of meer ontbreken.

Figuur 3.8
Het activiteitendiagram
'Klant plaatst order'

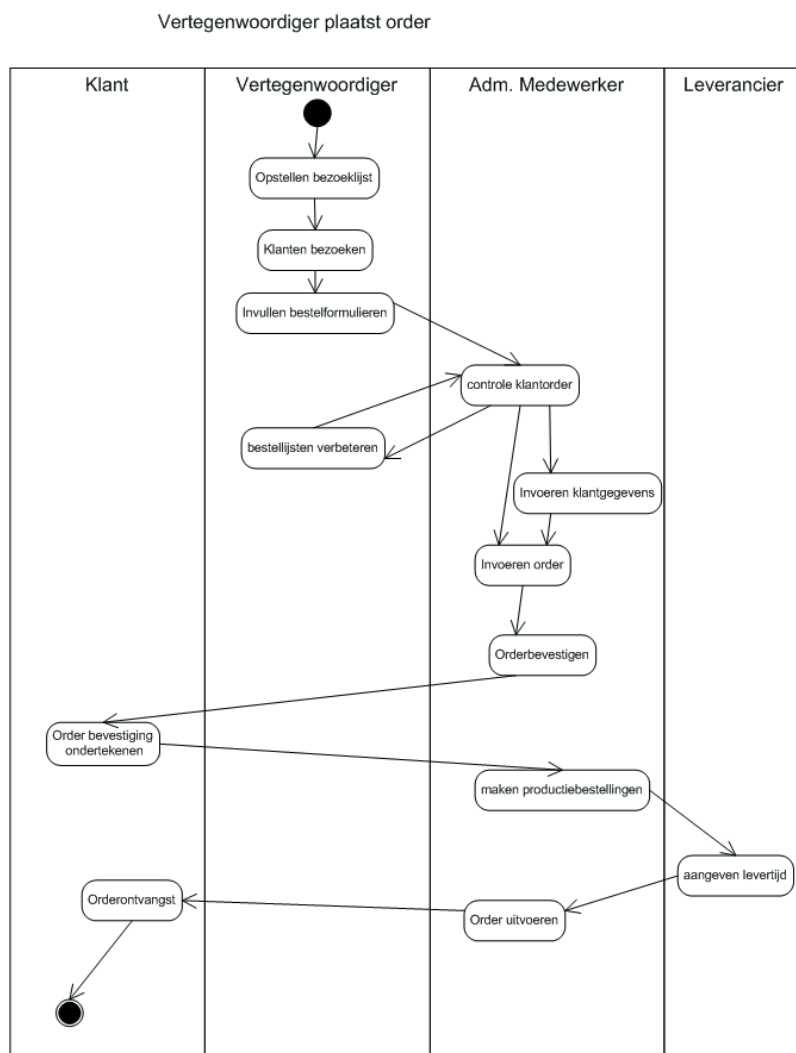


In figuur 3.9 kunnen we ons afvragen of de activiteiten 'klanten bezoeken' en 'invullen bestelformulieren' niet samengevoegd kunnen worden onder de laatst genoemde activiteit. Het kan echter zijn dat er in de organisatie redenen zijn om activiteiten apart op te nemen. Er moet dan wel bij iedere betrokkene een duidelijk beeld zijn wat

een activiteit inhoud en wie voor de uitvoering verantwoordelijk is. Een andere constatering is dat in figuur 3.9 het initiatief bij de uitvoering van dit proces bij de vertegenwoordiger ligt.

Figuur 3.9

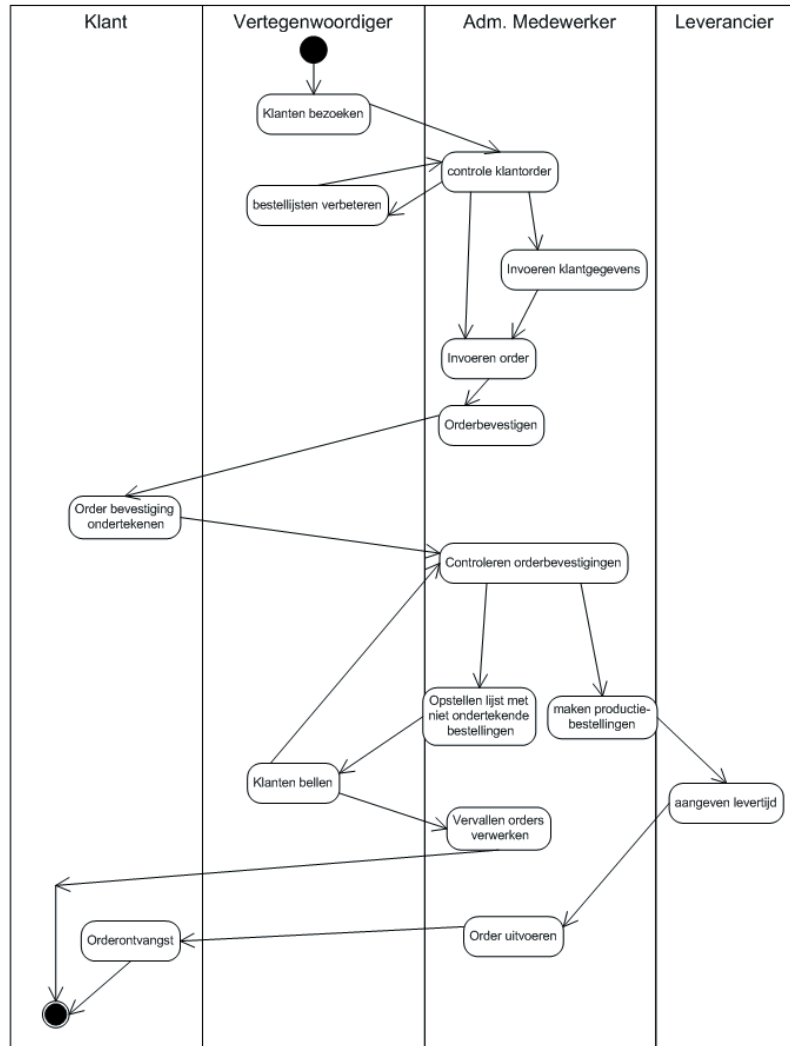
Het activiteitendiagram
'Vertegenwoordiger plaatst order'



In de casebeschrijving wordt ervan uitgegaan dat de administratie elk halfjaar zorgt voor het opstellen van enkele lijsten die de vertegenwoordiger gebruikt bij het maken van een bezoekschema. We kunnen deze correcties in dit diagram aanbrengen of besluiten om het opstellen van een bezoekschema uit het orderproces te halen. Het laatste verdient vanwege de vereenvoudiging de voorkeur.

Figuur 3.10
Het activiteitendiagram
'Vertegenwoordiger
plaatst order' met
aangebrachte correcties

Vertegenwoordiger plaatst order met correcties



Ten slotte kunnen we constateren dat in het huidige diagram nog geen rekening is gehouden met de mogelijkheid dat de klant de order vergeet of niet wenst te plaatsen. In de casebeschrijving wordt daarover gemeld dat de vertegenwoordiger telefonisch contact opneemt met de klant. Na het aanbrengen van de beschreven wijzigingen ziet het activiteitendiagram er uit zoals in figuur 3.10. Orders kunnen nu worden verwijderd door de administratieve medewerker.

3.5 Bijzondere bestellingen

Figuur 3.11
Activiteitendiagram voor
bijzondere bestellingen



Als de klant materiaal met opdruk wil, wordt een ander traject gevolgd dan bij de standaardorder. Uit de casebeschrijving blijkt dat er een reclamebureau bij dit proces is betrokken. Soms wordt dit door PartyCups zelf ingeschakeld, in andere gevallen blijkbaar door de klant. Er wordt een ontwerp van de afbeelding gemaakt en dit ontwerp moet worden goedgekeurd door de klant. Dan volgt er een proefdruk, maar het is onduidelijk wie dit doet en waar dit gebeurt. We nemen nu aan dat dit ook door het reclamebureau wordt verzorgd. De preproductie die daarna plaats vindt, wordt uitgevoerd bij het productiebedrijf. Na de laatste goedkeuring door de klant vindt de uiteindelijke productie plaats, waarna aflevering bij de klant kan volgen. In een activiteitendiagram ziet dit eruit als in figuur 3.11.

3.6 Samenvatting

Bij incrementeel ontwikkelen worden de onderdelen van de applicatie naast elkaar ontwikkeld. Hierdoor komen onderdelen al gereed terwijl andere onderdelen zich nog in de ontwerpfase bevinden.

Uit de use-case diagrammen worden de activiteitendiagrammen ontwikkeld. Hierin leggen we alle activiteiten in volgorde van uitvoering vast. We geven ook aan wie de activiteit uitvoert. Hiervoor gebruiken we ‘swimlanes’, verticale stroken voor elke actor in het activiteitendiagram waarin de activiteiten van de desbetreffende actor staan.

Elk activiteitendiagram heeft een startsymbool. Als een activiteit ergens eindigt, plaatsen we daar een stopsymbool. Als uit één activiteit er meerdere andere gelijktijdige activiteiten voortvloeien, of als er één activiteit het vervolg is van meerdere activiteiten, gebruiken we een synchronisatiebalk.

Opgaven

1. Maak het activiteitendiagram van de klachtafhandeling. Ga daar bij uit van de volgende actoren: klant, administratief medewerker, vertegenwoordiger en leverancier.
2. Maak het activiteitendiagram van het opstellen van de bezoekl lijst. Ga ervan uit dat het initiatief bij de administratie ligt. Er zijn twee actoren bij betrokken; de administratief medewerker en de vertegenwoordiger.
3. Maak het activiteitendiagram van het samenstellen van de product-catalogus.
4. Registratie zwemwedstrijd
Een zwemvereniging organiseert regelmatig kunstzwemwedstrijden voor kinderen. Hierbij komen teams bij elkaar voor wedstrijden op een wedstrijddag, waarop de kinderen figuren zwemmen. De figuren, die door de zwemmers individueel worden uitgevoerd, zijn bijzondere waterballetbewegingen, zoals het op je rug zwemmen met één been recht omhoog gestoken. Voor de figuren worden

punten gegeven. Kinderen moeten voor een wedstrijddag hun naam, leeftijd, adres en teamnaam opgeven. Om het geven van punten te vergemakkelijken krijgt elke deelnemer een nummer.

Op de wedstrijddag vinden er tegelijkertijd evenementen plaats (figuren) bij verschillende steunpunten die rond het zwembad zijn ingericht, doorgaans één in elke hoek. Verschillende juryleden en puntentellers worden voor een wedstrijddag aan een steunpunt toegewezen. In de loop van een seizoen kunnen juryleden en puntentellers bij verschillende steunpunten dienst doen. Omwille van de eenheid bij het puntentellen, wordt elk evenement met dezelfde juryleden bij precies één steunpunt uitgevoerd. In de loop van een wedstrijddag kan een steunpunt meerdere evenementen (figuren) verwerken.

De deelnemers worden opgedeeld in groepen en elke groep begint bij een ander steunpunt. Als een kind bij een bepaald steunpunt klaar is, gaat hij of zij naar een volgend steunpunt voor een ander evenement. Als iedereen een bepaald evenement bij het steunpunt uitgevoerd heeft, gaat het steunpunt over tot het volgende toegepaste evenement. Elke deelnemer mag bij een evenement één poging doen. Het nummer van de deelnemer die aan de beurt is, wordt aangekondigd aan de kinderen en aan de puntentellers. Soms raken de kinderen of de puntentellers over de volgorde in de war en stopt het steunpunt tot alles weer uitgezocht is.

Elk jurylid geeft een cijfer door een genummerde kaart (1 t/m 10) omhoog te steken. De cijfers worden door de puntentellers geregistreerd en omgerekend tot een score. Het hoogste en het laagste cijfer worden buiten beschouwing gelaten, en het gemiddelde van de overige cijfers wordt vermenigvuldigd met de moeilijkheidsfactor van de figuur. Aan het eind van de wedstrijddag worden er individuele prijzen en teamprijzen uitgereikt, gebaseerd op de hoogste individuele en teamscores. Er zijn verschillende leeftijds-categorieën, met prijzen voor elke categorie.

Het systeem moet alle informatie kunnen leveren en vastleggen die nodig is voor het plannen, registreren en scoren. Het systeem beperkt zich voorlopig tot alleen de figuren. Aan het begin van het seizoen worden alle zwemmers ingevoerd en wordt er een wedstrijdschema voor het seizoen opgesteld, inclusief de figuren die

gedurende de wedstrijddagen worden beoordeeld. Voor een wedstrijd worden de deelnemers geregistreerd, en tijdens de wedstrijd worden de scores vastgelegd en de winnaars bepaald.

Leg het registratiesysteem zwemwedstrijden vast in een of meer activiteitendiagrammen.

5. Bioscoopreservering

Een bioscoop heeft een mogelijkheid voor telefonische reservering. Voor elke film is het mogelijk om tot twee uur voor de aanvang van de film een of meer plaatsen te reserveren. Als er vaste plaatsen bij de film zijn, worden de stoelnummers bij de reservering vastgelegd anders alleen het aantal personen. Een reservering kan onmogelijk zijn indien het gewenste aantal plaatsen niet meer mogelijk is. De reservering is natuurlijk aan tijd gebonden en dient een half uur voor aanvang van de film te worden opgehaald.

We onderscheiden klant en bioscoopmedewerker. Maak het activiteitendiagram van de reservering. Welke informatie ontbreekt om een volledig diagram te kunnen maken?

6. Maak een of meer activiteiten diagrammen voor Hotel Amsterdam (zie opgave 10 van hoofdstuk2, p. 32).

7. Huisartsenpost

In een middelgrote gemeente in het westen van het land is een huisartsenpost opgezet om de avond- en weekenddiensten over te nemen van de aangesloten huisartsen. Elke dienst begint om 16.00 uur en duurt tot 08.00 uur de volgende morgen, er wordt met twee ploegen gewerkt. De weekenddienst begint om 08.00 uur op de zaterdag en eindigt om 08.00 uur op de maandagochtend, er wordt dan met drie ploegen gewerkt. De bezetting van de huisartsenpost bestaat uit twee assistenten (meestal verpleegkundigen) en de dienstdoende huisarts. Om de aangesloten huisartsen te informeren, wordt van elk patiëntenbezoek een registratie vastgelegd. Eventuele vervolghandelingen dienen immers ook door deze huisartsen te worden uitgevoerd.

De meeste incidenten komen telefonisch binnen. Afhankelijk van de vermeende ernst van de klacht wordt de patiënt doorverwezen naar de huisarts, gevraagd om langs te komen of gevraagd zich te

melden bij de afdeling spoedeisende hulp van het ziekenhuis. Van elke telefonische melding wordt het tijdstip, een beschrijving van de klacht en het gegeven advies vastgelegd en uiteraard de behandelende verpleegkundige of dienstdoende arts. Aan het begin van de dienst wordt een uitdraai gemaakt voor elke aangesloten huisarts. Deze uitdraaien worden elke werkdag aan de desbetreffende artsen toegezonden.

Elke maand wordt er voor het management van de artsenpost een rapportage gemaakt met daarin een overzicht van het aantal afgehandelde incidenten, verdeeld naar aard en tijdstip. Dit overzicht wordt gebruikt om de inzet van personeel beter te kunnen aansturen en tevens om de financiële vergoeding te regelen van de aangesloten artsen. Daartoe krijgt elke huisarts ook een overzicht van de verrichte behandelingen bij de bij hem of haar aangesloten patiënten. De huisarts kan daarmee de declaraties naar particuliere patiënten en ziekenfonds regelen.

Stel een activiteitendiagram op dat het bovenstaande proces weer-geeft.

8. Nationaal Instituut voor Information Engineers

Het Nationaal Instituut voor Information Engineers (NIIE) verzorgt opleidingen voor dertig modules op informaticagebied. De meeste modules leiden op voor een rijks erkend diploma, het Bedrijfsdiploma Information Engineer ofwel BIE. De overige modules behelzen een vakopleiding voor specifieke functies in de informaticasector. Het NIIE heeft jaarlijks zo'n 15.000 cursisten.

Gedurende het jaar worden per module een aantal cursussen georganiseerd. Op een cursus worden minimaal tien en maximaal dertig cursisten geplaatst. Een cursus kan door het gehele land gegeven worden, afhankelijk van de woonplaats van de grootste groep deelnemers. Nadat de inschrijving voor een cursus is geopend, wordt gedurende de inschrijvingsduur steeds gekeken of de locatie nog actueel is. Er zijn vaste lestijden. De cursus wordt zowel overdag als in de avonden aangeboden. Zodra een cursus tenminste tien kandidaten heeft, wordt een docent aan de cursus toegevoegd. Voor het geven van deze cursussen heeft het NIIE ruim 120 docenten in dienst. De meeste docenten werken op freelance basis. Slechts enkelen zijn in vaste dienst. Op basis van praktijkervaring

en opleiding wordt bepaald of een docent bevoegd is om een bepaalde module te geven. Docenten in vaste dienst ontwikkelen ook het lesmateriaal.

De organisatiestructuur van NIIE is eenvoudig. Aan het hoofd van het bedrijf staat de directeur, die wordt ondersteund door de administratief en educatief manager. De educatief manager stuurt de docenten in vaste dienst en freelancers aan. De administratief manager geeft leiding aan de afdelingen cursusplanning, cursusedministratie, personeelsadministratie en financiële administratie.

Om het NIIE voor de komende vijf jaar op een doelmatige en op de markt afgestemde wijze te kunnen laten opereren, is besloten om de verouderde systemen te vervangen door een geïntegreerd systeem. Het geïntegreerde systeem moet tevens remote toegang bieden aan de docenten ter ondersteuning van het onderwijs. Daartoe worden alle docenten voorzien van een laptop, waarmee ook (les)presentaties kunnen worden gegeven.

Het nieuwe systeem moet de volgende functionaliteit bieden:

- Het inschrijven van nieuwe (freelance) docenten. Daarbij wordt bij de inschrijving aangegeven welke modules zij bevoegd zijn te geven en in welke regio zij actief wensen te zijn. Het land is daarbij verdeeld in vier regio's, namelijk Noord-West, Zuid-West, Noord-Oost, Zuid-Oost.
- Het invoeren van nieuwe cursussen. Op grond van ervaringscijfers worden bestaande cursussen in gepland. Nieuwe cursussen worden als proef in de grote steden uitgebracht en, als ze succesvol zijn, opgenomen in het reguliere aanbod. Als een cursus gedurende een jaar niet wordt gegeven, wordt deze afgevoerd, tenzij de cursus deel uitmaakt van een officieel erkend diploma. Cursussen die zijn gepland worden opengesteld voor inschrijving.
- De inschrijving op een cursus kan op verschillende manieren plaatsvinden. Zo kan een cursist zich via internet inschrijven (wat een nieuwe mogelijkheid is), of zich telefonisch of schriftelijk opgeven. Cursisten krijgen een bevestiging van de inschrijving. Als een cursus op een bepaalde plaats en tijdstip is volgeboekt, worden eventuele alternatieven aangegeven. Een inschrijving is pas definitief als het cursusgeld is voldaan.
- Bij onvoldoende inschrijvingen wordt dit twee weken van tevoren aangegeven en wordt een alternatief aangeboden. Als een cursus

onverwacht enorm populair is, worden er extra cursussen opgenomen in de planning.

- Docenten krijgen elke maand een overzicht van de door hen gewerkte uren. Een keer per kwartaal volgt de financiële afrekening.

Om de inschrijving via internet mogelijk te maken, moeten de cursusplanning en de modulebeschrijvingen op internet worden geplaatst. Naast het geïntegreerde kantoorstelsel moet er een aparte server komen om de webtoegang te kunnen realiseren. Ook dient deze activiteit toegewezen te worden naar een onderdeel uit de organisatie.

De diplomaregistratie wordt bijgehouden om te controleren of een cursist inderdaad alle verplichte modules heeft behaald, zodra deze een aanvraag indient om in het bezit te komen van het BIE.

Maak een activiteitendiagram. Geef aan welke vragen je zou stellen naar aanleiding van de case en je analyse daarvan.

9. Boekenfonds

Op een school voor volwassenen is als serviceverlening aan de studenten een boekenfonds ingesteld. De studenten op deze school kunnen per vak afstuderen en kunnen in een verschillend leerjaar een vak volgen. Op het uitgereikte en voorgenummerde boekenfondsformulier moet de student aangeven voor welk vak of vakken (met het leerjaar) hij/zij aan het boekenfonds wil deelnemen.

Door de administratie wordt een formulier op verzoek aan een student verstrekt. Aan het eind van een schooljaar worden ze aan alle huidige leerlingen door de docenten uitgedeeld. Studenten kunnen de ingevulde formulieren bij de administratie inleveren. De administratie controleert een keer per week of de formulieren juist zijn ingevuld. Bij formulieren die onjuist zijn, wordt contact opgenomen met de leerling. De juiste formulieren worden in een bestand opgenomen.

Na het invoeren van de formulieren wordt een verzamellijst afgedrukt die aan de conciërge wordt overhandigd. Deze zorgt voor het ophalen van de gewenste titels uit het boekenmagazijn. Nadat de boeken zijn opgehaald, wordt door een administratief medewerker elk formulier gebundeld met de gewenste boeken. Als er boeken

niet voorradig blijken te zijn, wordt een bestelling geplaatst bij de lokale boekhandel.

De gebundelde formulieren en boeken worden apart gelegd en de student krijgt een briefje om de boeken op te halen en een borgsom te voldoen. Nadat de leerling de borgsom heeft voldaan en de boeken heeft opgehaald, wordt dit geregistreerd in het bestand. Aan het einde van het studiejaar worden de boeken weer ingenomen en de borgsom minus een vergoeding teruggegeven mits de boeken in goede staat zijn.

Stel een activiteitendiagram op dat het bovenstaande proces weer geeft.

10. E-Lectora

E-Lectora is ontstaan uit een fusie van drie grote boekenwinkels, een boekenclub en een groot postorderbedrijf. E-Lectora is inmiddels marktleider op het gebied van elektronische boeken. Ook het postordergedeelte van het bedrijf (waar iemand via internet boeken, cd's, video's, dvd's, pc-software en games kan bestellen) is een enorm succes. Klanten worden lid van E-Lectora en krijgen daardoor de mogelijkheid tot korting op de aanschaf van producten. De verplichting voor de klant is ten minste één bestelling per half jaar te plaatsen. Om klanten te helpen bij hun keuze, wordt elk half jaar een zogenaamde kroonkeuze aangeboden in elke categorie van producten.

Elk half jaar wordt aan alle klanten een catalogus gestuurd met de meest actuele producten waaruit een keuze gemaakt kan worden. Elk kwartaal wordt de website ververs met nieuwe informatie en titels die in het assortiment zijn opgenomen en worden titels die zijn uitverkocht verwijderd.

Om de voorraden zoveel mogelijk te beperken werkt E-Lectora met een zogenaamde 'ijzeren voorraad' en een statistisch aangestuurd besteltraject. Afhankelijk van de hoeveelheid aanvragen in een bepaalde maand, wordt de bestelling bij de diverse leveranciers steeds aangepast. Wordt gedurende een drietal maanden geen enkele bestelling meer voor een product geplaatst, dan wordt dit automatisch aangemerkt als 'obsoleet' en kan er niet meer direct een bestelling door het systeem worden gegenereerd.

Het product komt in de aanbieding en er wordt naar gestreefd om in een half jaar de voorraad weg te werken. De lijst van aanbiedingen wordt door het systeem gegenereerd, maar pas nadat de productmanager van de betreffende categorie producten zijn goedkeuring heeft verleend, wordt de aanbieding actief.

Bij nieuwe producten wordt de eerste voorraad bepaald door de productmanager van de desbetreffende categorie producten. Het eerste halfjaar worden deze producten aangegeven met de status nieuw. Het bestellen van een nieuw product levert de klant extra voordeel. E-Lectora heeft om klanten aan zich te binden een puntensysteem geïntroduceerd, waarbij elke aanschaf resulteert in het verkrijgen van een aantal punten dat direct gerelateerd is aan de hoogte van het factuurbedrag. Bij nieuwe producten krijgt de klant bonuspunten. Deze bonuspunten worden afzonderlijk per product door de productmanager vastgesteld.

Om de nieuwe producten te kunnen bepalen die worden opgenomen in het assortiment, worden elk kwartaal gegevens verzameld van de diverse leveranciers. Nieuwe producten worden ingedeeld in drie klassen, 'potential bestseller', 'regular sale', 'optional sale'. Afhankelijk van deze indeling krijgt het product meer of minder ruimte op de website en in de catalogus.

De informatie over de door de productmanagers vastgestelde nieuwe producten wordt door hen doorgegeven aan de webmaster en aan de catalogusbeheerder. De catalogusbeheerder zorgt elk half jaar dat een nieuwe catalogus wordt samengesteld. Het definitieve ontwerp wordt door hem naar de drukker gestuurd, waarna de gedrukte exemplaren via de afdeling Verzending naar alle klanten gaan.

Bestellingen komen op verschillende manieren binnen: via de website www.e-lectora.com, via de fax, via de post en telefonisch. Nadat een controle is uitgevoerd of de klant lid is van E-Lectora, wordt tevens gecontroleerd of aan deze klant geleverd kan worden in verband met uitstaande betalingen. Alleen wanneer beide checks positief verlopen, wordt de bestelling verder verwerkt.

Verdere verwerking houdt in dat de producten die de klant bestelt, worden toegewezen aan deze klant. Voorraden worden aangepast.

Het komt voor dat een artikel tijdelijk niet aanwezig is. De bestelling van de klant wordt dan bevroren. Als de bestelling compleet is, wordt deze overgebracht naar de afdeling Verzending. Tezamen met de bestelde producten, dan wel de halfjaarlijkse kroonkeuze, krijgen de klanten een factuur toegezonden met daarbij tevens een acceptgiro.

Klachten van klanten worden meestal via de winkels afgehandeld. E-Lectora heeft een niet-goed-geld-teruggarantie. Daarbij gelden echter enkele uitzonderingen. Bij de categorieën video's, dvd's, pc-software en games worden de producten gesealed uitgeleverd. Als bij de retourzending dit seal verbroken is, kan het artikel niet terug worden genomen.

Op de website is naast de producten die via de postordertak worden uitgeleverd een ruim aanbod van e-books. De meeste titels zijn overigens Engelstalig en veel titels zijn nog gebaseerd op boeken die vrij zijn van auteursrecht. Om het aanbod te actualiseren en te verbreden wordt door E-Lectora een actief beleid gevoerd om auteurs te werven. Naast een vaste vergoeding per titel, die overigens van tevoren wordt vastgesteld, krijgt elke auteur een vergoeding per verkocht product.

E-Lectora houdt daarom van alle e-books bij hoe vaak er een download plaatsvindt. Elk kwartaal worden de afrekeningen naar de auteurs gestuurd. Klanten van E-Lectora kunnen met de verkregen punten op een voordelige manier e-books bestellen. Bij elk e-book is aangegeven hoeveel de korting bedraagt bij inlevering van een aantal E-Lectora-punten. Klanten die e-books afnemen worden een keer per maand gefactureerd via een acceptgiro.

E-Lectora wil nu e-books verder uit bouwen tot e-learning en ook studieboeken opnemen, in combinatie met educatief film- en multimediamateriaal. Om een en ander zo min mogelijk te laten ingrijpen op bestaande processen, wil men dit traject als proefproject gebruiken. Allerlei technische hoogstandjes wil men introduceren bij het vernieuwde websitegedeelte waar de e-books worden aangeboden. Het hele traject, van creatie van het e-materiaal tot en met de uiteindelijke verkoop ervan, dient zoveel mogelijk geautomatiseerd plaats te vinden. De huidige bedrijfsvoering mag door deze ontwikkeling niet worden gehinderd. Door het succes van de

afgelopen jaren heeft het bedrijf inmiddels een zeer gezonde financiële basis en wil men dit high-tech-high-touchproject graag gerealiseerd zien.

Maak het activiteitendiagram.

4

Functioneel ontwerp – het bouwen van klassendiagrammen

4.1 Inleiding

Inmiddels hebben we al aardig in beeld gekregen hoe de bedrijfsprocessen in elkaar zitten. Ook de activiteiten zijn geregistreerd en we hebben de diverse naast elkaar lopende processen aangegeven. Nu wordt het tijd voor het functioneel ontwerp. Hierin gaan we aangeven welke onderdelen wezenlijk zijn voor de applicatie en welke contacten er bestaan tussen de onderdelen.

4.2 Het klassendiagram

Bij het **functioneel ontwerp** moeten we de onderdelen benoemen met al hun kenmerken en handelingen. Om dubbel werk te voorkomen, proberen we onderdelen te ontwerpen die op verschillende plaatsen bruikbaar zijn.

Terzijde

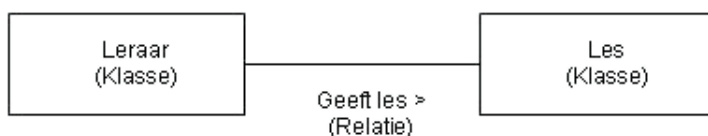
Een klok bestaat onder andere uit een aantal tandwielen. Alle tandwielen zijn verschillend. Sommige zijn dikker en zwaarder, andere juist erg dun en licht. De maat hangt af van de functie waarvoor ze gebruikt worden. Toch lijken alle tandwielen op elkaar. Ze zijn allemaal rond, hebben een aantal tanden en een as in het midden. Als we dus van tevoren bepalen hoe een tandwiel eruit ziet, hoeven we later alleen per tandwiel de verschillende variabele gegevens te definiëren. Het is zelfs mogelijk om de voor de klok omschreven kenmerken van een tandwiel ook te gebruiken voor de tandwielen van een grasmaaier of een fiets. Iets dergelijks doen we ook

bij het bouwen van software. We proberen algemene kenmerken die vaker kunnen voorkomen bij elkaar te groeperen. Hergebruik spaart erg veel tijd, maakt applicaties compacter en dus beter te onderhouden.

Het centrale onderdeel in ons ontwerp is de klasse. Deze bouwsteen is de basis van het gehele **objectgeoriënteerde ontwerp**. In een klassendiagram projecteren we de klassen en de relatie met andere klassen. Een goed **klassendiagram** kenmerkt zich door zoveel mogelijk onafhankelijke klassen en zo min mogelijk relaties met andere klassen. Natuurlijk moet wel alle functionaliteit aanwezig zijn.

Figuur 4.1

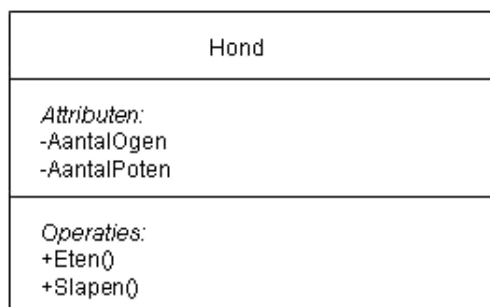
Een voorbeeld van een eenvoudig klassendiagram



Een **klasse** heeft zoveel mogelijk eigen kenmerken en handelingen. Deze kenmerken worden **attributen** genoemd, de handelingen operaties. Als we bijvoorbeeld de klasse Hond definiëren, kunnen we bepalen dat elke hond ogen en poten heeft en dat elke hond eet en slaapt. De ogen en poten zijn attributen, eten en slapen zijn handelingen. We noteren dat als volgt:

Figuur 4.2

De klasse Hond met attributen en operaties



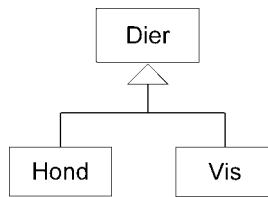
4.3 Overerving

Op soortgelijke wijze kunnen we de klasse Vis gaan bekijken. Net als een hond, heeft de vis ogen, eet hij en slaapt hij. In plaats van

poten heeft de vis vinnen. Daar is dus een verschil. Allebei (Hond en Vis) zijn dieren. Ze hebben een aantal gemeenschappelijke kenmerken en kunnen dus samen tot de klasse Dier gerekend worden. De afwijkende kenmerken (poten voor de hond en vinnen voor de vis) zijn dan geen kenmerken van de klasse Dier.

Figuur 4.3

De superklasse Dier met de subklassen Hond en Vis



We kunnen de klasse Dier bouwen en daarin alle gemeenschappelijke kenmerken voor Hond en Vis plaatsen. Vervolgens bouwen we de klassen Hond en Vis en zetten daarin alleen de afwijkingen (dus poten voor de hond en vinnen voor de vis). We verbinden de klassen Hond en Vis met de klasse Dier door een lijn met aan de kant van de klasse Dier een driehoek. Dit betekent dat de klassen Hond en Vis alle kenmerken van de klasse Dier erven. We noemen dit *overerving* of in het Engels *inheritance*. De klasse waarvan geërfd wordt, noemen we *superklasse*, de ervende klassen *subklassen*.

4.4 Van kandidaat-klassen tot klassen

Hoe *bepalen* we uit een probleem welke *klassen* we moeten gaan gebruiken? Een eenvoudige en doeltreffende methode is het onderstrepen van de zelfstandige naamwoorden in een tekst. Deze onderstrepingen leveren de *kandidaat-klassen* op, oftewel de mogelijke klassen. Als we alle zelfstandige naamwoorden gemarkeerd hebben, kijken we daarna of alle zelfstandige naamwoorden bruikbaar zijn. Sommige zijn synoniemen (verschillende woorden met dezelfde betekenis) of hebben een duidelijk verband met elkaar. Andere zijn vaag en daardoor onbruikbaar. Sommige zelfstandige naamwoorden geven eigenschappen weer van andere zelfstandige naamwoorden. Ook kunnen verschillende begrippen soms in één klasse gevangen worden. Laten we eens kijken naar de volgende tekst.

Terzijde

De keuze voor een computer hangt vooral af van het soort gebruik en de maximaal te betalen prijs. Een spelfanaat zal graag wat meer willen betalen voor een computer met uitgebreide multimediamogelijkheden terwijl een zakelijke computergebruiker vooral de stabiliteit van het systeem als belangrijkste aandachtspunt heeft. Toch kan men zich gemakkelijk vergissen. Tegenwoordig is een ingebouwde cd-brander zowel door de spelfanaat als de zakelijke gebruiker gewenst.

Als een zelfstandig naamwoord meer dan één keer voorkomt, onderstrepen we alleen de eerste keer dat het voorkomt. In een enkel geval nemen we ook het bijvoeglijk naamwoord mee als dit een verduidelijking van de kandidaat-klasse oplevert.

Als we gaan onderstrepen, levert dit de volgende zelfstandige naamwoorden op:

De keuze voor een computer hangt vooral af van het soort gebruik en de maximaal te betalen prijs. Een spelfanaat zal graag wat meer willen betalen voor een computer met uitgebreide multimediamogelijkheden terwijl een zakelijke computergebruiker vooral de stabiliteit van het systeem als belangrijkste aandachtspunt heeft. Toch kan men zich gemakkelijk vergissen. Tegenwoordig is een ingebouwde cd-brander zowel door de spelfanaat als de zakelijke gebruiker gewenst.

Er zijn synoniemen te vinden; zo komt zowel het woord computer als het woord systeem voor. Met beide wordt hier hetzelfde bedoeld. Multimediamogelijkheden en stabiliteit zijn beide eigenschappen van een computer evenals de aanwezigheid van een cd-brander. We houden dus na de eerste selectie de volgende kandidaat-klassen over:

- Keuze
- Computer
- Gebruik
- Prijs
- Spelfanaat
- Zakelijke computergebruiker
- Aandachtspunt

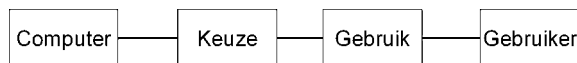
Bij deze kandidaat-klassen is in ieder geval Aandachtspunt onduidelijk. Deze schrappen we dus. Verder worden er twee soorten gebruikers vermeld. Die kunnen we vervangen door de algemenere klasse Gebruiker. Zo weten we het aantal klassen te beperken tot vijf:

- Keuze
- Computer
- Gebruik
- Prijs
- Gebruiker

De prijs is een kenmerk van de computer, dus een attribuut. Als we ervan uitgaan, dat we nu het optimale resultaat bereikt hebben, is de volgende stap om te bepalen welke klassen relaties met elkaar hebben. De gebruiker maakt zijn keuze en bepaalt hoe hij zijn computer gaat gebruiken. Een klassendiagram kan er dan zo uitzien:

Figuur 4.4

Een klassendiagram voor een computer



Terzijde

Het resultaat van elke bewerking hangt af van de interpretatie. Verschillende ontwikkelaars komen soms tot heel verschillende resultaten. Zowel beginners als gevorderden zijn geneigd te snel te willen modelleren. Daarna wordt het moeilijk om de bijzondere kanten van het probleem in te passen. Er wordt dan met allerlei middelen gerepareerd. Een eenvoudig en compleet resultaat leidt tot een versnelling van de rest van het project. Het gezegde ‘met passen en meten wordt de meeste tijd versleten’ geldt ook hier.

4.5 De case PartyCups

In hoofdstuk 3 is de organisatie van de onderneming PartyCups beschreven. Voor deze organisatie gaan we een relatiebeheersysteem bouwen. We moeten dus iets gaan doen aan klantenbeheer. Uit de omschrijving in hoofdstuk 3 selecteren we het voor ons belangrijkste deel:

De meeste klantcontacten lopen via de vertegenwoordigers. Ook zijn er klanten die telefonisch of schriftelijk via het kantoor in contact komen met PartyCups. Zodra een klant is opgenomen in het klantenbestand, wordt hij toegewezen aan een van de vertegenwoordigers. Bij bestellingen die spontaan bij PartyCups binnenkomen, wordt geen provisie berekend voor de vertegenwoordiger, in alle andere gevallen krijgt de vertegenwoordiger voor zelf aangeleverde orders 2% provisie en over de andere orders vanuit het klantenbestand van de vertegenwoordiger 1%.

De werkweek van een vertegenwoordiger bestaat grotendeels uit het bezoeken van klanten. Het streven is om bestaande klanten ten minste tweemaal per jaar te bezoeken. Dit wordt op dit moment niet altijd gehaald, omdat het soms onduidelijk is wanneer het laatste contact is geweest, of omdat het inplannen van het bezoek vergeten wordt. Daarnaast moet de vertegenwoordiger natuurlijk ook nieuwe contacten bezoeken. Die nieuwe contacten kunnen door de vertegenwoordiger zelf worden benaderd of zij zijn op eigen initiatief naar PartyCups gekomen. Ook worden potentiële klanten (leads) via het kantoor doorgegeven naar vertegenwoordigers.

Ook hier kunnen we de zelfstandige naamwoorden onderstrepen. Let wel goed op dat ons ontwerp alleen over het beheren van klantcontacten gaat, dus bijvoorbeeld niet over de bestellingen. Verder is een klassendiagram statisch. We leggen dus wel de klassen vast en de relaties tussen de klassen, maar niet hoe vaak een klant bezocht moet worden. Dergelijke toevoegingen komen later. Het gaat nu alleen over de hoofdlijnen.

Feitelijk zijn er maar drie klassen te destilleren uit de tekst, namelijk klant, contact en medewerker van PartyCups (vertegenwoordiger of medewerker op het kantoor).

4.6 Attributen en operaties

Zoals al eerder aangegeven, kunnen bij elke klasse attributen en operaties vermeld worden. Zo is het verstandig om bij de klasse

Klant bijvoorbeeld de volgende gegevens te vermelden:

- Bedrijfsnaam
- Vestiging (filiaal)
- Contactpersoon
- Adres van het filiaal
- Datum van de laatste bestelling

Als we de attributen en operaties in een applicatie willen gebruiken, zullen we ze daar moeten **declareren**: voordat we ze gaan gebruiken, melden we aan het programma aan welk **gegevenstype** ze voldoen. Zo kunnen we registreren of het gaat om gehele getallen, getallen met een komma erin, tekst, logische waarden (juist/onjuist), enzovoort. Elke programmeertaal heeft zijn eigen datatypes. In onze klassen vermelden we achter het attribuut of de operatie aan welk datatype de invoer, bewerking en uitvoer moet voldoen. Dat ziet er dan als volgt uit:

Figuur 4.5

Het aangeven van datatypes in Visual Basic

| |
|--|
| Bedrijf |
| Contactpersoon : String LaatsteContact : Date |
| |

Terzijde

Iedere programmeertaal kent zijn eigen datatypes en dus ook zijn eigen benamingen. Een aantal komt altijd voor: tekst (String of char), gehele getallen (Integer of int) en logische waarden (Boolean of bool). Voor gebroken getallen zijn er vaak verschillende aanduidingen waarbij de nauwkeurigheid en de grootte van het getal bepaalt welke variabele het meest geschikt is. In de tabel is te zien dat het aantal typen per programmeertaal verschilt.

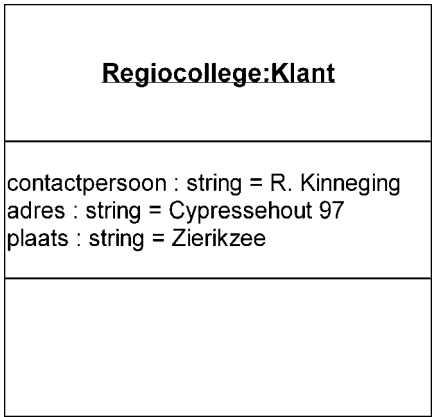
| Visual Basic | | C++ | | Java | |
|--------------|---------|-------------|----------------|---------|-------|
| Boolean | Long | bool | signed int | boolean | float |
| Byte | Object | char | signed long | byte | int |
| Currency | Single | double | signed short | char | long |
| Date | String | float | unsigned char | double | short |
| Double | Variant | int | unsigned int | | |
| Integer | | long | unsigned long | | |
| | | long double | unsigned short | | |
| | | short | void | | |
| | | signed char | wchar_t | | |

Tabel 4.1 Datatypes in verschillende programmeertalen

4.7 Klasse en object

Een klasse is altijd een abstract gegeven. Er wordt bijvoorbeeld een hond gedefinieerd. Als ik de kenmerken van mijn hond wil aangeven, dan kan ik die in de definitie van hond wel kwijt. Op het moment dat ik de voor mij van belang zijnde gegevens ga invullen, spreken we niet meer van een klassendiagram, maar van een *objectdiagram*. Een *object* bevat dus de beschrijving van een specifieke set gegevens, passend binnen de definitie van de klasse.

Figuur 4.6
Het object Regiocollege
van de klasse Klant



Als we een klasse dynamisch gebruiken en de waarden tijdens de uitvoering van het programma veranderen, kan het goed zijn om toch alvast beginwaarden te vermelden. Dat doen we op dezelfde wijze als in een objectdiagram.

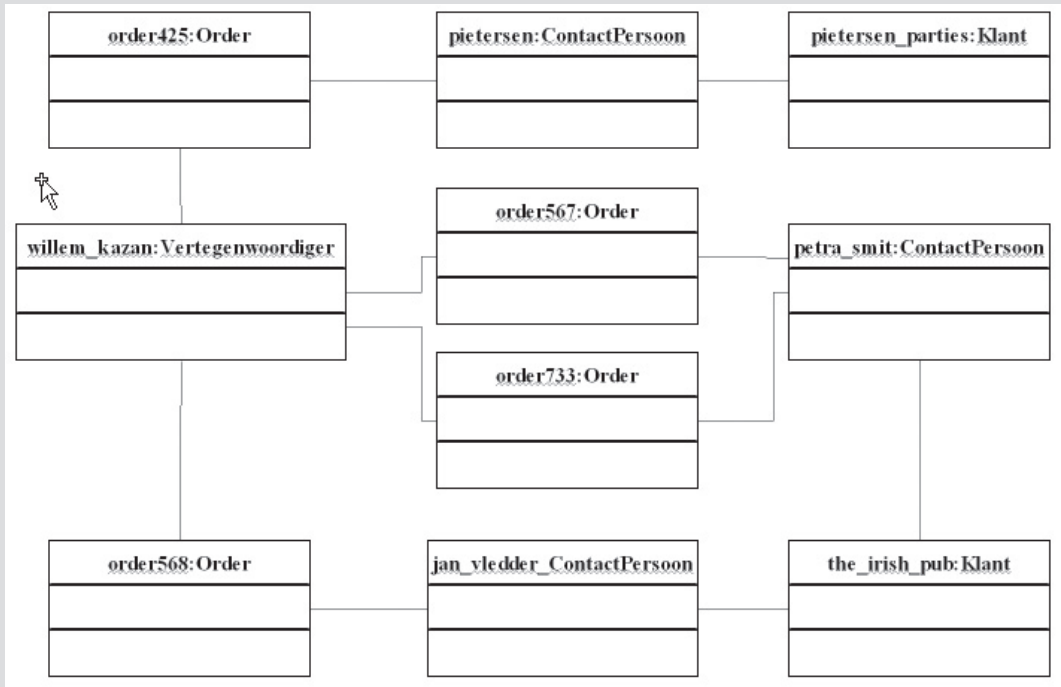
4.8 Samenvatting

Een klasse wordt aangegeven door een rechthoek. Deze rechthoek verdelen we in drie vakken. In het bovenste vak zetten we de naam van de klasse, in het vak daaronder de attributen en in het onderste vak de operaties. De verbindingslijn tussen twee klassen geeft de relatie aan. Klassen kunnen attributen en operaties erven van hogere klassen (superklassen). We geven dit aan met een pijl met een driehoekige, niet-ingekleurde pijlpunt. Deze overerving noemt men in het Engels *inheritance*. Voor het maken van een klassendiagram bestaat een eenvoudige trucje. In de beschrijving onderstrepen we alle zelfstandige naamwoorden; deze vormen de kandidaat-klassen. We bepalen welke aanduidingen vaag zijn, welke aanduidingen terugverwijzen naar andere zelfstandige naamwoorden, enzovoort. Van de overgebleven zelfstandige naamwoorden bepalen we de binding met de andere zelfstandige naamwoorden. Vervolgens tekenen we het klassendiagram. Een goed klassendiagram heeft zo min mogelijk bindingen. Achter elk attribuut en elke operatie kunnen we het datatype aangeven. De mogelijke datatypes zijn deels afhankelijk van de programmeertaal die gebruikt gaat worden. Een object bevat de ingevulde waarden van een klasse.

Opgaven

1. In figuur 4.2 staat een aantal kenmerken van de klasse Hond; er zijn echter veel meer kenmerken te benoemen. Geef nog twee attributen en twee operaties van de klasse Hond.
2. In de vorige opgave heb je twee attributen en twee operaties voor de klasse Hond toegevoegd. Ga na of deze bij de subklasse Hond horen of bij de superklasse Dier. Maak een nieuw klassendiagram met de attributen en operaties op de juiste plaats.
3. Voeg aan het klassendiagram ook een (sub)klasse Vogel toe. Controleer of alle kenmerken van de superklasse Dier ook bij Vogel voorkomen en noteer bij Vogel de afwijkende kenmerken.

4. Maak van onderstaande tekst een klassendiagram.
Voor mobiele telefoons bestaan allerlei verschillende abonnementsvormen. Gebruikers die slechts zelden zelf bellen, zijn meestal het voordeligst uit met een prepaid-abonnement. De veel-bellers kunnen kiezen uit verschillende abonnementsvormen, waarbij een aantal belminuten in het abonnementsbedrag zijn inbegrepen. Daarnaast bestaan er speciale abonnementen voor dataverkeer via GPRS en binnenkort UMTS. De leveringsvoorwaarde verschillen sterk per aanbieder (provider). De kwaliteit van de verschillende netwerken is tegenwoordig vrijwel gelijk.
5. Maak het klassendiagram voor het CRM-systeem.
6. ‘Medewerker van PartyCups’ is een generalisatie voor de vertegenwoordiger en de binnendienstmedewerker van de onderneming. Voeg deze subklassen toe aan het klassendiagram.
7. Aan het begin van paragraaf 4.6 wordt een aantal kenmerken opgesomd van de klasse Klant. Welke hiervan zijn attributen, welke operaties?
8. Bedenk nog twee kenmerken voor de klasse Klant.
9. Maak van het objectdiagram in figuur 4.7 een klassendiagram.
10. Een computersysteem bestaat uit een systeemkast met twee harddisks, een moederbord met een Intel-processor en 512 MB geheugen en een diskettestation, en daarnaast een monitor, een laserjet en een deskjet-printer, muis en toetsenbord. Maak een klassendiagram, inclusief super- en subklassen.

**Figuur 4.7** Het object diagram Plaatsen orders

Functioneel ontwerp - specificaties voor een database

5.1 Inleiding

In het vorige hoofdstuk bleek al dat klassendiagrammen krachtige instrumenten zijn bij softwareontwikkeling. In dit hoofdstuk gaan we de klassendiagrammen verder verfijnen. We zullen daarbij de relaties verder verfijnen.

5.2 Multipliciteit

Bij de **associatie** (de relatie) tussen twee klassen kunnen we aangeven hoeveel objecten van de ene klasse geassocieerd zijn (een relatie hebben) met een object van een andere klasse. Laten we dat eens aan de hand van een voorbeeld bekijken.

Een leraar zal één of meer lessen per week geven. Een les wordt altijd maar door één leraar gegeven.

Als we hiervan een klassendiagram maken, ziet het er simpel uit.

Figuur 5.1
Het Klassendiagram: Een
leraar geeft les



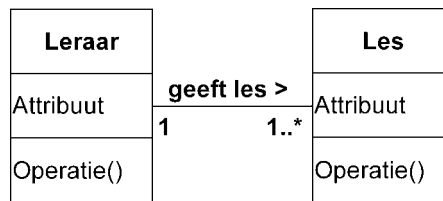
Het is nu eenvoudig om ook de aantallen die in de tekst vermeld zijn toe te voegen. We zetten naast de associatie getallen. Er zijn veel combinaties mogelijk:

- 1 betekent: het object aan deze kant komt 1 keer voor, dus niet minder (0) of meer dan 1.
- 1..8 betekent: het object aan deze kant komt in deze associatie 1 tot 8 keer voor.
- 2, 4, 6, 8 betekent: het object aan deze kant kan uitsluitend 2 of 4 of 6 of 8 keer voorkomen
- * betekent dat het object oneindig veel keren kan voorkomen.

Als we deze getallen naast de associatie zetten, spreken we van multipliciteiten.

We breiden het klassendiagram nu uit met **multipliciteiten** (zie figuur 5.2)

Figuur 5.2
Een klassendiagram met multipliciteiten



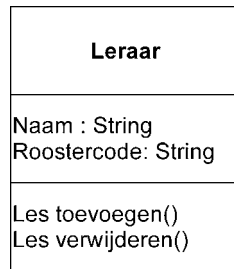
De leraren zullen niet blij zijn met dit resultaat, want er staat hier dat een leraar een onbeperkt aantal lessen per week kan geven.

5.3 Attributen en operaties

In de klassendiagrammen van figuur 5.1 en 5.2 is wel de driedeling gemaakt zodat attributen en operaties ingevuld kunnen worden, maar deze staan nog niet vermeld. Niet elke klasse kent attributen en/of operaties. Bovendien is het niet zo dat alle kenmerken van een klasse vermeld moeten worden. Alleen de **functionele kenmerken** zijn van belang. Als we naar de klasse **Leraar** kijken, moeten we bepalen welke attributen en welke operaties functioneel zijn. Voor de leraar kunnen we denken aan (uiteraard) zijn naam, de roostercode voor zijn naam en welke lessen hij geeft. Niet-functioneel zijn de leeftijd van de leraar, of hij een bril draagt, enzovoort. Attributen zijn vaststaande gegevens, operaties leiden tot veranderingen. De klasse **Leraar** kan er dus uitzien als in figuur 5.3.

Figuur 5.3

De klasse Leraar



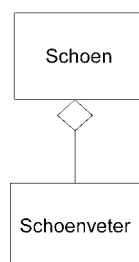
5.4 Aggregatie en compositie: gekwalificeerde associaties

In vorige paragrafen hebben we de verbinding tussen twee klassen met een eenvoudige rechte lijn aangegeven. We noemen deze verbinding een associatie. We kunnen deze associatie nog duidelijker aangeven met een **gekwalificeerde associatie**. Daarmee geven we aan dat de ene klasse is opgebouwd uit andere klassen.

De **aggregatie** is hiervan de eenvoudigste vorm. We geven een aggregatie aan met een open ruit (een wybertje) aan de kant van de gehele klasse. De betekenis is dat de onderdelen ook kunnen bestaan zonder de gehele klasse. Een voorbeeld is te zien in figuur 5.4.

Figuur 5.4

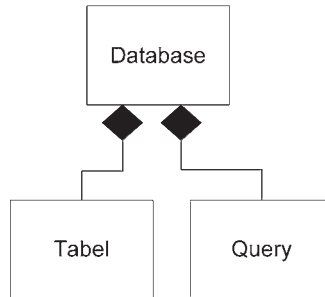
Een schoen met schoenveter. De veter kan bestaan zonder de aanwezigheid van de schoen



De **compositie** heeft een groter bindend karakter. Bij de compositie heeft de (deel)klasse geen bestaansrecht zonder de aanwezigheid van de gehele klasse. De aanduiding in het klassendiagram is een gesloten ruit aan de kant van de gehele klasse.

Figuur 5.5

Een database met tabellen en queries. Tabellen en queries hebben geen functie als zij geen onderdeel uitmaken van een database.



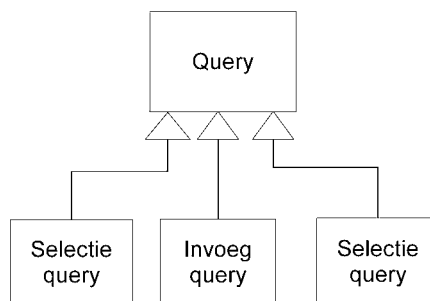
Terzijde

Het onderscheid tussen aggregatie en compositie is niet altijd even eenvoudig te maken. Let op dat de afhankelijke componenten kunnen bestaan zonder het geheel. Als je twijfelt, kun je eventueel kiezen voor de zwakste associatie, dus de aggregatie.

Eventueel kan in een klassendiagram zowel aggregatie/compositie als **generalisatie** voorkomen. Zo bestaan er verschillende soorten **queries**. We kennen bijvoorbeeld selectie-, invoeg- en update-queries. Deze worden samen als generalisatie ondergebracht onder de algemene klasse Query.

Figuur 5.6

Generalisatie van de klasse Query



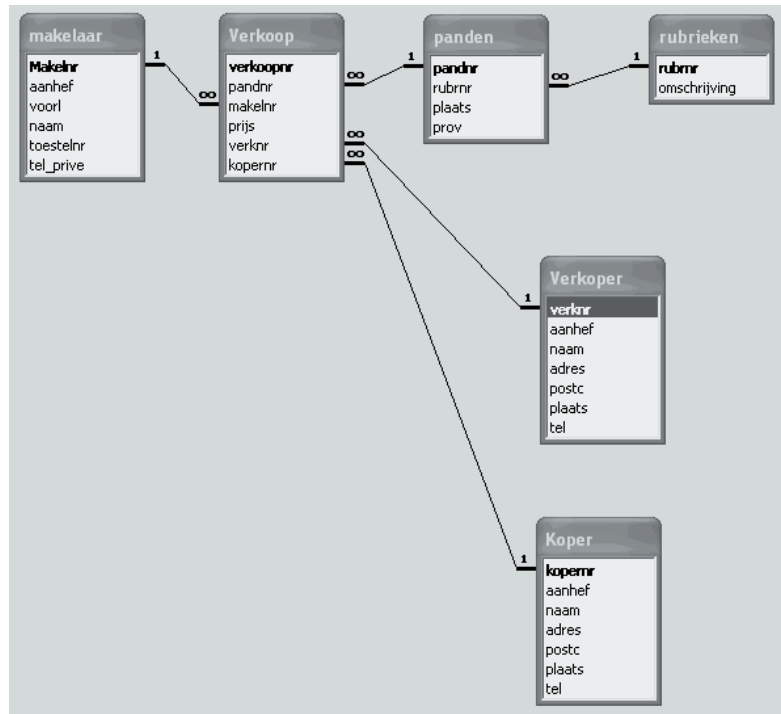
5.5 Toepassing op databases

In de vorige paragraaf gebruikten we de structuur van een database als voorbeeld. Ook de inhoud en de relaties van een **database** zijn zo in een **klassendiagram** onder te brengen. Velen zullen voor het weergeven van de opbouw van een database nog het klassieke

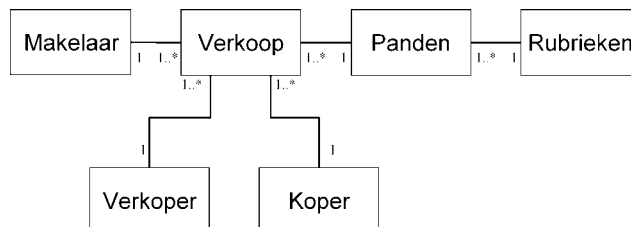
Entity Relation Diagram (ERD) gebruiken, maar een klassendiagram kan dezelfde functionaliteiten weergeven. We gaan hier dus niet in op het ERD, maar houden ons bij het klassendiagram. De kracht van het relationele model voor databases zit in de één-op-één en één-op-meer relaties van de tabellen in de database. In ieder geval levert een meer-op-meer relatie altijd grote problemen op. Het bouwen van een relationele database valt buiten het bestek van dit boek, maar de weergave van de tabellen en hun onderlinge samenhang bespreken we hier wel. Eigenlijk is het heel eenvoudig om een klassendiagram te tekenen van de tabellen met hun relaties. Elke tabel is een klasse en bij de associaties geven we de multiplaciteiten aan.

Figuur 5.7

Een databaseontwerp van een applicatie voor een makelaar in Microsoft Access

**Figuur 5.8**

Het klassendiagram van de applicatie van figuur 5.7



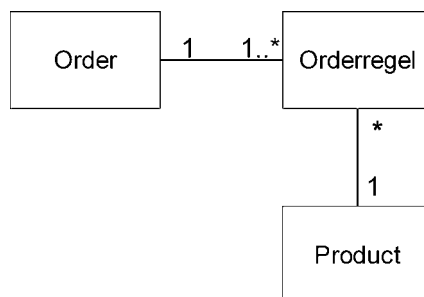
De velden in de tabellen kunnen in het klassendiagram als attributen weergegeven worden. Als we queries willen modelleren, zullen we bij deze klassen veel operaties vermelden; queries ‘doen’ immers iets met de database.

5.6 De database bij PartyCups

Ook bij PartyCups bestaat een genormaliseerde database. Een stukje van deze database laten we zien zoals deze ook beschreven staat in de tekst in hoofdstuk 3:

Bij een klantbezoek wordt de catalogus met de producten van PartyCups doorgenomen. De catalogus is ingedeeld in productgroepen: wegwerpartikelen, herbruikbaar plastic, serviesverhuur, meubilair (hieronder vallen ook de tenten en barbecues) en feestartikelen. Zoals al eerder aangegeven, worden de bestellingen genoteerd op een orderformulier. Elke week worden deze orderformulieren afgeleverd op het hoofdkantoor, waarna ze worden verwerkt.

Figuur 5.9
De tabellen uit de
orderdatabase van
PartyCups

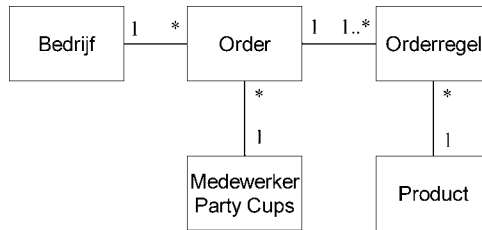


Eigenlijk is het een heel eenvoudige serie tabellen. Elke order bestaat uit een aantal orderregels. Op elke regel komt één product voor. Deze producten staan met hun kenmerken en leveranciers vermeld in andere tabellen. In een klassendiagram is het mogelijk niet alleen de tabellen, maar ook de betrokkenen bij het gebruik van de tabellen op te nemen. In dit geval gaat het over het bestellende bedrijf en de medewerker van PartyCups. Het klassendiagram wordt dan uitgebreid tot de afbeelding in figuur 5.10.

Let op de multipliciteiten in figuur 5.10. Een bedrijf kan veel orders plaatsen, maar elke order is bestemd voor maar één bedrijf. Bij elke order hoort maar één medewerker van PartyCups.

Figuur 5.10

Het uitgebreide klassendiagram voor de orderdatabase van PartyCups



Laten we nog eens kijken naar het klassendiagram van de order in figuur 5.9. Daar wordt Product als klasse genoemd in relatie met Orderregel. Uit de tekst van PartyCups blijkt dat ook de catalogus met behulp van de klasse Product wordt gerealiseerd. Om tot een catalogus te komen moet een aantal attributen van een product bekend zijn. Voor de hand ligt het om een naam, een omschrijving enzovoort als attributen te onderscheiden. Elk attribuut geeft daarbij een kenmerk weer van in dit geval het product.

Als PartyCups de catalogus in meerdere talen wil laten drukken, wordt de relatie met sommige attributen anders. In dit geval willen we nu ook weten in welke taal de omschrijving is vastgelegd. In plaats van een attribuut wordt omschrijving nu een zelfstandige klasse met een verwijzing naar een product. Elk product kan meer dan een omschrijving hebben. Dat wordt bepaald door de taal. Als we vinden dat elk product ten minste een omschrijving dient te hebben zullen we dit aangeven in de multipliciteit met een 1..*. Als er ook producten kunnen voorkomen zonder beschrijving dan wordt de multipliciteit *.

Bij het organiseren van het halfjaarlijkse feest wil men vastgelegd hebben wie allemaal een uitnodiging moeten krijgen. In de case-beschrijving wordt aangegeven dat het hier om medewerkers en vaste klanten gaat. Het is daarbij niet duidelijk of ook leveranciers worden uitgenodigd, maar gezien het karakter van het feest (onder andere productintroducties) is dat wel waarschijnlijk. Bij het opzetten van het relatiebeheersysteem voor PartyCups is er een beschrijving van Joep over het invoeren van klantgegevens in paragraaf 2.12. Bij een aantal grotere klanten zal het voorkomen dat meerde-

re personen deze klant vertegenwoordigen. We kunnen dus uitgaan van de situatie dat elk bedrijf ten minste een, maar soms meerdere contactpersonen heeft. Het relatiebeheersysteem wordt gevuld met de gegevens van contactpersonen, bedrijven en medewerkers. Bij het inbrengen van deze gegevens moet worden aangegeven of het om een medewerker, klant of leverancier gaat.

Al eerder is Medewerker als klasse weergegeven, zie figuur 5.10. Ook is daar de klasse Bedrijf aangegeven als klasse waardoor orders gekoppeld kunnen worden. Als orders echter geplaatst kunnen worden door meerdere personen moet het diagram worden aangepast. Een order wordt nu gekoppeld aan een contactpersoon en niet langer direct aan een bedrijf. Een contactpersoon kan echter maar bij één bedrijf horen, zodat een order toch gerelateerd blijft aan één bedrijf.

Joep: Het komt regelmatig voor dat de klant als hij hoort van de levertijd een deel van het materiaal dat al wel beschikbaar is al uitgeleverd wil hebben. We splitsen de order dan in deelleveringen uit. Bij het ontvangen van de producten die nageleverd moeten worden moet dan eerst worden gekeken naar deze deelleveringen. Nu gaat dat nog wel eens mis omdat door alle drukte vergeten wordt de deellevering te noteren. Bij het uitleveren aan de klant dient deze wel te tekenen. Bij de financiële administratie die door Jan wordt beheerd gaat er dan het een en ander mis. De klant krijgt dubbel geleverd en dubbel gefactureerd. Zelfs als de klant het te veel geleverde materiaal heeft teruggestuurd kan het soms lang duren voordat er door ons gereageerd wordt. Een aantal klanten is hierdoor erg boos geworden.

Leontien: In het nieuwe systeem moet van elke order zichtbaar zijn welke leveringen al hebben plaatsgevonden. Er kan alleen op de deelleveringen gefactureerd worden. Daarmee zal het probleem van dubbelleveringen gelukkig voorkomen worden. Bij ontvangst van nieuw materiaal moet er een lijst kunnen worden getoond van orders waarvan nog niet alles is uitgeleverd. Die lijst hoeft niet op papier te worden afgedrukt. Zoiets kan gewoon op het scherm worden getoond.

Marloes: Bij het samenstellen van de catalogus is zeker bij de nieuwe producten niet altijd meteen een afbeelding beschikbaar. Ook moet er voor elk nieuw product een beschrijving worden gemaakt. Ik voer meestal eerst alle nieuwe producten in en daarna in de loop van de tijd de beschrijvingen en de plaats waar de afbeelding is opgeslagen. De prijzen van de producten worden opgenomen in de database en niet afgedrukt in de catalogus, maar op aparte inlegvellen. Hierdoor blijft de catalogus langer bruikbaar. Het afdrukken van de catalogus, althans de eerste versie, gebeurt direct vanuit de producttabel. Er is soms enige discussie over de prijs van een product. Als een van onze vertegenwoordigers het verkeerde inlegvel gebruikt, kan het zijn dat een onjuiste prijs wordt gehanteerd. Het zou prettig zijn om ook naast de actuele prijs ook de historie van de prijzen terug te kunnen zien in een overzicht. Zo kunnen we ook beter met tijdelijke acties werken.

Jan: Bij het onderhandelen met onze leveranciers is het prettig om oude afspraken gemakkelijk terug te kunnen vinden. Met het nieuwe systeem voor onze klanten lijkt het me zinvol dit ook te gebruiken voor mijn contacten met leveranciers. Daarbij is een overzicht van in het verleden gehanteerde inkoopprijzen en verleende kortingen heel prettig om bij een gesprek als informatie te hebben.

Joep: Vooral bij het plaatsen van speciale orders, orders waarbij een opdruk moet worden gerealiseerd, is het belangrijk om de doorloopsnelheid van de order te bewaken. En er moet vanaf het begin ook aan verdiend kunnen worden en niet pas bij de volledige productie, zodat als een klant gedurende het traject van gedachten veranderd wij niet voor de kosten opdraaien. Er vindt immers veel terugkoppeling naar de klant plaats en dat moet goed geregistreerd worden. Met het nieuwe systeem kunnen we daar ook een bewaking op zetten. Er moeten overzichten uit het systeem gehaald kunnen worden en als met een bepaalde order een van tevoren afgesproken deadline overschreden wordt, moet het systeem dit automatisch signaleren.

Leontien: In de huidige situatie moeten alle klanten langs gelopen worden om te zoeken of er speciale orders zijn en wat de status daarvan is. Om dat te vereenvoudigen drukken we nu de afspraken van speciale orders af en slaan die op in een aparte ordner, die een keer per week wordt nagekeken. Dat is een saaie en tijdrovende klus. Bovendien komt het ook daar voor dat we vergeten een goedkeuring te noteren, waardoor er onnodig wordt gewacht. De klant moet ons dan uiteindelijk opbellen of wij bellen de klant, maar in beide gevallen is er dan toch sprake van irritatie.

Marloes: Je krijgt soms discussies met vertegenwoordigers over de volgens hen te laag berekende bonus. Dan blijkt dat soms hele orders niet zijn meegenomen of dat ten onrechte door de administratie een klant niet bij de vertegenwoordiger is geplaatst doordat deze te laat is ingevoerd. Dat uitzoeken kost bij de administratie ook veel tijd, want het kan soms wel twee maanden duren voordat het helemaal geregeld is. En al die tijd wordt ik dan bestookt met vragen, die ik dan ook niet kan beantwoorden. Het nieuwe systeem moet van alle uitgeleverde orders natuurlijk de bonussen meteen berekenen. Eigenlijk moet elke vertegenwoordiger zelf de orders kunnen invoeren en zijn aangepaste bonussen meteen kunnen inzien.

In de casus wordt gesproken over deelleveringen. Elke order kan dan in zijn geheel of in delen aan de klant geleverd worden. Over de deelleveringen die al verzorgd zijn moet ook indien van toepassing de bonus worden berekend. Dus hoort bij elke deellevering een bonus die ook op een overzicht per vertegenwoordiger getoond moet kunnen worden.

Bij het berekenen van de bonus voor elke vertegenwoordiger is er ook nog het probleem dat een bonus die al een keer is toegekend en zelfs al is uitbetaald niet nog een keer mag worden meegeteld. Daarom dient dit natuurlijk vastgelegd te zijn. In een van de klassen dient dit te worden bijgehouden. De bonus wordt in ieder geval bijgehouden bij een deellevering van een order.

5.7 Samenvatting

Bij de associaties tussen klassen kunnen multipliciteiten aangegeven worden. Daarbij is elke reeks mogelijk. Als we klassendiagrammen willen gebruiken voor de weergave van structuren in relationele databases, gebruiken we alleen één-op-één en één-op-meer relaties. De attributen bij de weergave van tabellen zijn de veldnamen, bij de weergave van queries komen ook veel operaties voor.

Opgaven

1. Pas het klassendiagram van Figuur 5.2 zo aan dat een leraar een bepaald maximumaantal lessen per week kan geven. Onderzoek zelf (vraag het, reken uit of bedenk een andere manier) hoeveel lessen een leraar maximaal per week kan geven. Let op, we leggen hier het maximum vast wat voor een leraar in het algemeen mogelijk of toegestaan is.
Elke leerling volgt ook een aantal lessen per week. Voeg een klasse Leerling toe, maak een associatie en geef de multipliciteiten aan. Onderzoek hoeveel lessen een leerling minimaal en maximaal volgt en hoeveel leerlingen er minimaal en maximaal in een les aanwezig kunnen zijn.
2. Bepaal attributen en operaties voor de klassen Leerling en Les.
3. Geef in de volgende voorbeelden van gehele klasse en deelklasse(n) aan of het gaat om aggregatie of compositie.
 - a. Een schaakbord met 64 velden.
 - b. Een auto met vier wielen.
 - c. Een boek met 312 bladzijden.
 - d. Een boormachine met een boor.
 - e. Een bierfles met bier.
4. In het voorbeeld van de compositie (zie figuur 5.5) zijn de tabellen een onderdeel van een database. Vul het klassendiagram aan met records en velden als onderdeel van de tabellen, en rapporten als onderdeel van de database.

5. Figuur 5.6 geeft de generalisatie van de klasse Query. Ook een veldtype in een database is een generalisatie. Welke veldtypen kunnen voorkomen? Teken de superklasse veldtype met minimaal drie van toepassing zijnde subklassen.
6. Figuur 5.10 geeft het uitgebreide klassendiagram voor de orderdatabase van PartyCups. Vaak zal PartyCups de producten uit de bestellingen van zijn klanten moeten doorplaatsen bij zijn leveranciers. Maak een klassendiagram voor de bestellingen bij leveranciers.
7. PartyCups wil de catalogus ook plaatsen op een website. De afbeeldingen die tot nu toe voor de catalogus worden gebruikt hebben een te hoge resolutie. Voor elk product wordt nu niet alleen een afbeelding met hoge resolutie, maar ook een met lagere resolutie en een snapshot opgeslagen. In de productklasse stond dit al als attribuut opgenomen. In dit attribuut is de plaats waar de afbeelding te vinden is opgeslagen. Geef de benodigde wijzigingen weer in een klassendiagram met alle benodigde klassen. Geef ook aan of het hier gaat om een compositie of aggregatie.
8. Pas het klasse diagram van figuur 5.10 aan aan de situatie zoals hiervoor beschreven. Maak ook het klassendiagram waarmee de uitnodiging aan alle betrokkenen kan worden gestuurd. Geef daarbij de attributen aan van de klassen, voorzover die in de tekst zijn aangegeven en nodig zijn voor een juist model.
9. Maak een klassendiagram van orders en deelleveringen waarmee zowel de overzichten van Leontien als Marloes gemaakt kunnen worden. Bij speciale orders moet rekening worden gehouden met een aantal productie stappen die bij de andere orders helemaal niet voorkomen. Hoe kunnen we dit vastleggen in het klassendiagram?
10. Als de vertegenwoordiger een overzicht wil hebben van de uitgekeerde bonussen van het lopende jaar die al zijn uitgekeerd en de uit te keren bonus van de huidige maand, welke aanpassingen moeten dan in het klassendiagram worden aangebracht (in welke klasse(n))?

6.1 Inleiding

Bij de aanpak van de bouw van softwaresystemen is in hoofdstuk 1 een aantal methoden in vogelvlucht behandeld. In dit hoofdstuk gaan we dieper in op een methode die meestal Unified Process wordt genoemd. Een uitgebreidere, commerciële variant van Unified Process is het Rational Unified Process. Naast het Unified Proces zijn er nog een aantal moderne **ontwikkelmethoden** die gezamenlijk ‘**agile**’ methoden worden genoemd. Voorbeelden daarvan zijn **Extreme Programming (XP)**, **Scrum**, **Dynamic Systems Development Method (DSDM)** en **Feature Driven Development (FDD)**. Al deze methoden hebben een aantal gemeenschappelijke uitgangspunten:

- verhelp de grootste risico's eerst, we noemen dat ook risico-gedreven;
- lever een product waar de klant om heeft gevraagd; zorg ervoor dat de vereisten (de requirements) gedurende het gehele ontwikkelproces zichtbaar zijn voor alle belanghebbenden;
- zorg zo snel mogelijk voor werkende programmatuur; de documentatie is belangrijk, maar alleen als er een werkend product ligt;
- werk als een team, niet als een verzameling individuen, zelfs als de werkdruk hoger wordt;
- maak kwaliteit een manier van denken; pas verbeteringen in product en proces meteen toe;
- de klant werkt mee in het project, waardoor we minder tijd besteden aan contractonderhandelingen;
- reageer op veranderingen in plaats van strikt een plan te volgen.

Bij al deze methoden gelden nog twee voorrangsregels:

- individuen en interacties gaan voor processen en gereedschappen;
- werkende software gaat voor uitvoerige documentatie.

6.2 Unified Process

Unified Procees (UP) onderscheidt bij het bouwen van software een viertal fasen. Omdat bij softwareontwikkeling erg veel Engels gebruikt wordt, geven we te onderscheiden fasen in het ontwikkelproces ook in die taal aan:

1. Startfase (inception phase)
2. Uitwerkingsfase (elaboration phase)
3. Bouwfase (construction phase)
4. Opleverfase (transition phase)

UP wordt aangestuurd door use-case diagrammen en een lijst met risico's. De meest bedreigende risico's moeten het eerst worden aangepakt. De use-case diagrammen staan centraal bij de ontwikkeling van het systeem. Een use-case diagram bevat een aantal use-cases. Elke use-case moet gerealiseerd worden als onderdeel van het te ontwikkelen systeem. Doordat het systeem niet in een keer eenduidig kan worden geformuleerd en ontworpen, zullen verschillende stappen bij het ontwikkelen een paar keer moeten worden overgedaan (iteratieve ontwikkeling) en zal het systeem stap voor stap gerealiseerd worden (incrementeel of evolutionair).

Use-case diagrammen vormen het hart van de ontwerptechniek van de Unified Modeling Language. In de **UML-diagrammen** kijken we op verschillende manieren naar een softwaresysteem. Het gaat daarbij om een logisch model, een implementatiemodel, een procesmodel en een aflevermodel (logical model, implementation model, process model en deployment model). Het use-case model is het verbindende vijfde element. Aan elk van de verschillende modellen kan in elke fase worden gewerkt. Het is echter gebruikelijk dat in de verschillende fasen sommige modellen wat meer en andere wat minder nadruk krijgen.

| Gedrag | Structuur | Interactie |
|---------------------|----------------|------------------|
| Use case diagram | Objectdiagram | Sequentiediagram |
| Activiteitendiagram | Klassendiagram | |
| Toestandsdiagram | | |

Tabel 6.1 De 14 verschillende UML-diagrammen zijn in drie groepen te verdelen. In deze figuur zijn slechts de in dit boek gebruikte diagrammen opgenomen

In dit boek bekijken we uitsluitend de linkerkant van figuur 6.1. De rechterkant met het implementatiemodel en het aflevermodel wordt vooral gebruikt bij de afronding van het ontwikkelingstraject.

Het **logische model** omvat de **klassendiagrammen** en **objectdiagrammen**. Met deze diagrammen proberen we een zo juist mogelijk model te maken van het probleemgebied waarin het systeem moet worden gerealiseerd. Deze diagrammen kennen verschillende niveaus van detaillering, zoals we in de voorgaande hoofdstukken hebben gezien. In eerste instantie zullen we alleen de klassen van een naam voorzien. We kunnen dan de associaties, de verbindingen tussen de klassen, toevoegen. Pas later voorzien we de klassen van attributen en operaties.

Bij het **procesmodel** beschrijven we het gedrag van het systeem dat ontwikkeld moet worden. Hierbij zijn de sequentiediagrammen een belangrijk hulpmiddel.

De Unified Modeling Language is een hulpmiddel bij alle fasen in een ontwikkelproces. UML kan bij elke softwareontwikkelingsmethode in alle fasen toegepast worden. Dat geldt dus ook voor de hier toegepaste methode Unified Process.

6.3 Startfase (inception phase)

Elk project start in een **inception phase** waarin alle betrokkenen, de zogenaamde **stakeholders**, hun goedkeuring aan het project geven.

Terzijde

In de bedrijfsvoering van grote ondernemingen wordt steeds vaker een onderscheid gemaakt tussen de **shareholders** en de stakeholders. De shareholders zijn de aandeelhouders. Hun belang is niet het bedrijf, maar vooral de winst die het bedrijf weet te behalen. Het zal de aandeelhouders een zorg zijn hoe dat resultaat wordt behaald. De stakeholders hebben een tegengesteld belang. Zij willen vooral dat hun onderneming effectief en efficiënt werkt. Daarvoor moeten natuurlijk voldoende middelen zijn. Als het bedrijf goed draait, zal er vanuit de visie van de stakeholders vanzelf een winstmaximalisatie ontstaan. Softwareontwikkeling wordt ingezet, uitgevoerd en aangestuurd door de stakeholders. De shareholders zullen vooral opletten dat de nieuwe software leidt tot verhoging van de winst.

In de inception phase moet de afbakening van het project vastgelegd worden. We bepalen de omvang van het project (de scope), leggen de doelstellingen vast en zorgen ervoor dat er voldoende middelen zijn om het project te kunnen starten. In deze fase worden ook de bestaande processen en procedures bestudeerd en in kaart gebracht.

Tijdens het werken aan het systeem in de inception phase wordt een visiedocument gemaakt, een voorlopig use-case model en een aantal activiteitendiagrammen om de voornaamste processen te beschrijven. Daarnaast wordt een eerste projectplan opgesteld en een business case. In een projectplan staat onder andere wie wat wanneer gaat doen. Het opstellen van een projectplan met bijbehorende planning kan zichtbaar maken waar eventuele knelpunten zitten bij het realiseren van het systeem. Zo kan een tekort aan hulpmiddelen of (geschikte) mensen voor de uitvoering van het project worden signaleerd. De business case beschrijft welke baten de organisatie verwacht van het te realiseren systeem en welke kosten daar tegenover staan.

Ook wordt een begin gemaakt met het opstellen van een lijst met risico's die het project bedreigen. De lijst is meestal gesorteerd van meest bedreigend naar minst bedreigend. In deze lijst leggen we

vast hoe groot de kans is en hoe groot de gevolgen zijn van een bepaalde calamiteit. Soms worden in deze fase ook al prototypes gerealiseerd, meestal om een bewijs van technische haalbaarheid aan te tonen ('**proof of concept**').

Terzijde

Laten we een softwareproject weer eens met de bouwwereld vergelijken. In Amsterdam is men begonnen met de aanleg van de noordzuidlijn van de metro. Deze loopt van Amsterdam-Noord onder het IJ en het Centraal Station verder door de historische binnenstad van Amsterdam. Bij elke passage wordt een risicoprofiel gemaakt. Er worden vragen gesteld als:

Stel dat we de fundering van het Centraal Station beschadigen.

1. Hoe groot zal dan de schade zijn?
2. Is deze schade wel te repareren? (Het Centraal Station is een rijksmonument.)
3. Hoe lang duurt het voor deze schade hersteld kan zijn?
4. Hoeveel vertraging loopt het hele project dan op?
5. Hoeveel mensen moeten we extra inschakelen voor het herstel?
6. Zijn deze mensen in Nederland beschikbaar of moeten we nu al op zoek gaan naar specialisten? (voor het geval dat...)
7. Moeten we dit risico verzekeren of kunnen we de meerkosten zelf dragen? (Is dit risico wel verzekeraar?)

Zo kunnen we bij elk onderdeel een aantal vragen formuleren en op zoek gaan naar de antwoorden. Het is een complex proces, maar als we dit niet doen, weten we helemaal niet waar we aan toe zijn.

Soms wordt tijdens deze fase al een eerste versie van het op te leveren systeem gerealiseerd. Als dat in het projectplan is opgenomen, wordt het prototype niet 'weggegooid' maar juist incrementeel (stap-voor-stap) verder opgeleverd.

De nadruk van de inception phase ligt dus op het vaststellen van de omvang van het te ontwikkelen systeem met de belangrijkste

vereisten (requirements) en op het analyseren van de betrokken werkprocessen. Bij de overgang naar de volgende fase zijn de volgende producten helemaal of in conceptvorm gereed:

- een visiedocument waarin de grenzen van het systeem worden aangegeven;
- een financiële onderbouwing van het project uitgewerkt in de vorm van een business case;
- een lijst met mogelijke risico's die het project kunnen schaden;
- eventueel enkele 'proof-of-concept' prototypes
- een of meerdere activiteitendiagrammen waarin de betrokken bedrijfsprocessen zichtbaar in worden gemaakt;
- een voorlopig use-case diagram.

Bij de opgeleverde producten moeten we erop letten dat deze op elkaar zijn afgestemd en ook dat ze geen fouten bevatten, of onderlinge tegenstrijdigheden. Die controle dient gedurende de gehele projectduur te blijven bestaan.

6.4 Uitwerkingsfase (elaboration phase)

Nadat de producten van de inception phase zijn opgeleverd, gaat het project over naar de **elaboration phase**. Nu moeten de vereisten (requirements) in detail worden vastgelegd. Het streven is om zeker de voor het systeem belangrijkste requirements te vinden en vast te leggen. In deze fase werken we met een klassendiagram om het model van het te realiseren systeem te bouwen. Er wordt vastgelegd wat er gebouwd gaat worden. Tijdens de elaboration phase worden al de eerste tests uitgevoerd op het ruw ontwikkelde systeem.

In nauwe samenwerking met de klant vullen we de lijst met requirements verder aan en werken we deze uit. Van elke requirement stellen we bijvoorbeeld vast of deze voldoende duidelijk is omschreven en ook prioriteit heeft om verder uitgewerkt te worden in een use-case. Daarnaast maken we uit deze nieuwe requirements de eerste ontwerpen van klassendiagrammen om de belangrijkste klassen zichtbaar te maken.

Tijdens de elaboration phase komen diverse producten tot stand:

- een conceptueel klassendiagram (de klassen zijn benoemd, maar de attributen en operaties nog niet);

- sequentiediagrammen om de interactie tussen verschillende objecten zichtbaar te maken;
- een herzien visiedocument;
- een bijgewerkte planning;
- een herziene risicolijst.

Doordat er toenemende kennis is over wat het systeem moet kunnen, moeten we zowel de documenten als de diagrammen uit de inception phase aanpassen. Het is gebruikelijk het visiedocument in de elaboration phase nog aan te vullen en hier en daar te wijzigen. Door het opstellen van de risicolijst en het uitwerken van de aangegeven oplossingen zullen sommige risico's verdwijnen of minder bedreigend worden. Ook kunnen er nieuwe risico's zijn ontdekt. De lijst met risico's wordt daarom ook aangepast. Al die veranderingen hebben ook gevolgen voor de planning.

Aan het eind van de elaboration phase is de reikwijdte van het te ontwerpen systeem veel duidelijker dan na de inception phase. Gewoonlijk wordt ervan uitgegaan dat 80 procent van alle use-cases nu bekend zijn en dat ook de meeste requirements wel zijn uitgewerkt. Uiteraard moet alle documentatie op elkaar aansluiten. De kleinste tegenspraak in documenten en diagrammen leidt onherroepelijk tot problemen. De onderlinge samenhang (consistentie) wordt tijdens de elaboration phase steeds belangrijker.

De requirements worden in drie groepen gesplitst. De functionele requirements zijn van direct belang. Als aan deze vereisten niet voldaan wordt, kan het product niet werken. De niet-functionele requirements zijn aanwezige vereisten. Er moet wel aan voldaan worden, maar ze spelen geen rol in de software. Pseudo-requirements zijn geen echte vereisten. Iedereen vindt ze nodig, maar de functie is onduidelijk.

Om een brood te snijden heb je een mes nodig. Een mes is een functioneel requirement voor het brood snijden. Het snijden gaat eenvoudig op een broodplank. Kunnen we echter zonder broodplank het snijproces niet uitvoeren? Een broodplank is dus een niet-functioneel requirement. De gesneden boterhammen leggen we in een broodmand. Dat vinden we normaal en stellen we dus als vereiste, maar waarom eigenlijk? We kunnen de boterhammen toch net zo goed op een bordje leggen? Een broodmand is hier een pseudo requirement.

De nadruk van de elaboration phase ligt dus op het vaststellen van de belangrijkste requirements en een voorlopig conceptueel klassendiagram. Bij de overgang naar de volgende fase zijn de volgende producten helemaal of in conceptvorm gereed:

- een definitief visiedocument waarin de grenzen van het systeem worden aangegeven;
- een herziene lijst met mogelijke risico's die het project kunnen schaden;
- een of meer use-case diagrammen;
- een lijst met requirements, uitgesplitst in functionele, niet-functionele en pseudo-requirements;
- een voorlopig conceptueel klassendiagram.

6.5 Bouwfase (construction phase)

Terwijl in de inception en elaboration phases de nadruk ligt op het wát, verschuift dit bij de **construction phase** naar het hóé. Wat het systeem moet kunnen, ligt nu grotendeels vast in een aantal documenten. Hoe dit gerealiseerd moet worden, krijgt in deze fase de nadruk. In de construction phase worden de requirements en modellen uitgewerkt tot een technisch ontwerp. In deze fase is het belangrijk om vast te houden aan de kwaliteit van het ontwerp. De nadruk ligt in deze fase dan ook bij het opleveren van een werkend systeem. Daarbij kan door middel van testen ook worden aangetoond dat de gewenste functionaliteit is gerealiseerd.

Naast het ontwerpen wordt er gelijktijdig aan de programmacode gewerkt. Programmeren en ontwerpen wisselen elkaar voortdurend af. Een aantal diagramtechnieken zoals sequentiediagrammen en toestandsdiagrammen gebruiken we om de dynamische aspecten van het systeem zichtbaar te maken.

Het conceptuele klassendiagram wordt tijdens deze fase omgebouwd tot een klassendiagram waar voor elke klasse ook de operaties en attributen zijn aangegeven. Met behulp van dit klassendiagram kan elke klasse die uitgemodelleerd is, in programmacode worden omgezet. Voor de overzichtelijkheid plaatsen we elke belangrijke klasse op een aparte bladzijde. We vermelden bij elke klasse de attributen en operaties. Als er in het overleg met de klant beperkingen zijn vastgesteld bij deze klasse, geven we ook deze constraints weer.

In deze fase werken we aan het ontwerpen van de schermopbouw en de navigatie door het systeem. Hoewel we willen dat klassen onafhankelijk van een programmeertaal gedefinieerd kunnen worden, blijkt dat elke programmeertaal toch zijn eigen declaratiemethodes kent. Zo zal in de ene taal een String gedefinieerd kunnen worden, terwijl hetzelfde gegeven in een andere taal als Char gedeclareerd moet worden. In het klassendiagram moeten we de relatie aanbrengen met de klassen uit de programmeertaal waarin het systeem wordt gerealiseerd. Bij moderne programmeertalen kunnen we daarbij gebruik maken van een programmabibliotheek, een **API (application programming interface)**.

De focus van de construction phase ligt dus op het realiseren van een technisch klassendiagram en een bijna volledig werkend systeem. Bij de overgang naar de volgende fase zijn de volgende producten helemaal of in conceptvorm gereed:

- een technisch klassendiagram, met voor elke klasse vastgelegd hoe deze wordt ontworpen (relatie met API) en welke attributen en operaties deze heeft; bovendien wordt per klasse aangegeven welke beperkingen (constraints) voor die klasse gelden;
- een herziene lijst met mogelijke risico's die het project kunnen schaden;
- een lijst met tests en testresultaten, waarbij is aangegeven welke requirements of constraints worden afgedekt;
- een voorlopige gebruikershandleiding voor het op te leveren systeem;
- een bijna volledig werkend systeem.

6.6 Opleverfase (transition phase)

In de laatste fase wordt de software in definitieve vorm feitelijk opgeleverd aan de klant. Tijdens deze fase worden de ontdekte defecten opgelost, worden de handleidingen afgerond en definitief gemaakt, en wordt uiteindelijk het project geëvalueerd.

De nadruk van de **transition phase** ligt op het realiseren en netjes afleveren van de applicatie, waarbij aangetoond moet worden of de oplevering binnen het afgesproken budget en met de gewenste functionaliteit is gebeurd. Aan de klant worden in ieder geval de volgende producten opgeleverd:

- een gebruikershandleiding voor het gebruik van het systeem;
 - een volledig werkend systeem.
- Afhankelijk van de afspraken met de klant wordt een aantal documenten ook aan de klant opgeleverd. Bij oplevering van het systeem aan de klant moeten in ieder geval intern gearcheveerd worden:
- een volledig functioneel ontwerp, met daarin het visiedocument, de definitieve lijst van requirements, het conceptuele klassendiagram enzovoort;
 - een volledig technisch ontwerp met daarin een technisch klassendiagram, enzovoort;
 - een volledige lijst met testen, waarbij aangegeven is welke requirements of constraints worden afgedekt, en de testresultaten.

6.7 De case PartyCups – het inlog subsysteem als voorbeeld

In het project rondom de case PartyCups hebben we in hoofdstuk 2 de use-cases bestudeerd en in hoofdstuk 3 de activiteitendiagrammen. Het gaat bij de inception phase natuurlijk ook om de afstemming en onderlinge consistentie van de diagrammen. Dus als er in de activiteiten diagrammen zijn genoemd waarvoor de use-cases niet aanwezig zijn, dan moeten deze als nog worden opgesteld.

Miep Gazouw heeft in overleg met Jan en Marloes voorgesteld om een CRM-systeem te laten ontwerpen voor PartyCups. Een mogelijkheid die tijdens dit overleg ter sprake is gekomen, is de vertegenwoordigers zelf orders te laten invoeren. Dit betekent wel iets meer werk voor de vertegenwoordiger, maar levert een verminderde werkdruk op bij de administratie. Om dit mogelijk te maken, kan er gekozen worden voor directe invoer tijdens (of direct na) een klantcontact of voor invoer van de gegevens aan het eind van elke werkdag.

Beide oplossingen moeten worden onderzocht. Wel is duidelijk dat de applicatie beschermd dient te worden tegen onbevoegd gebruik. Dit betekent voor PartyCups een kleine cultuurverandering: iedere medewerker zal worden opgenomen in de bij de applicatie behorende database en krijgt een toegangscode

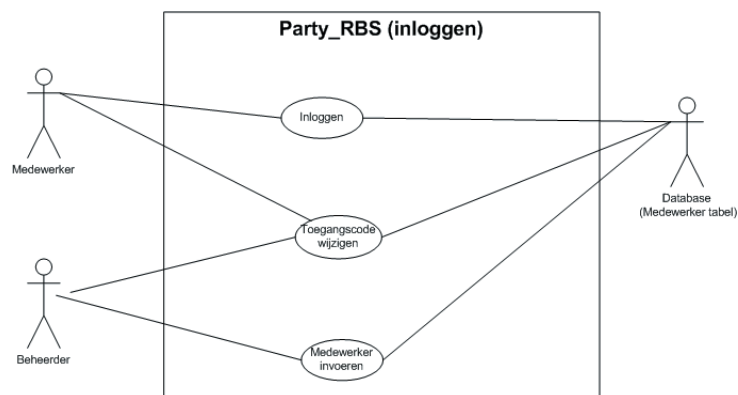
toegewezen. Het beheer van de toegangscode voor elke medewerker ligt in eerste instantie bij de medewerker zelf. Er komt echter ook een beheerder die medewerkers kan toevoegen, eventuele vergeten toegangscodes kan terugzetten, enzovoort. Besloten wordt om Jan beheerder te maken van het te ontwikkelen systeem.

Bij de discussies rond het nieuwe systeem is duidelijk dat medewerkers eerst zullen moeten inloggen, voordat zij de applicatie kunnen gebruiken. Het inloggen geschiedt op basis van de toegewezen gebruikersnaam en de daarbij behorende toegangscode. Jan is verantwoordelijk voor het opnemen van nieuwe medewerkers, het initieel toewijzen van een toegangscode, het opnieuw toewijzen van een toegangscode en het blokkeren van een medewerker. Afgesproken is dat Jan geen medewerker kan verwijderen uit de database omdat de applicatie deze gegevens nog nodig heeft. Er kunnen immers nog orders in het systeem staan die verwijzen naar de betrokken medewerker.

Uitwerking

Uit de beschrijving van de case PartyCups blijkt dat men bij de te ontwerpen applicatie uitgaat van een login-onderdeel. In een use-case diagram kunnen we de casebeschrijving over het inloggen weergegeven zoals in figuur 6.2.

Figuur 6.2
PartyCups subsysteem
'inloggen'



Uit het use-case diagram blijkt dat een medewerker ook de toegangscode kan wijzigen. Dat dit alleen de *eigen* toegangscode

is, blijkt niet uit het diagram maar moet worden vastgelegd met commentaar bij het diagram of in de use-case template. De use-case 'Medewerker invoeren' zal vermoedelijk verplaatst worden naar een ander subsysteem, namelijk bij het beheer van de tabel Medewerkers. We werken daarom nu alleen de use-cases 'Inloggen' en 'Toegangscode wijzigen' verder uit.

| | |
|------------------------|---|
| Naam | Inloggen |
| Versie | 1.0 |
| Actor | Medewerker, Database (tabel Medewerker) |
| Preconditie | Systeem is beschikbaar voor invoer |
| Beschrijving | Medewerker voert naam in Medewerker voert toegangscode in Drukt op OK-knop om toegelaten te worden, systeem toont hoofdscherm |
| Uitzonderingen | Toegang geweigerd in verband met onjuiste gegevens, melding toegang geweigerd en opnieuw inlogscherm tonen Gegevens incompleet, melding gegevens incompleet, aangeven welke gegevens ontbreken, terug naar inlogscherm |
| Niet-functionele eisen | Inloggen moet te realiseren zijn in 2 minuten |
| Postconditie | Medewerker is ingelogd, systeem wacht op volgende actie van medewerker |

Tabel 6.2 Use-case template 'Inloggen'

| | |
|------------------------|--|
| Naam | Toegangscode wijzigen |
| Versie | 1.0 |
| Actor | Medewerker, Beheerder, Database (tabel Medewerker) |
| Preconditie | Medewerker is ingelogd, systeem is beschikbaar voor invoer |
| Beschrijving | <p>Medewerker selecteert Toegangscode wijzigen</p> <p>Medewerker is beheerder dan</p> <p>Voert medewerker naam in</p> <p>Voert nieuwe code in</p> <p>Voert nieuwe code in controle veld in</p> <p>Anders</p> <p>Voert oude code in</p> <p>Voert nieuwe code in</p> <p>Voert nieuwe code in controle veld in Drukt op OK-knop om wijzigingen op te slaan systeem toont melding 'toegangscode gewijzigd'</p> |
| Uitzonderingen | Gegevens incompleet, melding gegevens incompleet, aangeven welke gegevens ontbreken, terug naar invoer scherm |
| Niet-functionele eisen | Wijzigen toegangscode moet te realiseren zijn in 5 minuten |
| Postconditie | Toegangscode is gewijzigd, systeem wacht op actie medewerker |

Tabel 6.3 Use-case template 'Toegangscode wijzigen'

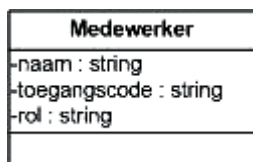
De informatie die in de use-case 'Wijzigen toegangscode' wordt vermeld, is veel uitgebreider dan bij de use-case 'Inloggen'. Er zijn geen twee actieve actoren, maar een van de twee kan de toegangscode wijzigen. De beheerder kan dit vanuit de beheerdersrol. De use-case kan ook gesplitst worden in twee use-cases om de use-case te vereenvoudigen. Dan verandert ook het bijbehorende use-case diagram.

We kunnen wel aangeven welke attributen en operaties opgenomen moeten worden bij de medewerker. Uiteraard ligt het voor de hand om naam en toegangscode als attributen bij de medewerker op te nemen. Door het toevoegen van een aparte beheerder in het systeem, zal daarvoor ook een attribuut aan de medewerker moeten worden toegevoegd. Dit attribuut kan rechtstreeks aangeven of de medewerker de beheerdersrol vervult. Het attribuut kan ook indi-

rect zijn, door de medewerker te laten verwijzen naar een groep waartoe deze behoort. Dit laatste komen we ook tegen bij besturingssystemen waar iemand bijvoorbeeld behoort tot de groep gast of administrator (Windows) en afhankelijk daarvan aanzienlijk minder of meer rechten (mogelijkheden) heeft.

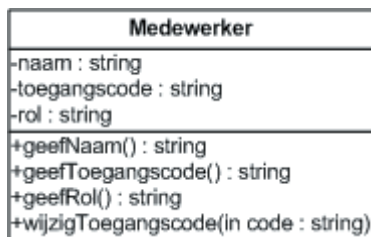
Voor het moment gaan we ervan uit dat de rol van beheerder direct is opgenomen als attribuut bij de medewerker als rol. Het klassendiagram met attributen zoals afgeleid uit de use-case voor de klasse Medewerker ziet er uit zoals in figuur 6.3.

Figuur 6.3
De klasse Medewerker
met attributen



Elke use-case van het systeem wordt zo doorlopen en we kunnen op deze manier veel van de attributen en operaties terugvinden.

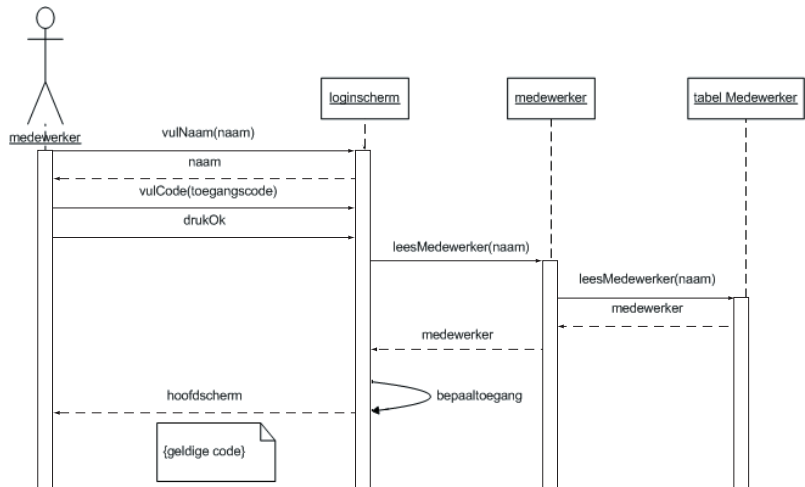
Figuur 6.4
De klasse Medewerker
met attributen en
operaties



Het toevoegen van operaties gebeurt ook met behulp van de use-case tekst. We zien uit de beschrijving van de case PartyCups dat zowel de naam van een medewerker als zijn of haar toegangscade opgevraagd moeten kunnen worden. Ook de rol moet gecontroleerd kunnen worden.

Soms wordt voor het vinden van de operaties gekozen om een *sequencediagram* te maken van de beschrijving uit de use-case template. Daarbij wordt in het stadium van de systeemanalyse uitgegaan van een enkel grafisch toegangsscherm dat de invoer van de actor naar de applicatie regelt en de uitvoer aan de actor toont, de grafische gebruikersinterface (of graphic user interface, GUI). Zo wordt ook de database opgenomen als passieve actor in het diagram. We krijgen dan voor het inloggen het sequencediagram van figuur 6.5.

Figuur 6.5
Het sequentiediagram
'Inloggen'



Volgens het sequentiediagram van figuur 6.5 lijkt het zo te zijn dat het loginscherm door middel van de 'bepaaltoegang' ook alle controle uitvoert. In werkelijkheid zal dit een apart object zijn, mogelijk zelfs een object van de klasse Medewerker. Het diagram geeft echter wel een indruk van wat het inloggen voor het systeem inhoudt, waarbij nog geen rekening is gehouden met de uitzonderingen die kunnen optreden.

Uit het scherm van figuur 6.6 blijkt dat de medewerker het inloggen alleen kan annuleren door het hele scherm weg te klikken. Dit zou ook anders kunnen, bijvoorbeeld door in het scherm een knop

Figuur 6.6
Het inlogscherm van het
RBS PartyCups



Annuleren op te nemen. Of dat wel of niet nodig is, behoort meer tot de ontwerpbeslissingen en zoals al eerder is aangegeven houden we ons daar in dit boek niet mee bezig.

6.8 Samenvatting

In dit hoofdstuk is een overzicht gegeven van het Unified Process als moderne iteratieve ontwikkelingsmethode. Aangegeven is welke vier fasen UP onderscheid bij het bouwen van softwaresystemen (inception phase, elaboration phase, construction phase, transition phase) en welke producten (artefacts) in elke fase worden opgeleverd. Een aantal van deze producten zijn we in voorgaande hoofdstukken al tegengekomen, een aantal andere nog niet. Ten slotte hebben we in vogelvlucht een functionaliteit van het PartyCups relatiebeheersysteem bekeken. We hebben gezien hoe de use-case het ontwikkelproces aanstuurt en hoe vanuit de use-case template een klassendiagram, een sequentiediagram en uiteindelijk het applicatieonderdeel wordt gerealiseerd.

Opgaven

1. Zoek op het Internet uit wat je kunt vinden over het Unified Process. Zijn er ook boeken over?
2. Welke voordelen zijn er als de gebruiker (klant) van een te ontwikkelen softwaresysteem vanaf het begin sterk betrokken is bij de ontwikkeling?
3. Vereenvoudig de use-case template 'Toegangscode wijzigen'. Werk deze alleen uit voor een medewerker.
4. Als de klasse Medewerker moet bepalen of een object van het type Medewerker toegang krijgt tot het systeem, wat moet er dan aan de klasse Medewerker worden toegevoegd?
5. Noem een aantal voordelen van een papieren prototype ten opzichte van een prototype met de computer. Welke nadelen kunnen er zijn?

6. Beschrijf de business case van PartyCups bij het realiseren van het relatiebeheersysteem.
7. Hoort het samenstellen van de catalogus (zie hoofdstuk 3) volgens jou nog onder dit relatiebeheersysteem?
8. Welke niet-functionele requirements kom je tegen bij de use-case templates 'Inloggen' en 'Toegangscode wijzigen'?
9. Stel dat we van PartyCups de opdracht krijgen het systeem in Java te realiseren, met als argument dat het later ook via Internet bruikbaar moet zijn. Kun je aangeven of dit een goed argument is om te kiezen voor de taal Java? Kun je ook een risico noemen uitgaande van je eigen kennis van Java?
10. In wat voor type requirement leg je vast dat het systeem moet werken met een in Access gerealiseerde database?

Sequentiediagrammen – Het gedrag in modellen en de communicatie tussen objecten

7.1 Inleiding

In de UML-modellen gaat het steeds over gedrag. Nadat we in vorige hoofdstukken bepaald hebben welke objecten met elkaar communiceren, geven we in dit hoofdstuk aan wanneer en hoe ze dat doen. We geven de volgorde van de berichten tussen klassen aan en ook of er antwoord verwacht wordt. Het hulpmiddel dat UML hiervoor aanbiedt is het **sequentiediagram**. Sequentiediagrammen behoren samen met collaboratiediagrammen (die we niet behandelen in dit boek) tot de interactiediagrammen van UML.

7.2 Het doel van een sequentiediagram

In sequentiediagrammen geven we aan hoe een proces verloopt. Elke handeling wordt ingetekend; elke keer dat een object aangesproken wordt, nemen we op in het sequentiediagram. Daarmee lijkt het veel op het activiteitendiagram. Het verschil is dat we ons bij een activiteitendiagram meer richten op het menselijk handelen, terwijl we nu over programmaonderdelen praten. We zijn nu dus meer met het programmaontwerp bezig. We hebben al klassen gevormd, we weten op welke wijze de klassen contact met elkaar hebben. Nu wordt het tijd om vast te leggen hoe de berichtensroom tussen de klassen verloopt. Dat lijkt heel simpel, maar we moeten ook rekening houden met alle mogelijke uitzonderings-situaties.

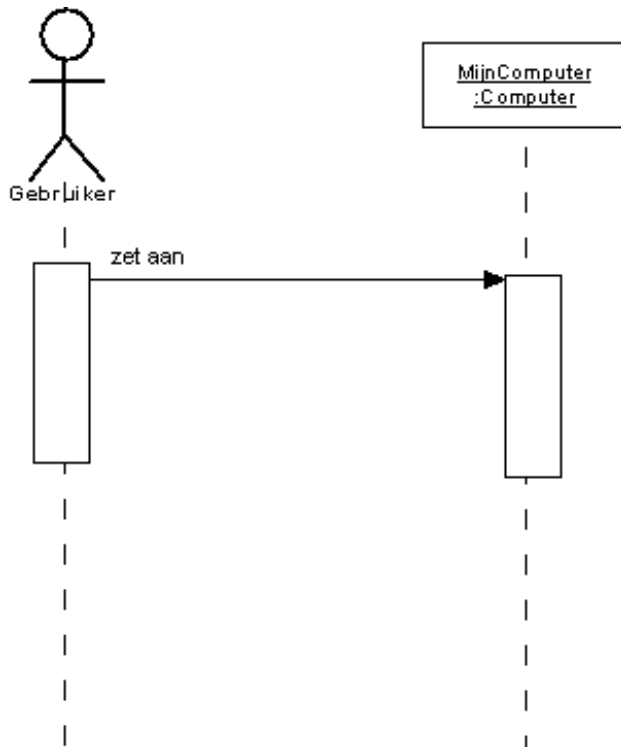
Terzijde

Stel dat je wilt vastleggen dat een printopdracht van de computer uitgevoerd wordt. Normaal gesproken is dat een muis-klik, waarna het besturingssysteem de printopdracht 'spoolt' en daarna na de printer stuurt. Vervolgens produceert de printer de afdrucken. Simpel dus. Er kan echter heel veel fout gaan. Zo kan de computer geheugenproblemen hebben waardoor de printopdracht niet 'gespoold' kan worden, het contact met de printer kan verbroken zijn, de printer kan zijn uitgezet of off-line staan, het papier of de toner kan op zijn en er kan nog veel meer mis zijn. Bij elke fout hoort een foutmelding. Alle problemen en hun oplossingen kunnen in een of meer sequentiediagrammen worden gezet.

7.3 De notatie van onderdelen van het sequentiediagram

Een sequentiediagram kent weer een hele reeks nieuwe symbolen. We gaan deze bekijken aan de hand van het voorbeeld in figuur 7.1.

Figuur 7.1
Een voorbeeld van een sequentiediagram



Aan de bovenzijde van de figuur treffen we een paar bekende symbolen. Het draadpoppetje aan de linkerkant is een actor. Deze actor, met de naam Gebruiker, doet hetzelfde als in een use-case diagram. Hij neemt het initiatief, hij zet iets in gang. Daarnaast staat het rechthoekje dat we kennen als klasse of object in een klassendiagram. In dit geval gaat het om het object MijnComputer van de klasse Computer. De schrijfwijze van object en klasse zijn hetzelfde als in het objectdiagram. Onder zowel actor als object bevinden zich stippellijnen. Deze worden levenslijnen (lifelines) genoemd. Zolang de actor of het object beschikbaar is, loopt de stippellijn door. Het rechthoekige blokje dat op de levenslijn getekend staat, geeft weer dat de actor of het object het proces bestuurt. De pijl vertelt wat er gedaan wordt. In figuur 7.1 zet de actor Gebruiker dus MijnComputer aan. Verticaal staat de verlopen tijd vermeld. Bij tijdkritische processen kan daar een schaal ingevuld worden. Wij doen dat nog niet. Wij proberen alleen alle stappen in de juiste volgorde aan te geven.

7.4 De berichten

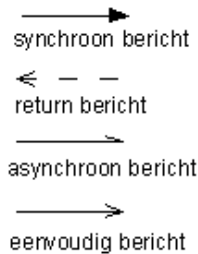
De pijl in het sequentiediagram vertelt ons niet alleen wat er gedaan wordt. Een pijl met een gesloten driehoek als pijlpunt (dus zoals in figuur 7.1) vertelt dat het om een **synchronous bericht** (**synchronous message**) gaat. Bij een synchronous bericht neemt het bericht de sturing over tot het bericht voltooid is. Daarna volgt er een terugmelding dat het resultaat bereikt is. In dit geval zet de gebruiker MijnComputer aan. Het bericht zorgt ervoor dat alle starthandelingen van de computer uitgevoerd worden en na afloop daarvan, dus bijvoorbeeld na het Windows-melodietje, krijgt gebruiker de besturing weer terug.

Soms is het zinnig om aan te geven dat de besturing vrijgegeven wordt. Meestal is dat vanzelfsprekend. Als we de reactie willen laten zien, kunnen we gebruikmaken van de Return-pijl. Dit is een gestippelde pijl met een normale pijlpunt die terugverwijst naar het object waar het bericht vandaan kwam.

Naast **synchrone berichten** bestaan er ook **asynchrone berichten**. Hier zendt het actieve object een bericht, maar wacht niet op een reactie. Er kan direct een volgend bericht gestuurd worden. Asynchrone berichten worden aangegeven met een halve pijlpunt.

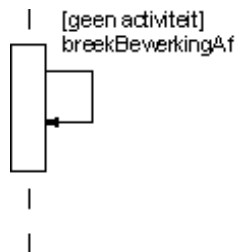
Als we echt niet weten of een bericht synchroon of asynchroon is, kunnen we een gewone open pijlpunt gebruiken. Deze betekent dat niet de ontwerper (wij dus), maar de uitvoerder, de programmeur bepaalt hoe het bericht uitgevoerd zal worden.

Figuur 7.2
De verschillende berichten



In een aantal gevallen kan een bericht terugverwijzen naar de eigen activering. Dat geven we aan met een gebogen pijl.

Figuur 7.3
Terugverwijzende berichten



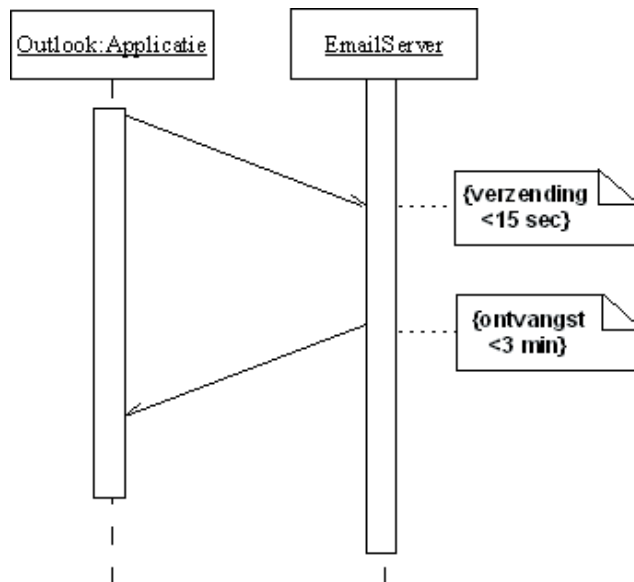
7.5 Tijdgebonden overgangen

Normaal gesproken worden alle berichten als horizontale lijnen aangegeven. Eigenlijk is dit vreemd als we bedenken dat we tijd als verticale as in het activiteitendiagram aangeven. We tekenen dus dat de activering door een actor of object tijd kost, maar het sturen van berichten niet. Vaak is het ook zo dat de bewerking door een object veel meer tijd kost dan het zenden of ontvangen van een bericht. Wat we tekenen is dus niet helemaal juist, maar de afwijking van de werkelijkheid is slechts gering. Als we willen aangeven dat het zenden van een bericht wel tijd kost, dan kunnen we dit doen door de pijl niet horizontaal, maar schuin naar beneden gericht te tekenen. Het is nu duidelijk dat dit bericht echt tijd kost. We doen dit vooral in applicaties die echt tijdkritisch zijn. We geven dan

niet alleen aan dat het zenden en ontvangen van een bericht tijd kost, maar ook hoeveel tijd het kost. Eventueel geven we ook de grenswaarden aan. We melden dan de minimale en/of maximale tijd. **Tijdgebonden berichten** vinden we in veel **embedded systems**. Dit zijn apparaten met een eigen ingebouwd besturingssysteem gericht op de functie van het apparaat. Denk bijvoorbeeld aan een verkeerslicht dat op basis van registratie van verkeersbewegingen een procesgang start.

Als we kunnen benoemen hoe lang de toegestane tijd is, kunnen we dit toevoegen aan ons sequentiediagram. Een dergelijke voorwaarde noemen we een constraint. Deze plaatsen we tussen accolades.

Figuur 7.4
Constraints bij tijd-
gebonden berichten

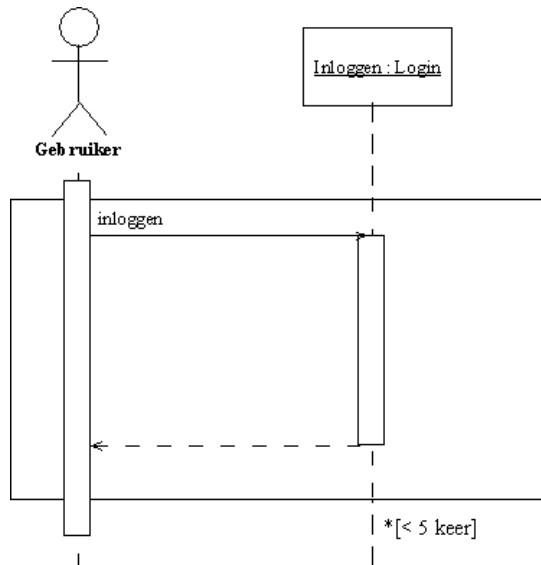


7.6 Iteraties

Iteraties zijn herhalingen: we doen iets vaker dan één keer. Soms staat van tevoren vast hoe vaak we iets doen, soms gaan we door met herhalen tot een bepaalde toestand bereikt is. Als we herhaling in een sequentiediagram willen aangeven, tekenen we een rechthoek om de te herhalen routine en geven rechts onder de rechthoek de voorwaarde (de **guard-conditie**).

Figuur 7.5

Er kan een aantal pogingen gedaan worden om in te loggen; rechts-onder staat tussen rechte haken de guardconditie



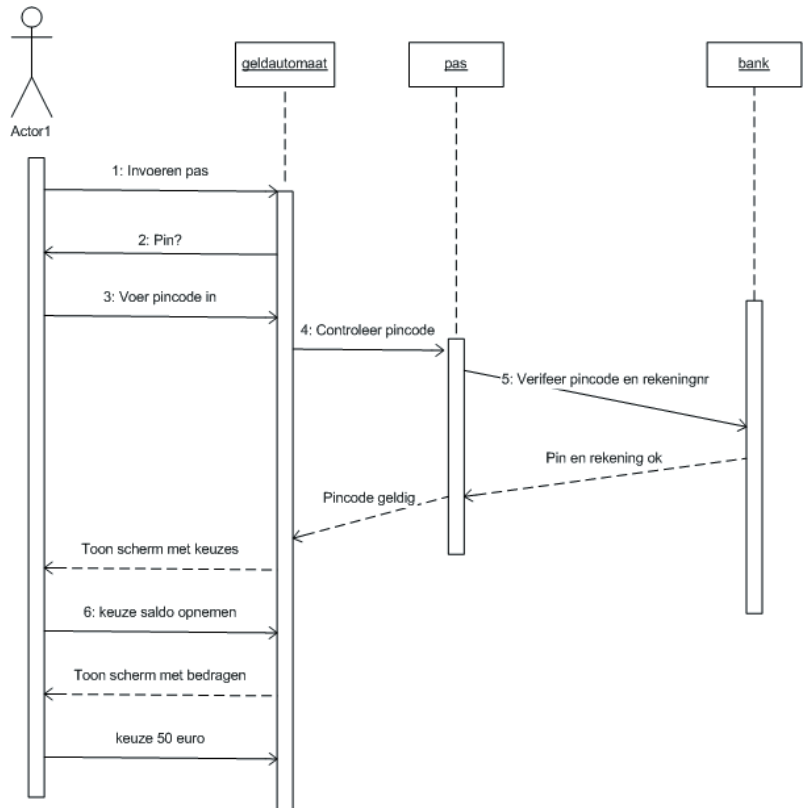
7.7 Voorbeelden van sequentiediagrammen

Bij het opzetten van een sequentiediagram maken we gebruik van de use-case beschrijving of van enkele door de klant aangegeven scenario's. We willen immers de communicatie met het te ontwerpen systeem eenduidig vastleggen. Het kan gebeuren dat we tijdens het opstellen van een sequentiediagram ontdekken dat de klant onvolledige of soms zelfs onjuiste, tegenstrijdige informatie heeft verstrekt. Dit heeft niets te maken met onwil van de klant, maar vooral met de moeite om precies te formuleren wat er werkelijk gebeurt. Een beschrijving die voor mensen hanteerbaar is, bijvoorbeeld een procedurebeschrijving, zit vaak nog vol met hiaten voor een te ontwerpen systeem.

Bij het maken van de sequentiediagrammen streven we naar een zo goed mogelijke weergave van de werkelijkheid. Elk sequentiediagram moet in ieder geval technisch correct zijn. Ook is het sequentiediagram bedoeld om de communicatie met de klant te ondersteunen. Het vaststellen dat de verstrekte informatie onjuist of onvolledig is, levert voldoende materiaal op om nieuwe vragen aan de gebruiker te stellen. Ook het maken van sequentiediagrammen gebeurt dus iteratief.

Bij elk bankkantoor in Nederland is het tegenwoordig mogelijk om bij een automaat geld op te nemen. De geldautomaat staat de hele dag te wachten op een klant. De klant kan zijn of haar pas invoeren, waarna de automaat zal vragen om de pincode die bij de kaart en rekening van de gebruiker hoort. Nadat de pincode gevalideerd is en de gebruiker toegang is verleend tot het systeem, wordt een menu getoond met twee mogelijkheden, geld opnemen en saldo tonen. Als de gebruiker daarna kiest voor geld opnemen worden een aantal opties getoond met verschillende bedragen. Na de keuze van een bedrag volgt het uitwerpen van de pas en daarna het uitwerpen van het gekozen bedrag. Ten slotte keert het apparaat terug in de wachttoestand.

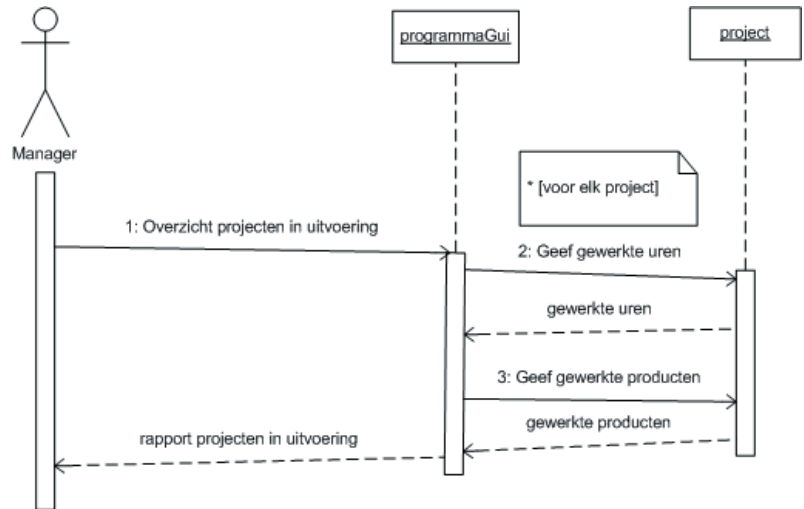
Figuur 7.6
Sequentiediagram met
deel van een geldopname
bij een geldautomaat



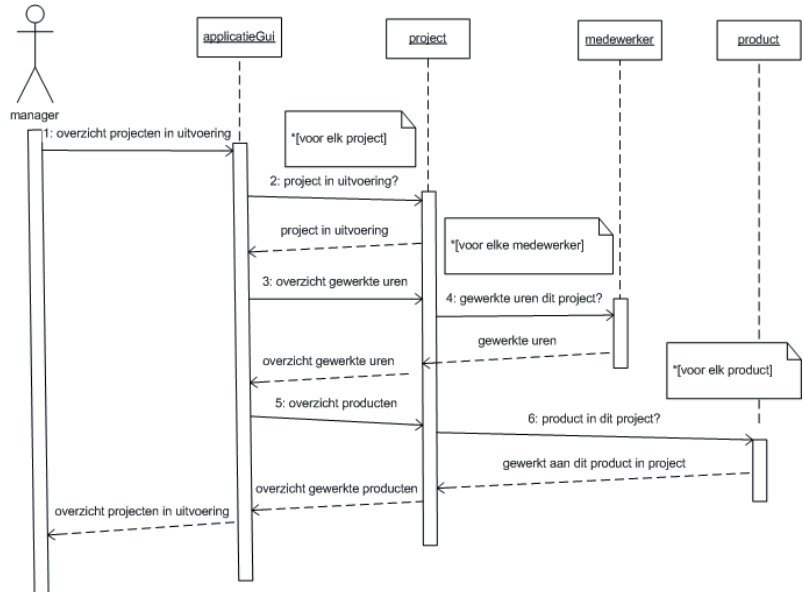
Een manager wil een overzicht hebben van alle projecten die in uitvoering zijn. Hij wil weten wie er bij betrokken zijn, hoeveel uur er aan gewerkt is en aan welke producten. In figuur 7.7 zie je

een mogelijke uitwerking. Figuur 7.8 geeft een tweede uitwerking weer.

Figuur 7.7
Overzicht van projecten
in uitvoering
(uitwerking 1)



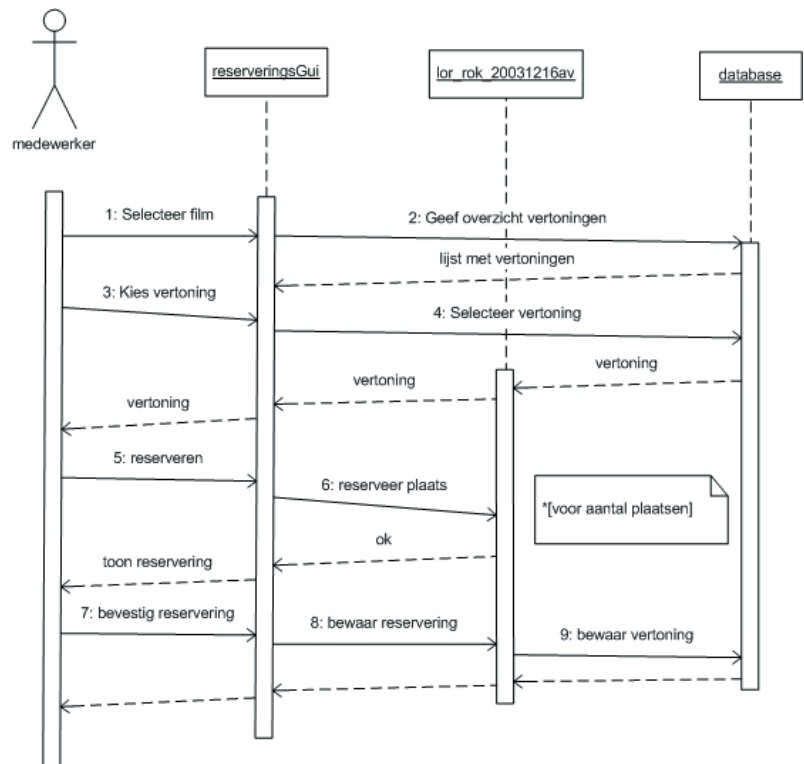
Figuur 7.8
Overzicht projecten in
uitvoering (uitwerking 2)



Een bioscoop heeft een aantal zalen waar films worden getoond. Tot een uur voor de aanvang van een filmvertoning kan er door een klant een film gereserveerd worden. Uiteraard kan dit alleen indien er nog plaatsen voor de filmvertoning beschikbaar zijn. Een scenario:

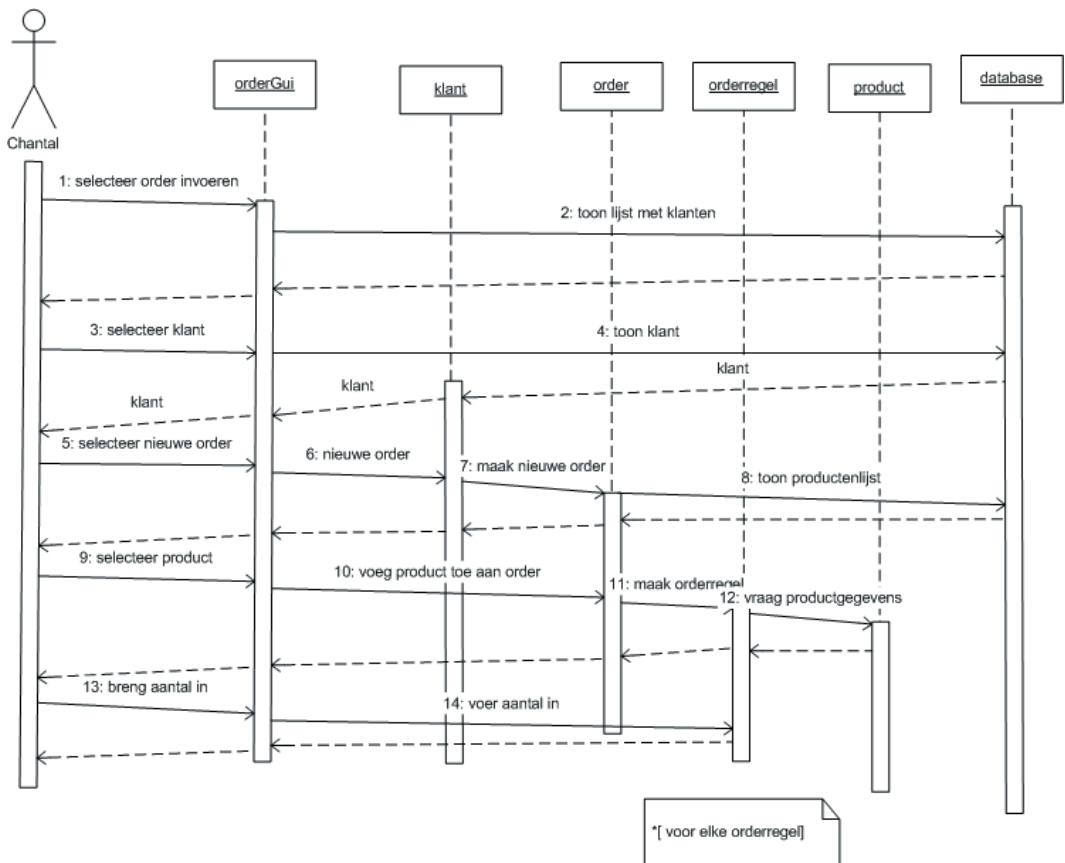
De heer Jansen belt op en vraagt of er nog plaatsen beschikbaar zijn voor de film Lord of the Rings, Return of the King. De vraag van de bioscoopmedewerker is welke vertoning de klant wil zien, dat wil zeggen de datum en het eventuele tijdstip. De klant geeft aan dat het om de donderdagavondvoorstelling gaat. De heer Jansen wil twee plaatsen reserveren. De medewerker geeft aan dat er nog plaatsen beschikbaar zijn op rij 3 en vraagt of de klant daarmee akkoord kan gaan. De heer Jansen bevestigt dit en vraagt hoe laat de gereserveerde kaarten moeten worden afgehaald. De medewerker reserveert de kaarten in het systeem en vertelt de heer Jansen dat de gereserveerde kaarten uiterlijk een half uur voor het begin van de voorstelling afgehaald moeten worden. De heer Jansen vraagt wat het verschuldigde bedrag is. Dit wordt door de medewerker doorgegeven, waarna het gesprek wordt beëindigd.

Figuur 7.9
Reservering van een
filmvertoning



7.8 Sequentiediagrammen bij PartyCups

Chantal: “Bij een klantgesprek moet ik de mogelijkheid hebben om een bestelling in te voeren. Ik selecteer de klant en vervolgens de gewenste producten. Bij elk geselecteerd product voer ik het gewenste aantal in. Nadat alle producten zijn ingevoerd, kan ik de order bevestigend afsluiten. Het systeem toont een bevestiging van de opslag van de zojuist geplaatste order.”



Figuur 7.10 Een deel van het sequentiediagram voor het inbrengen van een nieuwe order

Jan: “Voordat ik naar een fabrikant toe ga, wil ik eerst een overzicht van voorgaande gesprekken met de contactpersonen bij deze fabrikant. Ook wil ik een overzicht van de laatst gehanteerde inkoopprijzen.”

Marloes: “Nadat ik de afbeeldingen in de directory heb geplaatst waar deze altijd worden opgeslagen, moet ik de koppeling invoeren in de database. Voor elk product met een (nieuwe of gewijzigde) afbeelding, voeg ik de informatie toe aan de producttabel in de database.”

7.9 Samenvatting

Het gedrag van objecten wordt in UML weergegeven met het sequentiediagram. Een sequentiediagram laat de communicatie over en weer tussen verschillende objecten in de tijd zien. Een aantal voorbeelden daarvan zijn aan de orde gekomen. Van inloggen, een geldautomaat, een projectadministratie en het reserveren van een film zijn scenario's omgezet in sequentiediagrammen. Ook uit de case PartyCups is een sequentiediagram getoond. Opnieuw hebben we vastgesteld dat er bij het ontwerpen van de diagrammen technische eisen zijn. Als daaraan wordt voldaan en als we ingaan op het beschrijven van het gedrag van het te ontwerpen systeem, kunnen we bij het beoordelen van diagrammen alleen spreken van onvolledige of betere diagrammen. Beter betekent dan dat er meer van het ontwerp wordt gecommuniceerd.

Opgaven

1. Veel programmeeromgevingen kennen een compileerprogramma. De door de programmeur gemaakte (leesbare) code wordt door het compileerprogramma omgezet tot een uitvoerbaar programma, een executable. Noteer minimaal vijf redenen waardoor het compileerprogramma niet goed werkt.
2. Nadat een gebruiker een munt inwerpt in een snoepautomaat, verschijnt op het scherm van de automaat de waarde van de ingeworpen munt. Pas nadat het bedrag verschijnt, kan de volgende handeling uitgevoerd worden. Teken het sequentiediagram.

3. Een leraar wil inloggen op het netwerk. Na het invoeren van naam en wachtwoord op het invulscherf, zoekt de netwerkinterface contact met de database met inloggegevens. Blijkt de combinatie naam en wachtwoord te bestaan, dan geeft de database de besturing terug aan de netwerkinterface en de netwerkinterface geeft de toegang tot het netwerk vrij aan de leraar. Teken het sequentiediagram
4. Er zijn veel situaties waarbij berichten optreden die tijd kosten, bijvoorbeeld ook in een koffieautomaat zijn er berichten die tijd kosten. Onderzoek of bedenk bij welke handelingen berichten vertraging zullen oplopen.
5. In figuur 7.5 staat als 'guard'-conditie het aantal pogingen om in te loggen vermeld. Hoeveel pogingen kunnen ondernomen worden?
6. Bij het opruimen van de harde schijf besluiten we om alle bestanden die een extensie .mp3 hebben via de Windows Verkenner één voor één te wissen. Teken een sequentiediagram, voeg de herhaling toe en vermeld de 'guard'-conditie.
7. Vul het sequentiediagram uit figuur 7.6 aan, zodat het de gehele transactie laat zien.
8. Figuur 7.7 en 7.8 geven beide een overzicht van projecten in uitvoering. Kies het diagram dat volgens jou de beste weergave is van de door de manager gewenste rapportage. Motiveer je keuze.
9. Maak een sequentiediagram van het invoeren van een klant zoals beschreven in tabel 2.2 Toevoegen klant uit hoofdstuk 2.
10. Maak het sequentiediagram van het overzicht dat Jan wil hebben alvorens naar een fabrikant toe te gaan (zie paragraaf 7.8).

8

Toestandsdiagrammen – het veranderende gedrag van één object

8.1 Inleiding

In dit hoofdstuk gaan we kijken naar het gedrag dat een object in verschillende toestanden kan vertonen. Het gaat daarbij maar om één object. We geven van dat object de verschillende toestanden aan waarin het object zich kan bevinden. We geven ook aan hoe het object van de ene toestand in een andere toestand komt. Het hulpmiddel dat UML hiervoor aanbiedt is het toestandsdiagram.

8.2 Het doel van een toestandsdiagram

In een toestandsdiagram geven we aan in welke toestanden een object zich kan bevinden. Bovendien maken we in het diagram zichtbaar hoe een object van de ene toestand in de andere terecht kan komen. We kunnen toestandsdiagrammen ook gebruiken om de navigatie in een applicatie te laten zien. De verschillende schermen worden dan als toestand opgevat, terwijl de toetsaanslagen om naar een ander scherm te komen in het schema worden vastgelegd.

Ook eventuele blokkades of voorwaarden waaraan al of niet moet worden voldaan nemen we op in het toestandsdiagram. Hierdoor wordt vaak in een oogopslag duidelijk hoe een object reageert op andere objecten en uiteindelijk welke attributen en operaties een klasse nodig heeft om zo'n object te kunnen realiseren. We moeten ook hier rekening houden met alle mogelijke uitzonderingen.

Terzijde

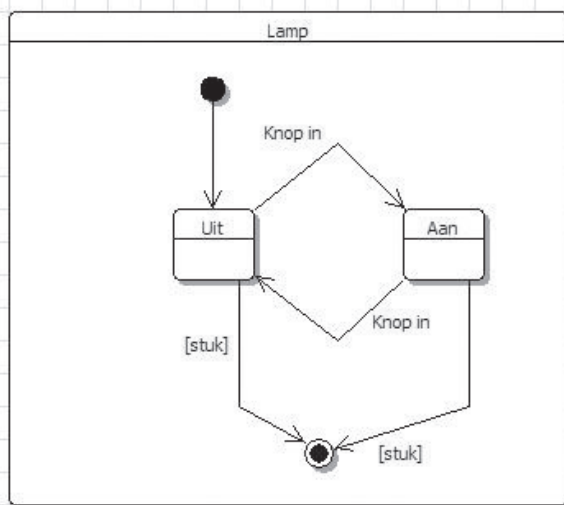
Stel dat je wilt vastleggen hoe een boek uitlening verloopt. In een activiteitendiagram heb je vastgelegd hoe dit als bedrijfsproces verloopt en welke actoren daarbij betrokken zijn. Nu wil je echter beschrijven wat het betekent voor een boek om te zijn uitgeleend. Het begint voor een bibliotheek bij het aanschaffen van een boek. Na de aanschaf moet het boek gereedgemaakt worden voor uitlening. Soms heeft een bibliotheek meerdere exemplaren van een boek. Elk exemplaar moet wel uniek zijn. Nadat het boek uitleenbaar is, kan het door een lezer worden geleend. Na een bepaalde uitleentermijn, zeg voor het gemak twee weken, komt het boek weer terug en kan het opnieuw worden uitgeleend. Na een paar jaar wordt het boek uit de collectie genomen en verkocht of vernietigd. Dan verdwijnt het boek uit het systeem, het object van dit boek bestaat dan niet langer. En dan hebben we het nog niet gehad over de mogelijkheid dat het boek tussentijds gerestaureerd kan worden en dus tijdelijk uit de uitlening kan worden gehaald.

8.3 De notatie van onderdelen van een toestandsdiagram

Een toestandsdiagram kent een aantal symbolen. We gaan deze bekijken aan de hand van het voorbeeld in figuur 8.1.

Figuur 8.1

Een voorbeeld van een toestandsdiagram



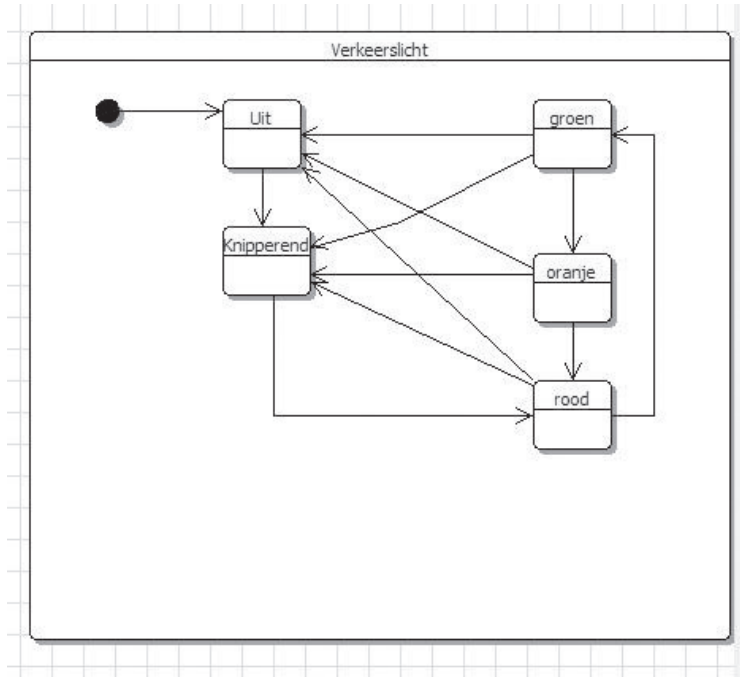
Aan de bovenzijde van de figuur treffen we een gesloten rondje aan. Met dit rondje geven we het beginsymbool aan voor het toestandsdiagram. Daarna wordt met behulp van een pijl aangegeven dat de beginsituatie van dit object de toestand ‘uit’ is. Door een knop in te drukken gaat het object van de toestand ‘uit’ naar de toestand ‘aan’. Door de knop weer in te drukken gaat het object van de toestand ‘aan’ naar de toestand ‘uit’. Als de lamp stuk gaat wordt het object verwijderd. Een object dat niet langer bestaat wordt aangegeven met een eindsymbool, een rondje met een cirkel er omheen. In dit geval laten we zien in welke toestanden een gewone lamp zich kan bevinden en hoe de overgangen tussen de verschillende toestanden worden gerealiseerd. Zo’n overgang noemen we ook wel een gebeurtenis, of in het Engels *event*. Bij de overgang naar het eindsymbool staat een voorwaarde tussen rechte haken. Deze voorwaarde, in dit geval de opmerking ‘stuk’, geeft aan wanneer naar de volgende toestand mag worden gegaan. Het is een voorwaarde of beperking, niet een gebeurtenis. Met deze onderdelen kunnen we toestandsdiagrammen maken.

8.4 Voorbeelden van toestandsdiagrammen

Om het gebruik van een toestandsdiagram nog wat te verhelderen volgen nu enkele voorbeelden. De lamp is natuurlijk als voorbeeld erg eenvoudig. Wat ingewikkelder wordt het als we een verkeerslicht nemen. Zo’n verkeerslicht kent verschillende toestanden. Het kan helemaal uit zijn. Het kan met het middelste oranje licht knipperen. Het kan een van de drie kleuren aangeven groen, oranje en rood. Ook de overgangen zijn niet meteen duidelijk. In Nederland begint een stoplicht meestal met helemaal uit. Daarna kan het in een knipper toestand terecht komen of rood. Van rood gaat het over in groen en van groen gaat het over in oranje. Van oranje gaat het dan weer naar rood enzovoort. Als we al deze toestanden en overgangen afbeelden in een toestandsdiagram krijgen we figuur 8.2.

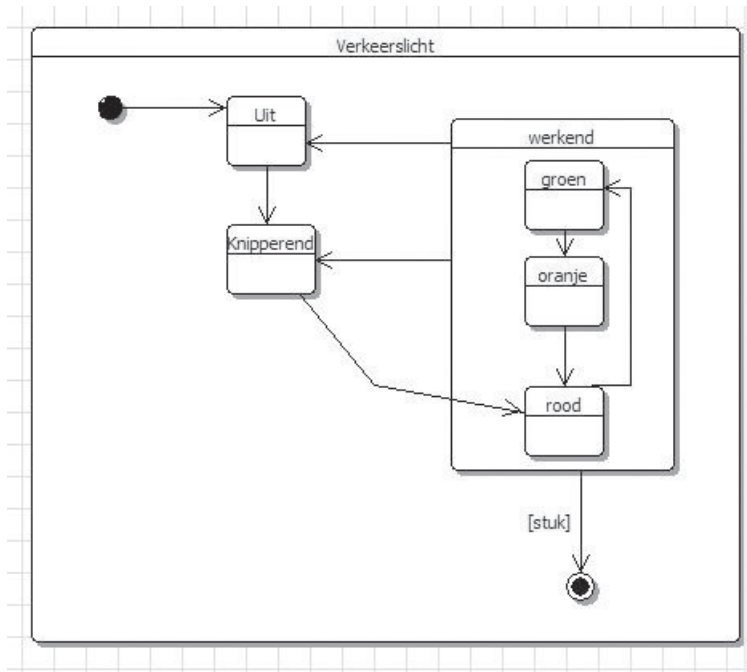
Het toestandsdiagram is zowel onduidelijk als onvolledig. Het wordt onduidelijk doordat vanuit de toestanden groen, oranje en rood altijd terug gevallen kan worden op de toestand knipperend of uit. Daardoor gaan de overgangen kruisen en wordt het onoverzichtelijk. Ook is onduidelijk wat er gebeurt als het stoplicht defect is. De eindtoestand is niet meegenomen in het diagram.

Figuur 8.2
Toestandsdiagram van
een verkeerslicht
(onvolledig)



Bij het verminderen van alle overgangen zoeken we eigenlijk naar de mogelijkheid om een algemenere toestand aan te geven. In het geval van een verkeerslicht is die aanwezig. Want als het verkeerslicht in werking is, is het in een van de toestanden groen, oranje of rood. Uit en knipperend zijn eigenlijk bijzondere toestanden. Door een supertoestand ‘werkend’ in te voeren kunnen we het aantal overgangen drastisch terugbrengen.

Figuur 8.3
Toestandsdiagram van
een verkeerslicht
(verbeterd)

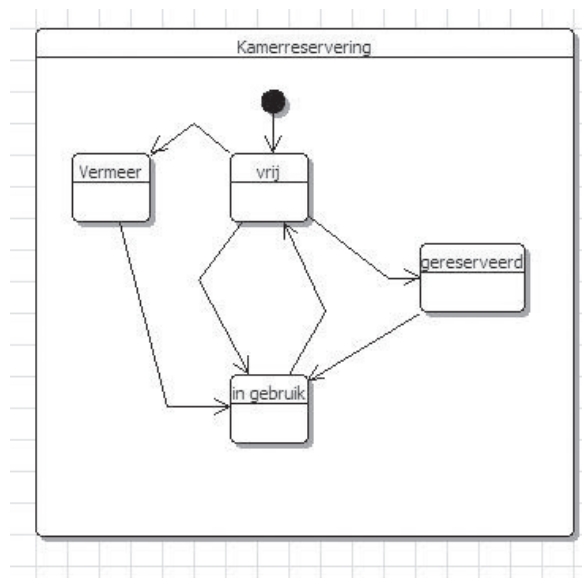


Als we nu het diagram bestuderen dan kunnen we vaststellen dat vanuit een werkende situatie van het verkeerslicht (groen, oranje, rood) het verkeerslicht in de toestanden uit en knipperend kan komen. Of als het stuk is in de eindtoestand. Volgens hetzelfde toestandsdiagram kan een verkeerslicht niet stuk gaan als het uit of knipperend is. Of zoiets klopt moeten we weer vragen aan een domeinsdeskundige. Iemand die alles weet van verkeerslichten. Zodra we namelijk naar Duitsland zouden afreizen moet het diagram worden aangepast. Daar is loopt de toestandsverandering altijd via oranje. Dus ook na rood komt eerst oranje.

In de opgave over hotel Amsterdam wordt het reserveringsysteem van de kamers beschreven. Een kamer kan gereserveerd zijn door de reisorganisatie Vermeer, een kamer kan vrij beschikbaar zijn, gereserveerd door een klant of in gebruik zijn bij een klant. De overgang van de kamerverhuur is niet altijd even duidelijk. Wel kan de beginsituatie worden vastgesteld, dat is immers de situatie dat de kamer vrij is. Daarna kan de kamer worden geclaimd door de reisorganisatie of gereserveerd worden door een andere klant. Als de reisorganisatie een claim op de kamer heeft gelegd kan deze via Vermeer worden verhuurd. De status verandert dan

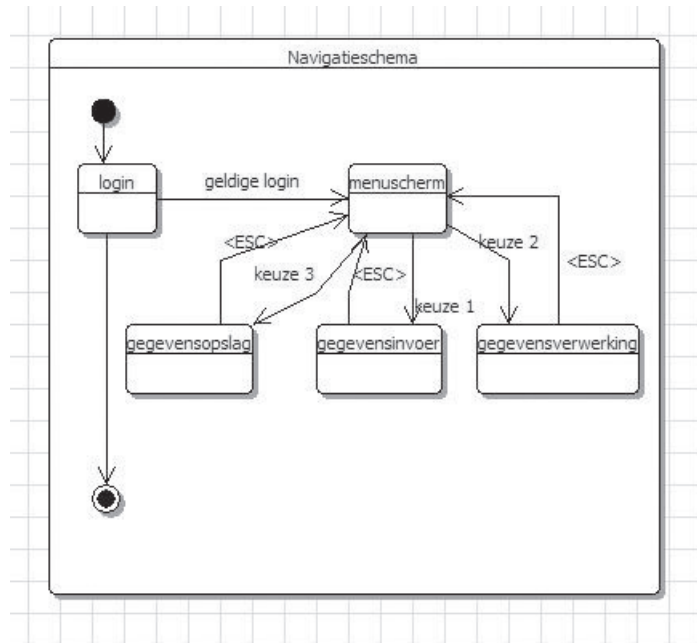
in gereserveerd. Vanuit deze toestand kan de kamer weer vrijkomen als de reservering wordt opgeheven. Maar de kamer kan ook daadwerkelijk gebruikt worden door de klant. Na gebruik wordt de kamer gereinigd en daarna weer vrij gegeven voor een reservering. Een kamer hoeft natuurlijk niet gereserveerd te zijn maar kan ook rechtstreeks worden verhuurd. Het toestandsdiagram dat deze situatie weergeeft is figuur 8.4.

Figuur 8.4
Kamerreservering van
Hotel Amsterdam



Een navigatieschema voor een internetapplicatie kan ook met behulp van een toestandsdiagram worden afgebeeld. Daarbij zijn de toestanden de schermen en de overgangen de toetsaanslagen om van het ene scherm naar het andere te komen. De internetapplicatie toont aan de start het loginscherm. Bij een geldige login komt men in een menuscherm. Vanuit het menuscherm komt men in een van de drie opties, gegevensinvoer, gegevensverwerking en gegevensopslag. Vanuit deze opties kan men met een druk op de Esc-toets weer terugkeren in het menuscherm. Vanuit het menuscherm kun je ook uitloggen en de applicatie verwijderen. Het navigatieschema is afgebeeld in figuur 8.5.

Figuur 8.5
 Navigatieschema van
 een internetbrowser



8.5 Samenvatting

Het gedrag van een object wordt in UML weergegeven in een toestandsdiagram. Een toestandsdiagram laat de toestanden zien waarin een object zich kan bevinden. Een aantal voorbeelden daarvan zijn aan de orde gekomen (lamp, verkeerslicht, reserveringssysteem, navigatieschema). Opnieuw hebben we vastgesteld dat er bij het ontwerpen van de diagrammen technische eisen zijn. Als daaraan wordt voldaan en als we dieper ingaan op het beschrijven van de toestanden waarin een object zich kan bevinden, dan kunnen we bij het beoordelen van de diagrammen alleen spreken van onvolledige of betere diagrammen. Beter betekent ook hier dat er meer van het ontwerp wordt gecommuniceerd.

Opgaven

1. Een leraar wil inloggen op het netwerk. Na het invoeren van naam en wachtwoord op een invulscherf klikt deze leraar op de loginknop. Daarna wordt in de applicatie bepaald of toegang kan worden verleend of moet worden geweigerd. Als de toegang wordt geweigerd wordt een melding getoond. Na het indrukken van een willekeurige toets wordt opnieuw het loginscherf getoond. Als toegang wordt verleend toont de applicatie een menuscherf. Maak het navigatieschema.
2. Er mogen maar drie pogingen worden ondernomen om in te loggen. Als het drie keer fout gaat moet de applicatie de internetbrowser afsluiten. Pas het navigatieschema aan.
3. Een geldautomaat staat de hele dag te wachten op een klant. De klant kan zijn pas invoeren, waarna de automaat zal vragen om de pincode die bij de kaart en rekening van de gebruiker hoort. Na validering van de pincode krijgt de gebruiker toegang tot het systeem om geld op te nemen of saldo te tonen. Als de gebruiker kiest voor geld opnemen, wordt een aantal opties getoond met verschillende bedragen. Na de keuze van een bedrag volgt het uitwerpen van de pas en daarna het uitwerpen van het gekozen bedrag. Ten slotte keert het apparaat terug in de wachttoestand. Maak het toestandsdiagram van de geldautomaat.
4. Een bioscoop heeft een reserveringssysteem ingesteld. Tot een uur voor de aanvang van een filmvertoning kan een film gereserveerd worden. Uiterlijk een half uur van te voren moeten deze worden opgehaald. Als ze worden opgehaald wordt de status van deze stoelen als bezet aangegeven. Na afloop van de film worden alle stoelen weer vrijgegeven. Maak het toestandsdiagram van een stoel object.
5. In hoofdstuk 3 wordt het order proces van PartyCups beschreven. Geef in een toestandsdiagram de verschillende toestanden van een order aan.

6. In hoofdstuk 3 wordt in opgave 4 een zwemwedstrijd beschreven. Maak een toestandsdiagram van de verschillende toestanden van een deelnemer.
7. In hoofdstuk 3 wordt in opgave 5 ook over een bioscoopreserve-ring gesproken. Dat wijkt af van het in dit hoofdstuk in opgave 4 beschreven systeem. Pas het diagram zo aan dat het van toepassing is op de opgave van hoofdstuk 3.
8. Maak een toestandsdiagram van een cursus zoals beschreven in hoofdstuk 3 opgave 8 bij het Nationaal Instituut voor Information Engineers.
9. Maak een toestandsdiagram van een product zoals beschreven in hoofdstuk 3 opgave 10, E-Lectora.
10. Medewerkers van een organisatie kunnen op hun eigen werkplek artikelen bestellen via een website. De applicatie is daarvoor 7 dagen per week 24 uur bereikbaar en werkt met de meest gebruikte browsers. Om artikelen te bestellen bij een magazijnafgifte punt is wel goedkeuring nodig van de directe chef. Een aanvraag die op de website wordt gedaan gaat daarom eerst naar de chef van de medewerker ter goedkeuring. Na goedkeuring wordt de aanvraag in bewerking genomen. Bestellingen worden zoveel als mogelijk in een keer geleverd. Een keer per maand worden de afgeleverde bestellingen intern gefactureerd, echter alleen als de bestelling volledig is uitgeleverd. Een bestelling die gefactureerd is en waar geen klachten over zijn gemeld wordt daarna gearcheveerd. Een klacht moet worden opgelost, daarna kan de bestelling alsnog worden gearcheveerd. Maak het toestandsdiagram van een bestelling.

Index

A

9-9-9 regel 2
 activiteitendiagram 37
 actor 18
 in use-case digram 21
 aggregatie 75
 API 93
 application programming interface 93
 associatie 73
 gekwalficeerde 75
 asynchroon bericht 105
 attributen
 van klasse 62

B

bericht
 asynchroon 105
 synchroon 105
 tijdgebonden 107
 bouwfase *zie* construction phase
 business constraints 22

C

compositie 75
 constraint
 bij tijdgebonden bericht 107
 construction phase 92
 CRM-systeem 16
 Customer Relation Management System
 zie CRM-systeem

D

declareren 67
 deskresearch 17
 DSDM 85
 Dynamic Systems Development Method
 85

E

elaboration phase 90
 embedded system 107
 Entity Relation Diagram *zie* ERD
 ERD 77
 extreme programming 11, 85

F

FDD 85
 Feature Driven Development 85
 flow 38
 fonteinmodel 10
 functioneel ontwerp 61
 functionele eis
 zie functionele requirements
 functionele kenmerken 74
 functionele requirements
 aanpassen bestaande systemen 17
 nieuwe systemen 19

G

gegevenstype 67
 gekwalficeerde associatie 75
 generalisatie 76
 guard-conditie 107

I

inception phase 87
 incrementeel en iteratief 10
 incrementeel model 7
 inheritance 63
 iteratie 107
 iteratief en incrementeel 10

K

kandidaat-klassen 63
 klantrelatiebeheersysteem
 zie CRM-systeem

klasse 62
 bepalen van 63
klassendiagram 62, 87
 van database 76

L

levenscyclusmodellen 10
lineaire methode 1
logisch model 87

M

multipliciteiten 74

N

navigatie 115
non-functional requirements 22

O

object 68
objectdiagram 68, 87
objectgeoriënteerde methode 10
objectgeoriënteerd ontwerp 62
observatie
 bij systeemanalyse 19
onderhoudskosten 5
ontwikkelmethoden
 agile 85
operaties
 bij klassen 62
opleverfase *zie* transition fase
orderadministratie 46
orderproces 46
overerving 63

P

procesmodel 87
proof of concept 89
pseudo-requirements 23

Q

queries 76

R

RAD 9

Rapid Application Development *zie* RAD
rapid prototyping model 6

S

scope 16
Scrum 85
SDM 6
sequentiediagram 103
 notatie 104
 voorbeelden 108
shareholders 88
spiraalmodel 9
stakeholders 87
startfase *zie* inception phase
subklasse 63
superklasse 63
swimlane 38
synchronisatiebalk 39
synchroniseer-en-stabiliseermodel 12
synchronous message 105
synchroon bericht 105
System Development Method *zie* SDM

T

tijdgebonden bericht 107
transition phase 93

U

uitwerkingsfase *zie* elaboration phase
UML 20
UML-diagrammen 86
Unified Modeling Language *zie* UML
Unified Process 86
UP 86
use-case 21
use-case diagram 20, 21
use-case diagram en user story 25
use-case template 26
user stories 23
user story en use-case diagram 25

W

watervalmethode 2