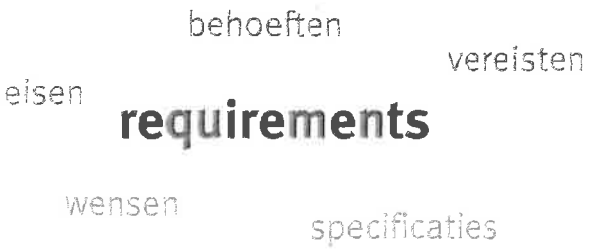
5 Functioneel ontwerp

Bij de ontwikkeling van applicaties wordt er voor het ontwerp bijna altijd gebruik gemaakt van een functioneel en een technisch ontwerp.
Het functioneel ontwerp beschrijft wat de te bouwen applicatie of website aan functionaliteit moet bieden. Het moet zo omschreven zijn dat een klant het document goed kan begrijpen. Daarnaast moet de informatie ook nuttig zijn voor iemand die de site moet gaan bouwen. Het komt erop neer dat het functioneel ontwerp niet te technisch mag zijn. Het functioneel ontwerp is de blauwdruk van de applicatie/website.
In het technisch ontwerp wordt bepaald hoe de software gemaakt gaat worden. Welke technieken worden gekozen? Hoe en waar wordt de data opgeslagen? Hoe worden gegevens verwerkt? Het technisch ontwerp is niet bedoeld voor de klant en hoeft dan ook niet 'eenvoudig' te zijn in taalgebruik. In het technisch ontwerp staat vaktaal.
Een belangrijk onderdeel op basis waarvan een functioneel ontwerp wordt gemaakt zijn de requirements. Het begrip requirements is een van oorsprong Engels begrip en een goed Nederlands begrip bestaat er niet voor, daarom blijven wij ook gewoon van 'requirements' spreken. De betekenis ligt ergens in het midden van behoeften, eisen, specificaties, vereisten of wensen. Een belangrijk deel van de requirements heb je al vastgelegd met behulp van de MoSCoW-methode bij het programma van eisen.

5.1 Requirements

Wil je het functioneel ontwerp goed kunnen opzetten, dan moet je een scherp zicht hebben op de requirements. De requirements beschrijven welke elementen en functies benodigd zijn voor een specifiek project.



We onderscheiden verschillende soorten requirements.

Business requirements
Business requirements zijn de verbeteringen die de onderneming wil realiseren in een nieuw of bestaand proces, bijvoorbeeld:
- groter marktaandeel
- hogere efficiëntie
- hogere omzet
- hogere tevredenheid van klanten
- lagere kosten
- meer winst
- kortere doorlooptijd
- minder risico's
- kortere inwerktijd

Business requirements zijn belangrijk omdat hiermee het te maken systeem wordt afgebakend. Er wordt een gemeenschappelijk beeld gedefinieerd van wat het systeem moet kunnen.

User requirements
Dit zijn de activiteiten of processen die de gebruikers met het ICT-systeem gaan uitvoeren. Om deze in beeld te brengen wordt vaak de techniek van use cases gebruikt. User requirements heten ook wel proces-requirements. Ze moeten niet te globaal, maar ook niet te gedetailleerd zijn.

Functionele requirements
Functionele requirements beschrijven de functies die het systeem moet vervullen. Functionele requirements werken we vaak uit in een use-case-diagram. Hiermee kunnen we grafisch weergeven wat het systeem moet gaan doen en wie er van welke functie gebruik gaat maken.

Niet-functionele requirements
Niet-functionele requirements geven criteria aan om het functioneren van het systeem te beoordelen, maar beschrijven niet het specifieke gedrag zelf. Het gaat meestal over betrouwbaarheid, onderhoud, performance en veiligheid.

5.2 Unified Modeling Language (UML)

Unified Modeling Language (UML) is een modelmatige taal om objectgeoriënteerde analyses en ontwerpen voor een informatiesysteem te kunnen maken. UML is bedoeld om een beschrijving van het systeem te kunnen maken op basis waarvan vervolgens het systeem kan worden gebouwd.
UML is geen ontwikkelmethode! UML vertelt niet hoe je een systeem moet ontwerpen. Het vertelt ook niet wat je eerst moet doen en wat daarna. Wat UML

wel doet is je helpen het systeem te visualiseren en er over te communiceren met anderen. Het is een visuele modelleertaal die begrippen en diagrammen standaardiseert.
UML is ontworpen door Grady Booch, James Rumbaugh en Ivar Jacobson in de jaren negentig en is sinds 1997 een standaard. UML staat onder toezicht van de Object Management Group (OMG) en is de industriestandaard voor het grafisch weergeven van software.
Als je een systeem beschrijft, dan zijn er onderdelen van het systeem die processen weergeven, wat in UML wordt beschreven in dynamische diagrammen. Andere onderdelen duiden zaken en niet de processen. Deze onderdelen worden statische diagrammen genoemd.

Structuur- en gedragsdiagrammen

UML biedt de volgende verzameling van structuur- en gedragsdiagrammen:

- Statische diagrammen
  - klassendiagram (class diagram)
  - objectendiagram (object diagram)
  - componentendiagram (component diagram)
  - gebruiksdiagram (deployment diagram)
- Dynamische diagrammen
  - use-case-diagram
  - collaboratiediagram (collaboration diagram/communicatiediagram)
  - volgordediagram/sequentiediagram (sequence diagram)
  - activiteitendiagram (activity diagram)
  - correlatiediagram
  - toestanddiagram (state diagram)

UML bestaat uit symbolen waaruit de modellen worden samengesteld. Daarnaast is er een verzameling van regels voor het gebruik van deze symbolen. Deze regels geven aan hoe de symbolen eruit zien en hoe ze gecombineerd worden. Ook geven de regels met betrekking tot de betekenis van elk symbool afzonderlijk aan hoe deze in context staat tot andere symbolen. Een modelelement dat in verschillende diagrammen wordt gebruikt heeft altijd dezelfde betekenis in elk diagram.
Niet alle diagrammen worden altijd gebruikt, dat is aan de ontwikkelaar. Diagrammen die vaak worden gebruikt zijn:

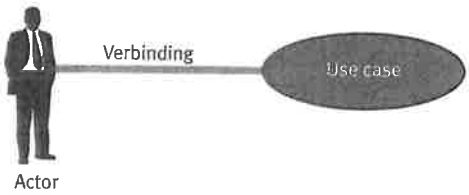
- Statische diagrammen
  - klassendiagram (class diagram)
- Dynamische diagrammen
  - use-case-diagram
  - volgordediagram/sequentiediagram (sequence diagram)
  - activiteitendiagram (activity diagram)

Deze diagrammen komen in de volgende hoofdstukken afzonderlijk aan de orde.

5.3 Use-case-diagram

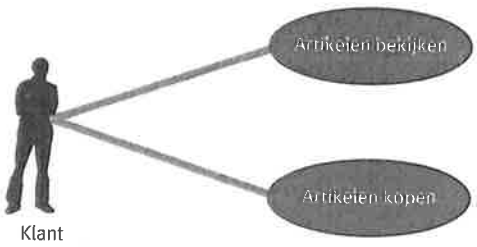
In het rapport Functioneel ontwerp zorgen we ervoor dat de activiteiten of processen die de gebruikers met het ICT-systeem gaan uitvoeren duidelijk zijn omschreven. Wat is er mooier dan een volledig overzicht op één A4-tje? Dit kunnen we bereiken met behulp van een use-case-diagram, op zich een eenvoudige techniek die eenvoudig is aan te leren. Hiervoor zijn heel veel verschillende tools beschikbaar, zowel gratis als commercieel. Bekende ontwikkelomgevingen zijn Visio, ArgoUML en Violeteditor. Dit zijn drie totaal verschillende voorbeelden uit een veel groter assortiment. Er bestaat niet een beste of slechtste. Wanneer jij vindt dat een tool doet wat jij nodig hebt, dan is dat een goede tool voor jou.

Een use-case-diagram is opgebouwd uit een paar standaardelementen, te weten:
Actor: een gebruiker (en heel soms een iets) die een actie in gang zet.
Use case: de actie die onderdeel vormt van het systeem.
Verbinding: de relatie tussen de gebruiker en de actie.

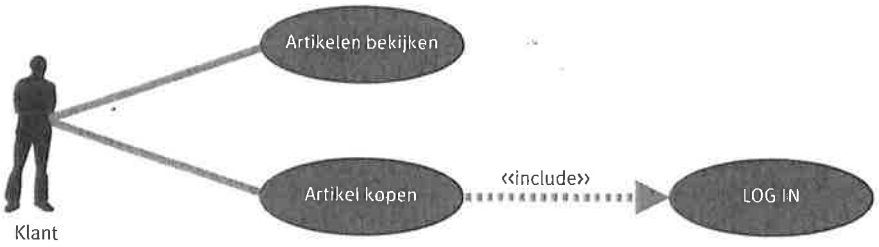


Voorbeeld van een webshop

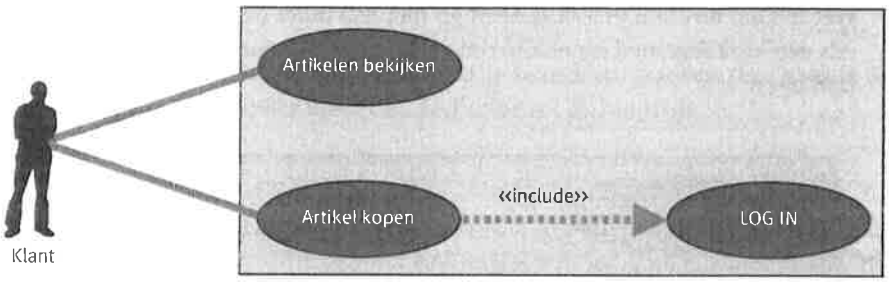
Je hebt vast wel eens iets bij een webshop gekocht. Het use-case-diagram van de webwinkel zou er zo uit kunnen zien:



Onze klant kan in de webshop twee verschillende dingen. Hij kan namelijk artikelen bekijken en artikelen kopen.
Wil onze klant echter een artikel kunnen kopen, dan moeten zijn gegevens bij ons bekend zijn. Daarom gaan we het use-case-diagram aanpassen. We eisen dat de klant inlogt. Dat laten we in onze use-case-diagram zien door middel van een 'include'-pijltje. Als je wilt kopen *moet* je inloggen.

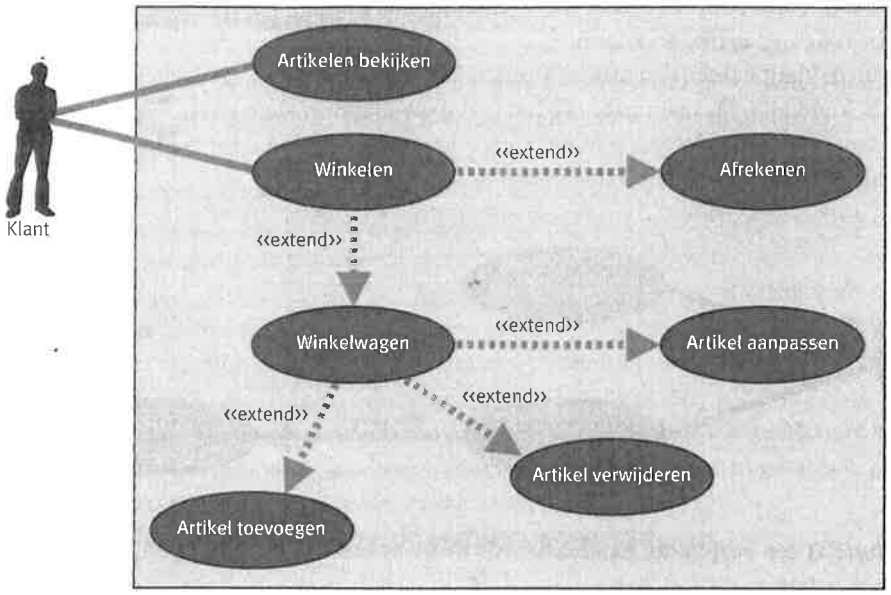


'Include' is een verplichte handeling. Als de klant een artikel wil kopen, dan *moet* hij ook inloggen.

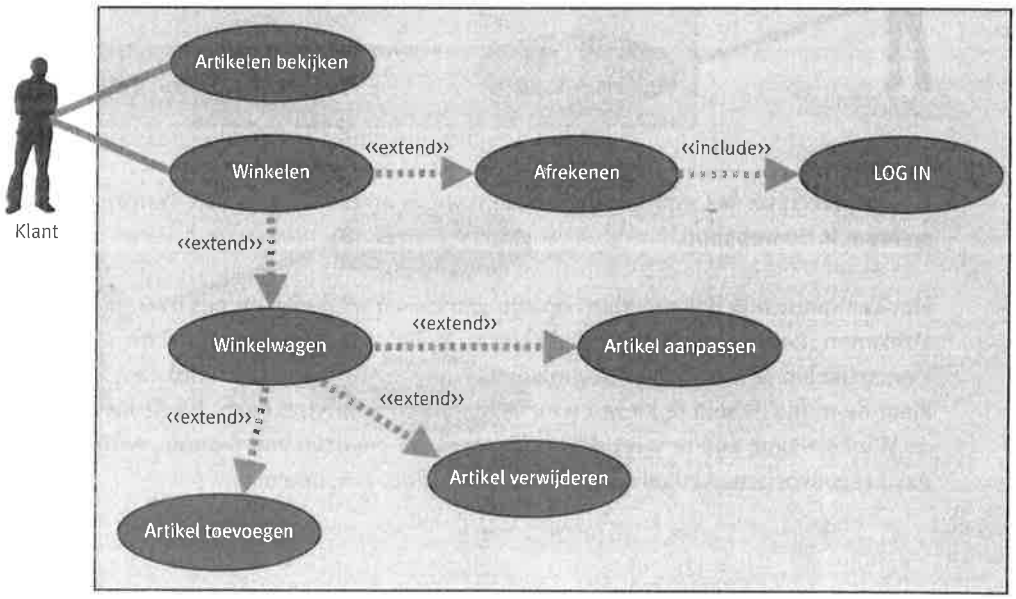


Vaak kaderen we het systeem in en zetten we de actoren buiten het systeem. Het systeem is de webshop.

Het kan natuurlijk dat een klant op zijn gemak wil winkelen en pas over gaat tot afrekenen (kopen) als hij helemaal klaar is. In het voorbeeld kan de klant kiezen alleen artikelen te bekijken of te winkelen. Wanneer hij kiest voor winkelen, krijgt de klant de mogelijkheid te kiezen voor Winkelwagen of Afrekenen. Bij de keus voor de Winkelwagen zijn er weer drie opties waaruit gekozen kan worden, namelijk: Artikel toevoegen, Artikel verwijderen en Artikel aanpassen.

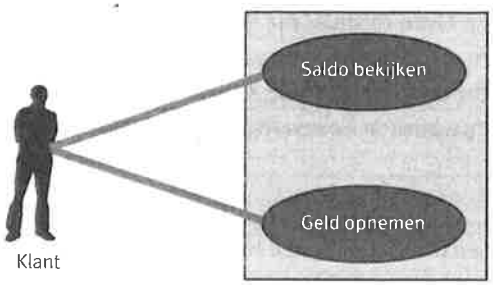


Het feit dat de klant een *keus* heeft en niet iets moet geven we aan met een *'extend'*. Als een kant ingelogd moet zijn om te kunnen afrekenen, dan moeten we dat nog opnemen.



5.4 De use case

Om alles compleet te maken zullen we de use case verder moeten uitwerken. Use-case-diagrammen tonen actoren, dat zijn mensen of andere onderdelen van het systeem. De use cases zijn de scenario's wanneer zij het systeem gebruiken en hun relaties. In de use case beschrijven we het scenario zo volledig mogelijk (user stories).
Als voorbeeld gebruiken we een geldautomaat. Dit omdat iedereen weet hoe zo'n apparaat werkt en herkent waar het over gaat.



Actor
Na de naam en de versie geven we aan wie de betrokken actoren zijn. Vaak is er sprake van één actor, maar meer dan één actor is ook mogelijk.

Precondities
Vervolgens geven we aan wat de precondities zijn. Precondities zijn voorwaarde waaraan voldaan moet zijn. Je kunt bedenken dat je een bankpas moet hebben als je geld uit een geldautomaat wilt halen. Hier kan het ook om meer dan één voorwaarde gaan.

Beschrijving
Bij de beschrijving omschrijf je zo volledig en beeldend mogelijk wat er gebeurt. Bij de betaalautomaat zou dat kunnen zijn:

- voer pas in in automaat
- voer pincode in
- voer bedrag in
- bevestig bedrag
- geef antwoord op wel/geen kwitantie
- neem pas uit
- neem geld uit
- (neem bon uit)

Uitzondering

Vervolgens beschrijven we de uitzonderingen. Voor de geldautomaat zou dat kunnen zijn:

- foute pincode
- onvoldoende saldo
- onvoldoende geld in kas
- geen papier in de printer

In elk van de bovenstaande situaties zal er een ander scenario gaan spelen.

Niet-functionele eisen

Bij de niet-functionele eisen komen eisen te staan die niet direct de functionaliteit betreffen maar zaken daaromheen. Als de geldautomaat er 10 minuten over doet om één klant af te handelen, dan functioneert hij nog steeds. Een niet-functionele eis zou kunnen zijn dat een tansactie binnen 40 seconde afgerond kan zijn.

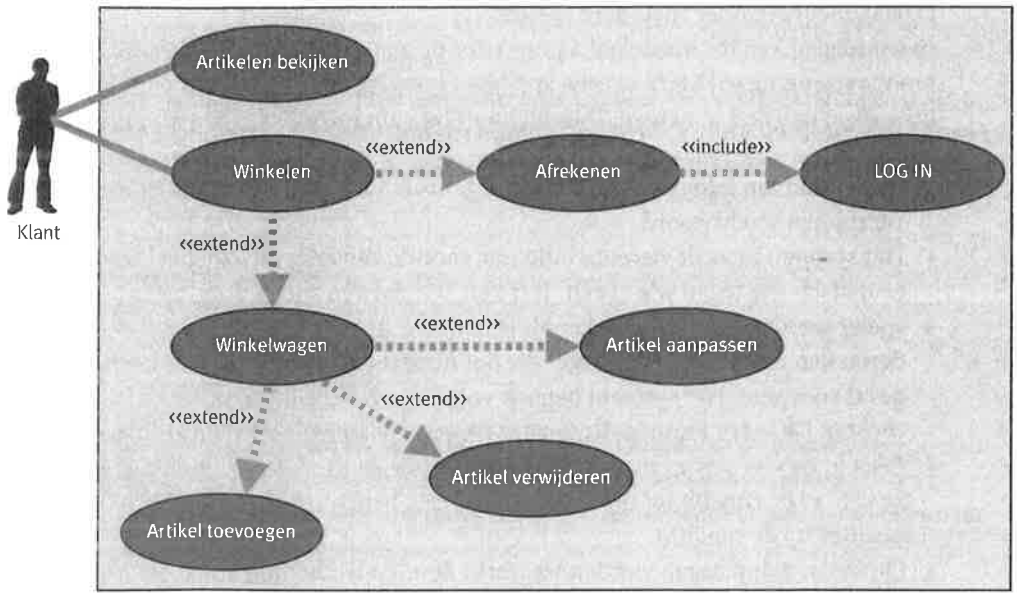
Postconditie

Bij de postconditie is de situatie beschreven nadat de handeling zich heeft voltrokken. In het voorbeeld van de geldautomaat: de klant heeft zijn geld, de automaat wacht op invoer van een nieuwe pas.
Alles bij elkaar gaat het er als volgt uitzien.

Naam	Geldautomaat
Versie	1.0
Actor	Klant
Preconditie	Klant is in bezit van bankrekening en bankpas
Beschrijving	1 Klant neemt geld op a Voer pas in in automaat b Voer pincode in c Voer bedrag in d Bevestig bedrag e Geef antwoord op wel/geen kwitantie f Neem pas uit g Neem geld uit h Neem bon uit
Uitzonderingen	2 Klant heeft onvoldoende saldo a Voer pas in in automaat b Voer pincode in c Voer bedrag in d Bevestig bedrag e Melding onvoldoende saldo f Ander bedrag g Geef ja-nee (ja scenario 1) h Nee: neem pas uit

Naam	Geldautomaat
Uitzonderingen	3 Geen papier in printer a Voer pas in in automaat b Voer pincode in c Voer bedrag in d Bevestig bedrag e Melding geen kwitantie mogelijk f Neem pas uit g Neem geld uit 4 Onvoldoende geld in automaat a Voer pas in in automaat b Voer pincode in c Voer bedrag in d Melding, onvoldoende geld in kas, kies ander bedrag e Scenario 1
Niet-functionele eisen	Een tansactie moet binnen 40 seconde afgerond kunnen zijn
Postconditie	Klant heeft zijn geld, automaat wacht op invoer van een nieuwe pas

5.5 Opgaven



1 Maak de use case voor de actie Artikel toevoegen uit het use-case-diagram Webshop.
2 Maak de use case voor de actie Artikel verwijderen uit het use-case-diagram Webshop.

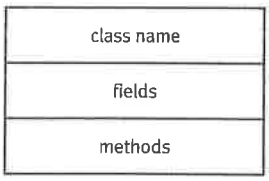
3 Maak de use case voor de actie Artikel aanpassen uit het use-case-diagram Webshop.
4 Maak de use case voor de actie Afrekenen uit het use-case-diagram Webshop.
5 Wat zijn requirements?
6 Noem een aantal voorbeelden van business requirements.
7 Wat zijn functionele requirements?
8 Maak een use-case-diagram voor een snoepautomaat. De snoepautomaat bevat verschillende soorten snoep. Een klant heeft een pas waarmee de automaat in werking kan worden gesteld. Op deze pas staat een tegoed. De automaat zal als afronding van de tansactie het tegoed aanpassen. Het is mogelijk om in de automaat het tegoed op de pas te verhogen. Hiervoor zal ook een bankpas aan de automaat moeten worden aangeboden.
9 Maak de use case voor het ophogen van het saldo op de snoepautomaatpas.
10 Maak de use case voor het kopen van snoep.
11 Maak een use-case-diagram voor tennisvereniging TopSpin.
De leden van de tennisvereniging TopSpin tennissen wekelijks op woensdagavond. Zij houden een ranglijst bij. De ranglijst is eenvoudig. Je mag altijd iemand uitdagen die maximaal vijf posities hoger op de ranglijst staat. Als je wint van iemand hoger in de ranglijst dan wissel je van positie op de ranglijst. Per avond worden er door een lid maximaal twee wedstrijden gespeeld.
Een speler mag maximaal twee spelers die hoger in de ranglijst staan uitdagen, waarbij er een voorkeur moet worden opgeven. Natuurlijk kan een speler ook uitgedaagd worden door meerdere spelers.
Het uitdagen kan tot maximaal 24 uur voor de aanvang van de speelavond. De tennisvereniging wil hiervoor een app laten ontwikkelen die de clubleden kunnen gebruiken om deel te nemen. Daarbij moet rekening worden gehouden met het volgende.
- Om te kunnen inloggen maakt een lid gebruik van zijn lidnummer en een zelf verzonnen wachtwoord.
- Het systeem bepaalt vierentwintig uur voor de aanvang van de speelavond automatisch wie tegen wie speelt. Het kan zijn dat een speler door meer dan één speler wordt uitgedaagd. Zeker als je bedenkt dat de uitgedaagde zelf ook anderen kan uitdagen. De uitdager die het hoogst in de ranglijst staat heeft in dat geval voorrang. Het systeem bepaalt volgens een bepaald algoritme een goed rooster. De leden kunnen dit rooster vervolgens in het systeem raadplegen.
- Eens in de vier weken mag je je afmelden voor de speelavond. Indien je je een keer te vaak afmeldt (of niet op tijd aanwezig bent), zak je automatisch twee posities op de ranglijst.
- De wedstrijduitslagen worden verwerkt door de wedstrijdleiding. De wedstrijdleiding zorgt ook voor het beheer van de leden. Het is ook de wedstrijdleiding die vastlegt dat iemand te laat is.
- Het systeem zorgt ervoor dat na het invoeren van een wedstrijduitslag de ranglijst automatisch wordt bijgewerkt. De clubleden kunnen de ranglijst met behulp van het systeem inzien.

5.6 Klasse en klassendiagram

Willen we een applicatie bouwen die succesvol is en doet wat zij moet doen, dan zullen we orde moeten aanbrengen in de zaken die we gebruiken. Die orde kunnen we aanbrengen met behulp van een klassendiagram (class diagram).
Orde aanbrengen is iets wat we van nature ook doen. Trek je een besteklade open dan is de kans groot dat er orde is aangebracht. De dingen in de la hebben een naam en in de meeste gevallen ook een eigen plaats. Maar ook buiten de la hebben de dingen om ons heen een naam. In de meeste gevallen hoort een ding tot een groep en als het even kan een subgroep.
In de besteklade is het niet ondenkbaar dat we een driedeling hebben gemaakt.

Bestek —> Lepel, Vork, Mes

Op één of andere manier weten we wat bestek is en weten we het verschil tussen een mes, een lepel en een vork. Dat komt omdat we voor elk object een specificatie hebben, een omschrijving die specifieke kenmerken van het object beschrijft. Die specificatie hebben we vaak ongemerkt aangenomen. Maar wanneer we een applicatie gaan bouwen kunnen we niet volstaan met 'ongemerkt aannemen'. We gaan objecten classificeren en hun onderlinge relatie in beeld brengen. We gaan een klassendiagram maken.
Als voorbeeld nog even de besteklade. In de besteklade liggen dingen (objecten) die we bestek noemen. Die objecten kunnen we als groep omschrijven. Zo'n omschrijving noemen vanaf nu een klasse.
Klassendiagrammen worden gebruikt in de Unified Modeling Language (UML), welke principes we vaak toepassen binnen de software-ontwikkeling. Een klasse wordt getekend als een rechthoek, die is opgebouwd uit weer drie rechthoeken.



- De bovenste rechthoek bevat de 'class name' (naam van de klasse).
- Het middelste rechthoek bevat de 'fields' (attributen van de klasse).
- Het onderste deel bevat de 'methods' (functies of handelingen die men kan uitvoeren op deze objecten).

Name

De naam van de klasse is belangrijk omdat hiermee wordt bepaald over welk object je het hebt. Aan de naam worden de volgende eisen gesteld:

- hij is uniek in het diagram
- hij is altijd in enkelvoud

Fields

De attributen (fields) beschrijven in feite het object. Zo zal een object Auto bestaan uit de attributen Motor, Wielen, Carrosserie, Stuur en nog een aantal attributen.
Attributen kunnen de volgende specificatie hebben:

- schrijfwijze: zichtbaarheid naam: type = startwaarde
- (private aangeraden voor attributen)
- zichtbaarheden
- +: publiek
- -: privaat
- #: protected

Methods

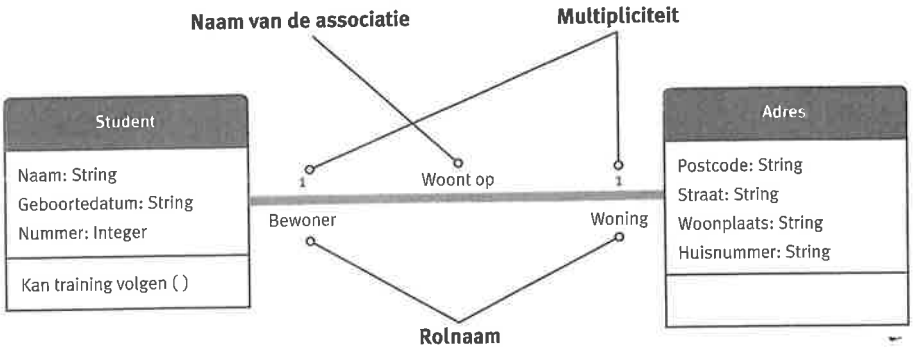
Het onderste deel bevat de methoden, methods.
Methoden zijn functies of handelingen die men kan uitvoeren op deze objecten. Methoden kunnen bewerkingen uitvoeren op de attributen en zo de nodige gegevens ophalen of wijzigen.

- schrijfwijze: zichtbaarheid naam(parameter:type): teruggeeftype
- zo veel mogelijk protected aangeraden
- bijvoorbeeld LeesNaam()
- WijzigNaam('Nieuwe naam')
- + maakTekst(naam:String):String

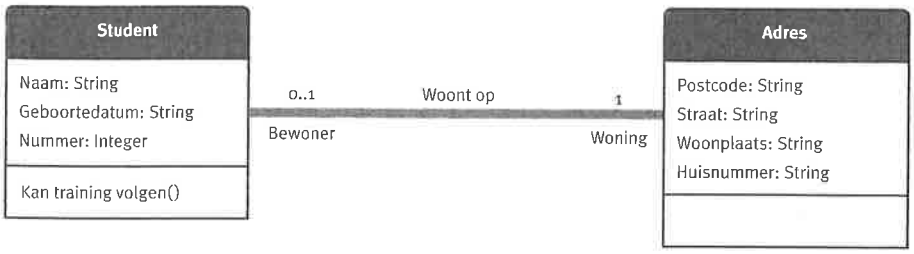
5.7 Relaties in een klassendiagram

De verschillende klassen in een klassendiagram kunnen allerlei relaties hebben. De verschillende relaties en de consequenties daarvan worden hier uitgelegd.
Als voorbeeld gebruiken we de cursusinschrijving op een school.

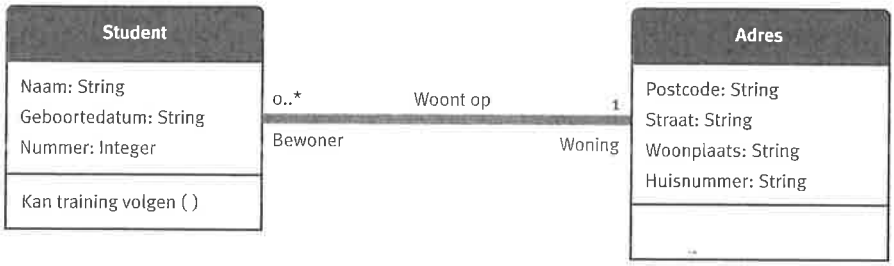
Association (associatie)



In dit klassendiagram zijn er twee klassen: Student en Adres. De klasse Student wordt gekenmerkt door de attributen (fields) Naam, Geboortedatum en Nummer. De klasse Adres wordt gekenmerkt door de attributen Postcode, Plaats, Woonplaats en Huisnummer.
Er is sprake van een associatie (association) tussen Student en Adres. In dit voorbeeld leeft er 1 student op 1 adres.
Het zou zo maar kunnen dat er op een bepaald adres *geen* studenten leven. Het bijbehorende klassendiagram gaat er dan zo uitzien. Een student leeft op 1 adres, maar op een adres leven 0 of 1 studenten.



In de praktijk komt het voor dat er meer studenten op één adres wonen. In dat geval moet het klassendiagram er zo uitzien.

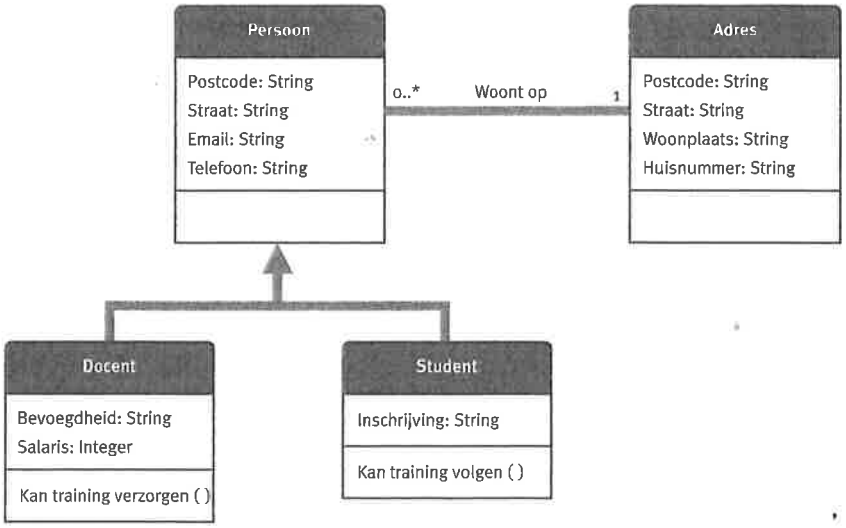


Een student leeft op 1 adres, maar op een adres leven 0 of meer studenten.
We kennen de volgende associaties:

Multipliciteit	Betekenis
1	Enkel 1
0..1	Nul of 1
0..*	Nul of meer
1..*	Een of meer
n	n (als n > 1)
0..n	Nul tot n (als n > 1)
1..n	Eén tot n (als n > 1)

Inheritance relationships (overerving)

Het komt vaak voor dat twee verschillende klassen een gedeeltelijke overeenkomst hebben. In een degelijk geval maken we gebruik van Inheritance relationships (overerving).



Student en Docent zijn beiden een persoon. Alle eigenschappen van Persoon gelden voor beiden. Ze hebben allebei een Naam, Adres, E-mail en Telefoon. De Student heeft specifiek nog een inschrijving, de Docent een Bevoegdheid en een Salaris.
Dit wordt ook wel generalisatie of specialisatie genoemd. De student heeft alles van een persoon en mogelijk meer.

- Een student is een specialisatie van persoon
- Een persoon is een generalisatie van student

De associatie geeft aan dat er op een adres 0 of meer personen wonen en dat elk persoon op 1 adres woont.
Docent en Student worden ook wel subklasse van de klasse Persoon genoemd.

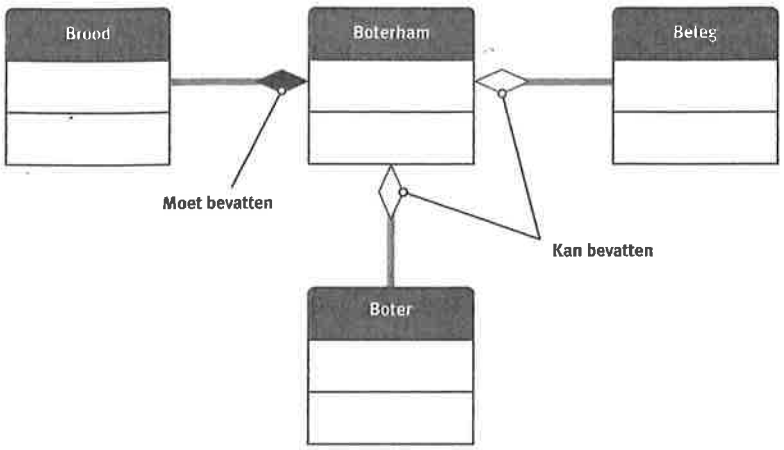
Compositie en aggregaties

Aggregaties

Aggregaties worden gebruikt voor klassen die bestaan uit twee of meer kleinere, op zichzelf staande, klassen. Aggregaties worden getekend met een witte ruitvormige pijlpunt die wijst naar het doel of de bovenliggende klasse. De bovenliggende klasse kan de onderliggende klasse bevatten.

Compositie

Compositie is een speciaal type aggregatie dat sterke relaties heeft. De bovenliggende klasse moet de onderliggende klasse bevatten. Compositie wordt getekend met een gevulde ruitvormige pijlpunt die wijst naar het doel of de bovenliggende klasse.
We nemen als voorbeeld een boterham.



Een boterham is met brood verbonden door een Compositie. De boterham *moet* brood bevatten.
Een boterham is met boter en beleg verbonden door een Aggregatie. De boterham *kan* boter en beleg bevatten.

Ontwerpen van een klassendiagram

Een klassendiagram wordt ontworpen op basis van de beschrijving van een systeem. Een manier om tot een klassendiagram te komen is dat je de informatiebehoefte goed ontleedt. Om te beginnen accentueer je alle zelfstandig naamwoorden in de tekst.
Zelfstandige naamwoorden zijn woorden die 'een zelfstandigheid' aanduiden. Dat kunnen mensen, dieren en dingen zijn. Maar het kunnen ook plaatsen en abstracte zaken als gevoelens, eigenschappen, gebeurtenissen, tijdsruimten en denkbeeldige personen of zaken zijn.
Zelfstandige naamwoorden kunnen meestal gecombineerd worden met een van de lidwoorden *de, het* of *een*. In veel zinnen staat er geen lidwoord bij het zelfstandig naamwoord:

'Hij heeft boter op zijn hoofd'
de boter –> boter is een zelfstandig naamwoord
het hoofd –> hoofd is een zelfstandig naamwoord

'Er zit muziek in'
de muziek –> muziek is een zelfstandig naamwoord

Voorbeeld ontwerpen van een klassendiagram voor TopSpin

Tennisvereniging TopSpin wil een app laten ontwikkelen voor de volgende situatie. De leden van tennisvereniging TopSpin tennissen wekelijks op woensdagavond. Zij houden een ranglijst bij. De ranglijst is eenvoudig. Je mag altijd iemand uitdagen die maximaal vijf posities hoger op de ranglijst staat. Als je wint van iemand hoger in de ranglijst, dan wissel je van positie. Per avond speelt een lid maximaal twee wedstrijden.

In de aangeleverde tekst worden de zelfstandig naamwoorden gemarkeerd.
De **leden** van **tennisvereniging** TopSpin tennissen wekelijks op **woensdagavond**. Zij houden een **ranglijst** bij. De **ranglijst** is eenvoudig. Je mag altijd **iemand** uitdagen die maximaal vijf **posities** hoger op de **ranglijst** staat. Als je wint van **iemand** hoger in de **ranglijst**, dan wissel je van **positie**. Per **avond** speelt een **lid** maximaal twee **wedstrijden**.

De zelfstandig naamwoorden worden geordend, dubbele worden weggelaten.

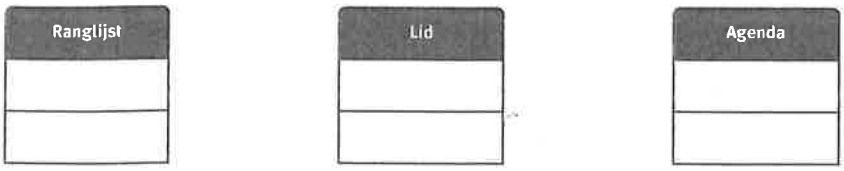
- Leden
- Tennisvereniging — Opdrachtgever TopSpin
- Woensdagavond
- Ranglijst
- Iemand
- Positie
- Avond
- Lid
- Wedstrijden

Vervolgens gaan we ze ordenen en kunnen we proberen groepen te maken.

- Leden — LID
- Tennisvereniging — Opdrachtgever TopSpin
- Woensdagavond — AGENDA
- Ranglijst — RANGLIJST
- Iemand — LID
- Positie — RANGLIJST
- Avond — AGENDA
- Lid — LID
- Wedstrijden — AGENDA

LID: Leden, Iemand en Lid behoren allemaal tot dezelfde klasse **LID**.
AGENDA: Woensdag, Avond en Wedstrijden zijn aanduidingen voor de planning en zetten we in de klasse **AGENDA**.
RANGLIJST: Ranglijst en Positie zijn beide bedoeld om ordening aan te brengen en komen samen in de klasse **RANGLIJST**.

Feitelijk krijgen we nu te maken met drie klassen:
- LID
- AGENDA
- RANGLIJST



We hebben nu de klassen benoemd en moeten de onderlinge relatie aangeven.
- Een LID kan een vermelding hebben op de RANGLIJST maar niet meer dan één (0..1). Op de RANGLIJST kunnen 0 tot veel leden gerankt zijn (0..*).
- Voor een LID kunnen één of meer speeldata worden gepland. Op de AGENDA staan 0 of meer leden gepland.

Dit zou het bijbehorende klassendiagram kunnen zijn:

Voorbeeld ontwerpen van klassendiagram Air-concepts

Air-concepts is een luchtvaartmaatschappij die een heldere planning voor elke vlucht hanteert. Ruim voor de vertrekdatum wordt het vluchtnummer en de vertrektijd vastgesteld. De vliegtuigen, piloten en overige bemanning zijn voor Air-concepts belangrijke resources. Alle werknemers, piloten en overige bemanning hebben een vliegveld als thuisbasis. Air-concepts werkt met verschillende typen vliegtuigen en afhankelijk van het type zijn er twee of drie piloten. Voor elk toestel geldt dat er een vliegveld als thuisbasis is.
Van de piloten en overige bemanning worden naam, adres en woonplaatsgegevens vastgelegd. Daarnaast wordt voor een piloot het aantal jaren ervaring vastgelegd. Voor de overige bemanning ligt vast in welke rol zij hun taak uitoefenen.
De thuisbasis van vliegtuig, piloot en overige bemanning heeft een code en een naam. De vlucht van vliegtuig, piloot en overige bemanning wordt ruim van tevoren gepland.

Om te beginnen zoek je alle zelfstandig naamwoorden. Deze hebben de potentie om in het klassendiagram te worden opgenomen
Air-concepts is een **luchtvaartmaatschappij** die een **heldere planning** voor elke **vlucht** hanteert. Ruim voor de **vertrekdatum** wordt het **vluchtnummer** en de **vertrektijd** vastgesteld. De **vliegtuigen**, **piloten** en **overige bemanning** zijn voor Air-concepts belangrijke **resources**. Alle **werknemers**, **piloten** en **overige bemanning** hebben een **vliegveld** als **thuisbasis**. Air-concepts werkt met verschillende typen **vliegtuigen** en afhankelijk van het **type** zijn er twee of drie **piloten**. Voor elk **toestel** geldt dat er een **vliegveld** als **thuisbasis** is.
Van de **piloten** en **overige bemanning** worden **naam**, **adres** en **woonplaatsgegevens** vastgelegd. Daarnaast wordt voor een **piloot** het **aantal jaren ervaring** vastgelegd. Voor de **overige bemanning** ligt vast in welke **rol** zij hun **taak** uitoefenen. De **thuisbasis** van **vliegtuig**, **piloot** en **overige bemanning** heeft een **code** en een **naam**. De **vlucht** van **vliegtuig**, **piloot** en **overige bemanning** wordt ruim van tevoren gepland.

We maken een overzicht van de zelfstandig naamwoorden waarbij we *redundante* gegevens weglaten. Redundant is de dure ICT-uitvoering van het woord 'dubbele'.

- Luchtvaartmaatschappij
- Heldere planning
- Vlucht
- Vertrekdatum
- Vluchtnummer
- Vertrektijd
- Vliegtuigen
- Piloten
- Overige bemanning
- Resources
- Werknemers
- Thuisbasis
- Verschillende typen vliegtuigen
- Type
- Vliegveld
- Naam
- Adres
- Woonplaatsgegevens
- Aantal jaren ervaring
- Rol
- Taak
- Code
- Naam Thuisbasis

Nu gaan we een groepering aanbrengen:

• Luchtvaartmaatschappij	Daar gaat het over
• **Heldere planning**	Planning vlucht
• Vlucht	Planning vlucht
• Vertrekdatum	Planning vlucht
• Vluchtnummer	Planning vlucht
• Vertrektijd	Planning vlucht
• Vliegtuigen	Vliegtuig
• Piloten	Werknemer
• Overige bemanning	Werknemer
• Resources	Werknemer
• Werknemers	Werknemer
• Thuisbasis	Basis
• Verschillende typen vliegtuigen	Vliegtuig
• Type	Vliegtuig
• Vliegveld	Basis
• Naam	Werknemer
• Adres	Werknemer
• Woonplaatsgegevens	Werknemer
• Aantal **jaren ervaring**	Werknemer (specifiek piloot)
• Rol	Werknemer (specifiek overige bemanning)
• Taak	Werknemer (specifiek overige bemanning)
• Code	Basis
• Naam Thuisbasis	Basis

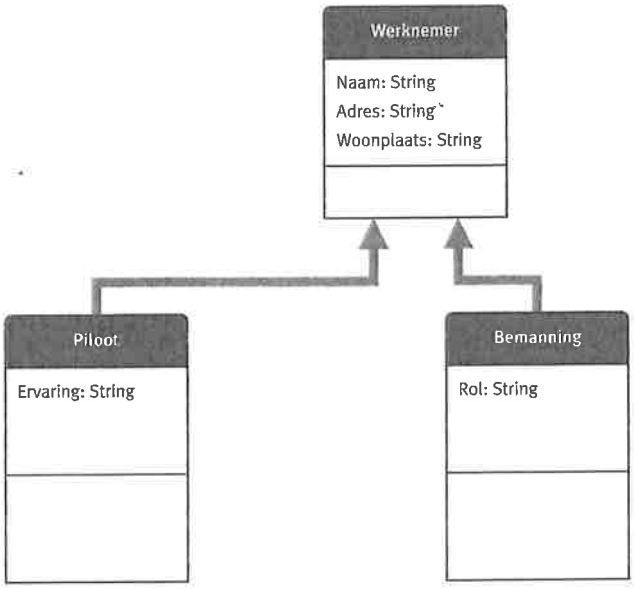
We zien een grote groep gerelateerd aan Werknemer.

• Piloten	Werknemer
• Overige bemanning	Werknemer
• Resources	Werknemer
• Werknemers	Werknemer
• Naam	Werknemer
• Adres	Werknemer
• Woonplaatsgegeven	Werknemer
• Aantal **jaren ervaring**	Werknemer (specifiek piloot)
• Rol	Werknemer (specifiek overige bemanning)
• Taak	Werknemer (specifiek overige bemanning)

Hierbij valt op dat er synoniemen worden gebruikt, die we verwijderen.

• Piloten	Werknemer
• Overige bemanning	Werknemer
• Werknemers	Werknemer
• Naam	Werknemer
• Adres	Werknemer
• Woonplaatsgegevens	Werknemer
• Aantal **jaren ervaring**	Werknemer (specifiek piloot)
• Rol	Werknemer (specifiek overige bemanning)

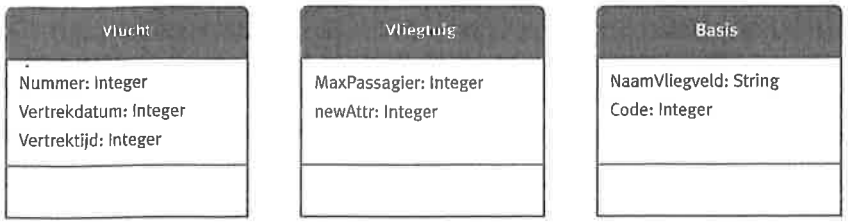
Wat opvalt is dat zowel Piloten als Overige bemanning Werknemer zijn maar dat zij elk een specifiek kenmerk hebben. Dat kan resulteren in een volgend klassendiagram.
In dit geval maken we gebruik van Inheritance relationships (overerving).



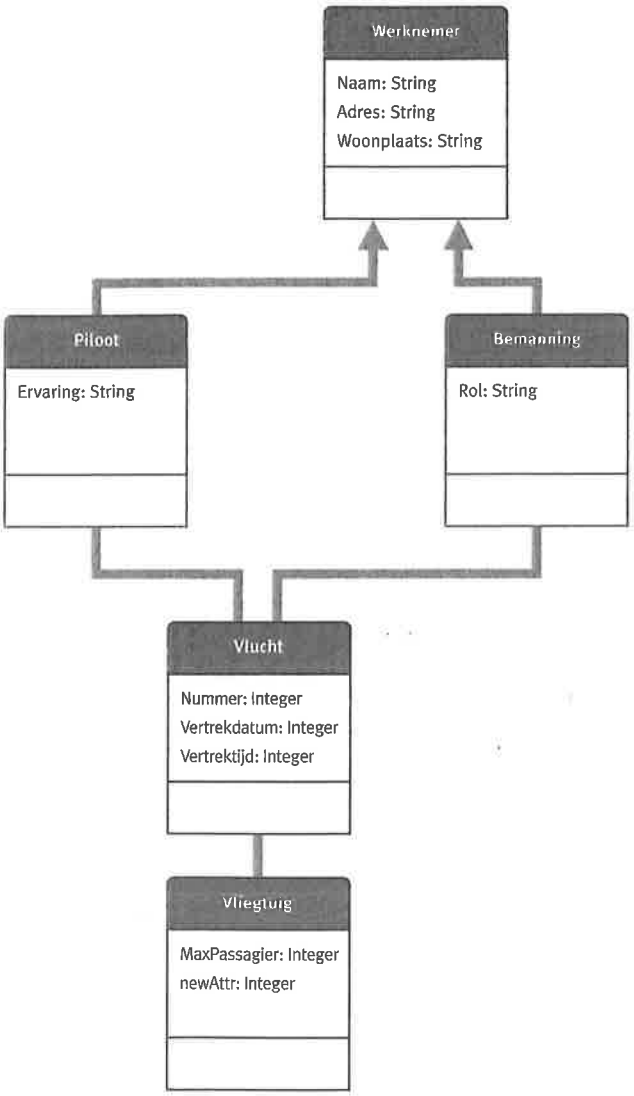
Naast de werknemers (Piloot en Bemanning) kunnen we nog de volgende groepen herkennen:

• Heldere **planning**	Planning vlucht
• Vlucht	Planning vlucht
• Vertrekdatum	Planning vlucht
• Vluchtnummer	Planning vlucht
• Vertrektijd	Planning vlucht
• Vliegtuigen	Vliegtuig
• Verschillende typen vliegtuigen	Vliegtuig
• Type	Vliegtuig
• Thuisbasis	Basis
• Code	Basis
• Naam Thuisbasis	Basis

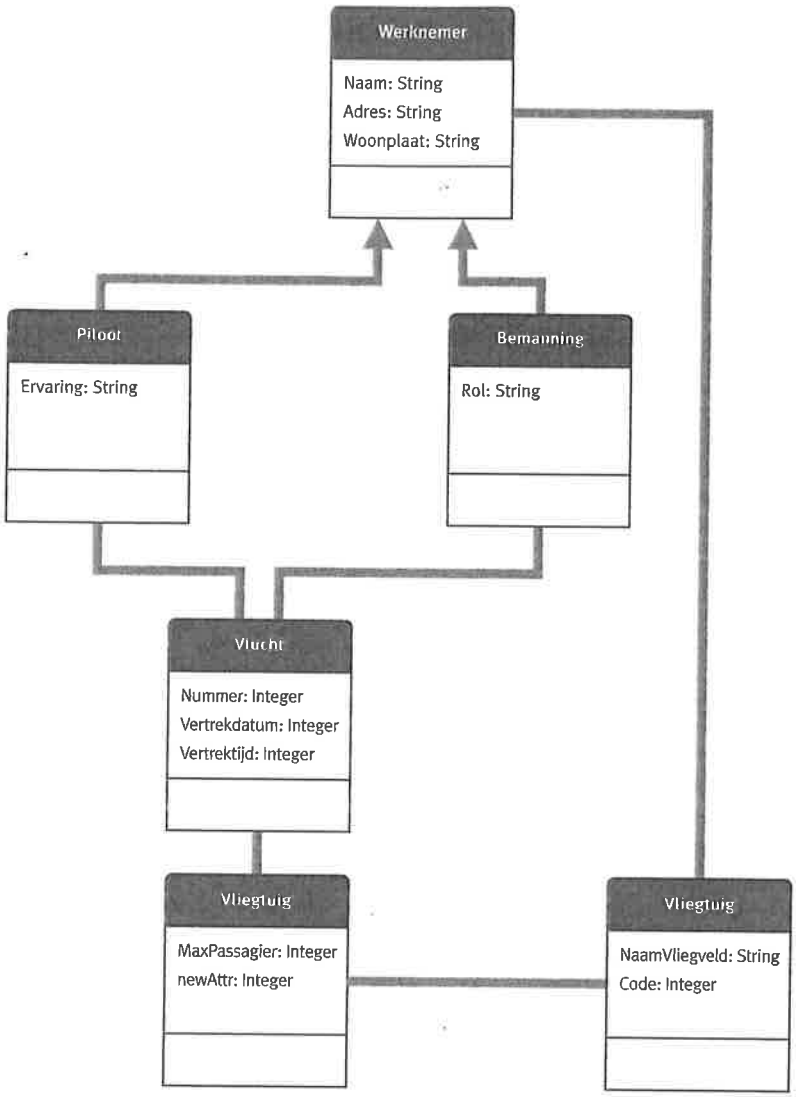
Deze tabellen geven we een adequate naam en we halen de synoniemen weg. Dan houden we de volgende klassen over:



De piloot en de bemanning zijn gepland op een bepaalde vlucht. Hieraan is natuurlijk ook een vliegtuig gekoppeld. Dat zou er zo uit kunnen zien:



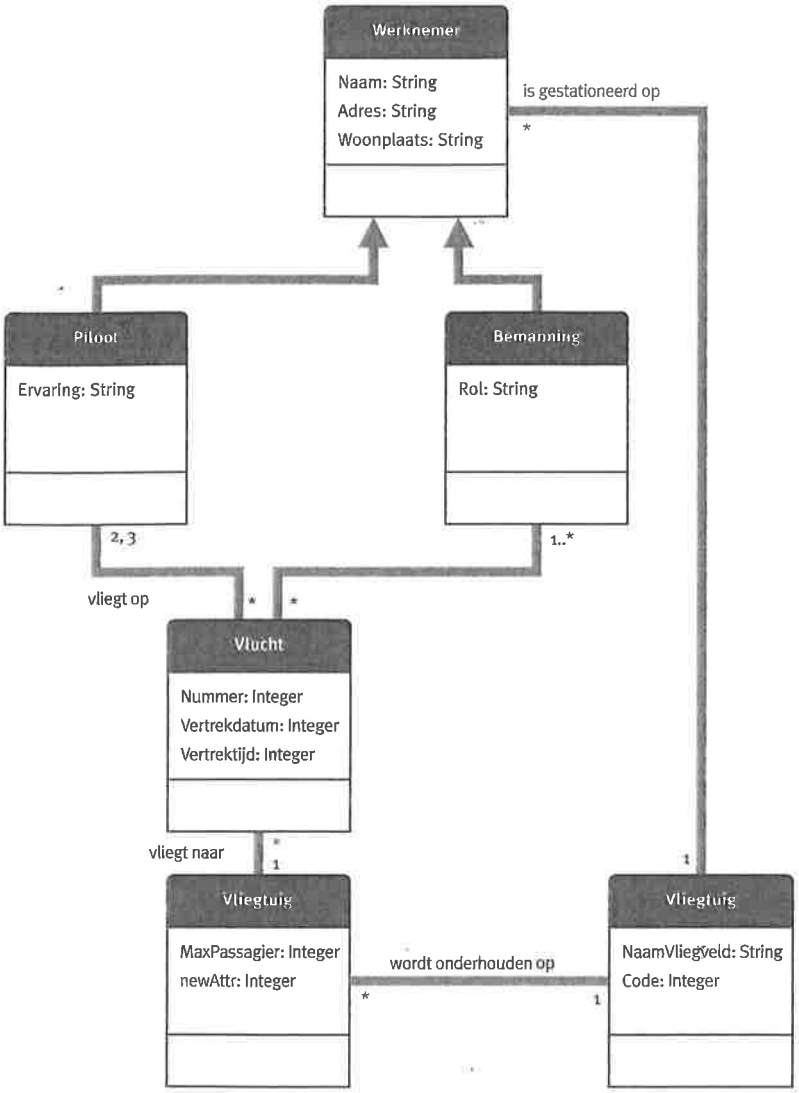
Zowel de bemanning als een vliegtuig heeft een specifiek vliegveld als thuisbasis. Dat kan worden vertaald in het volgende diagram:



Wat er nu nog moet gebeuren is dat de betreffende relaties worden gelegd.

- Een piloot kan meer verschillende vluchten doen, maar op een vlucht zitten 2 of 3 piloten.
- Bemanning kan meer verschillende vluchten doen, maar op een vlucht zit 1 of meer bemanningsleden.
- Een vliegtuig maakt meerdere vluchten, maar een bepaalde vlucht wordt slechts door één vliegtuig uitgevoerd.
- Op een basis (vliegveld) zijn meerdere vliegtuigen gestationeerd.
- Op een basis (vliegveld) zijn meerdere werknemers gestationeerd.

Het gehele klassendiagram ziet er dan zo uit:



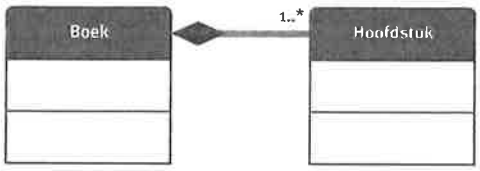
5.8 Opgaven

1 Teken het klassendiagram behorende bij de volgende tekst:
- Elke student doet mee aan twee sporten.
- Aan elke sport wordt door veel studenten deelgenomen.

2 Teken het klassendiagram behorende bij de volgende tekst:
- Een auteur heeft een of meer boeken uitgegeven.
- Een boek wordt geschreven door een of meer auteurs.
- Een boek wordt uitgegeven door precies één uitgever.
- Een uitgever geeft een of meer boeken uit.

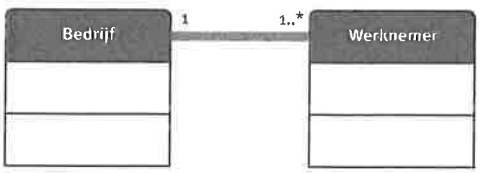
3 Het rapport van een leerling bestaat uit de naam van die leerling en een aantal resultaten. Elk resultaat is opgebouwd uit de naam van een bepaald vak en het cijfer. Teken het klassendiagram dat bij deze case hoort.

4 De onderstaande figuur is de visuele representatie van een 'xyz' in UML.



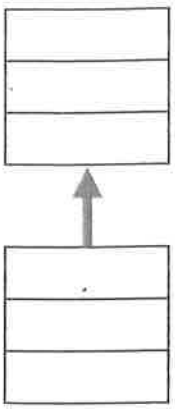
'xyz' is een:
a generalisatie
b associatie
c aggregatie
d compositie

5 De onderstaande figuur is de visuele representatie van een 'yzx' in UML.



'yzx' is een:
a generalisatie
b associatie
c aggregatie
d compositie

6 De onderstaande figuur is de visuele representatie van een 'zxy' in UML.



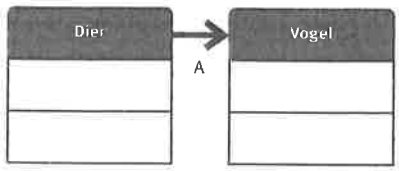
'zxy' is een:
- a generalisatie
- b associatie
- c aggregatie
- d compositie

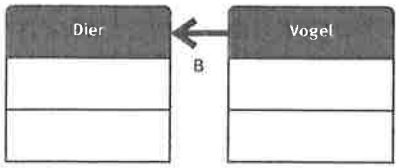
7 Attributen en methoden kunnen worden getoond met hun zichtbaarheid. Geef de functie van + aan:
- a + staat voor public attributen
- b + staat voor protected attributen
- c + staat voor private attributen
- d + wordt niet gebruikt bij zichtbaarheid

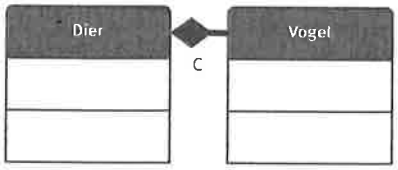
8 Attributen en methoden kunnen worden getoond met hun zichtbaarheid. Geef de functie van - aan:
- a – staat voor public attributen
- b – staat voor protected attributen
- c – staat voor private attributen
- d – wordt niet gebruikt bij zichtbaarheid

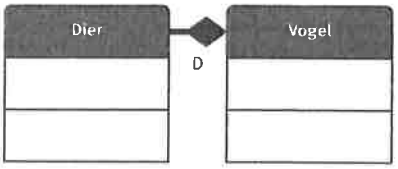
antal re-
het cijfer.

9 Welke van de onderstaande afbeeldingen geeft de generalisatie tussen vogels en dieren weer?

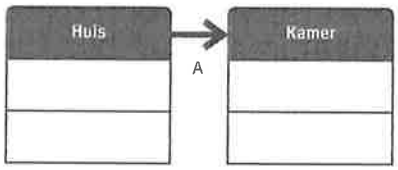


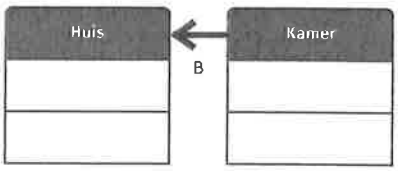


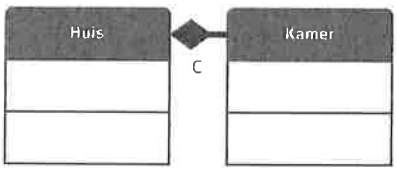


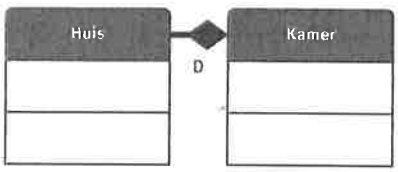


10 Welke van de onderstaande afbeeldingen geeft de compositie tussen huis en kamer weer?

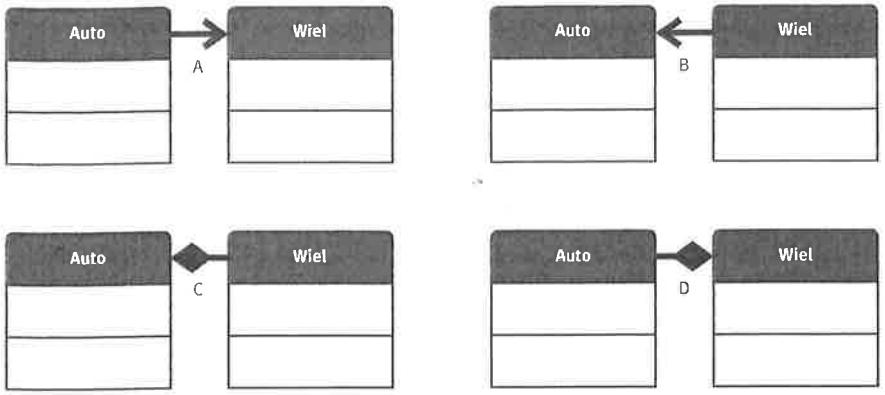




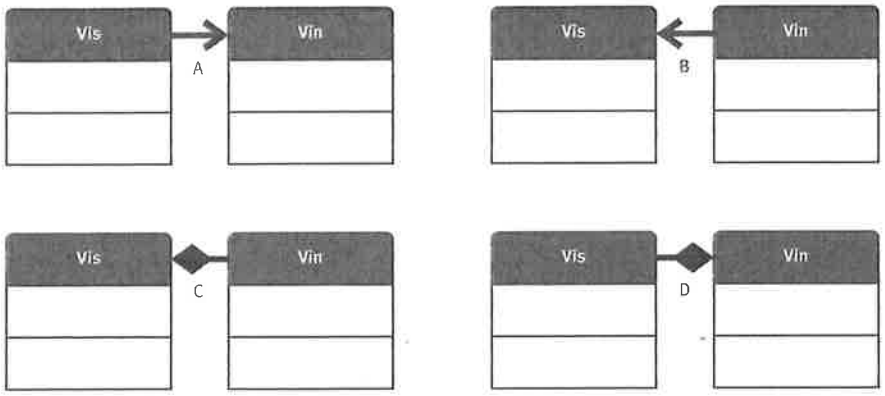




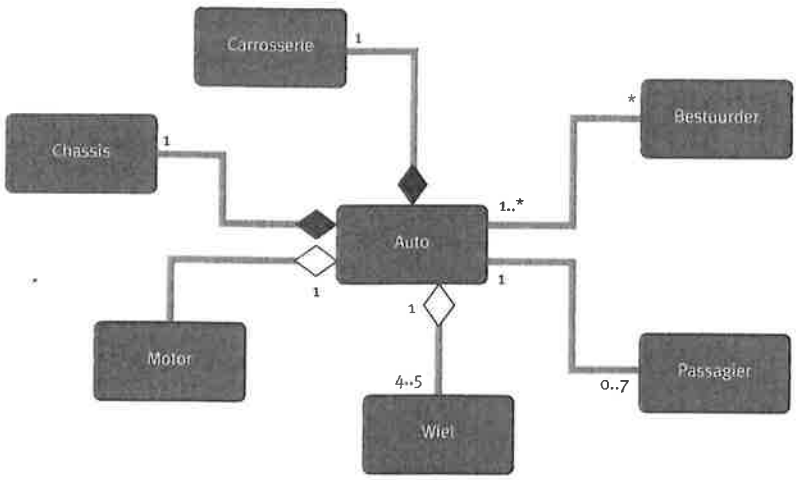
11 Welke van de onderstaande afbeeldingen beeldt uit: geen auto zonder een wiel?



12 Welke van de onderstaande afbeeldingen beeldt uit: een vis kan vinnen hebben?



13 Welke van de beweringen hoort bij onderstaande figuur?

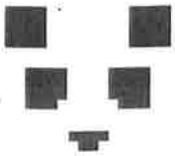


a Een auto heeft een motor
b een auto heeft minimaal één wiel
c een auto heeft een chassis
d een auto kan een carrosserie hebben

14 Brinkman Uitgeverij
Brinkman Uitgeverij geeft boeken uit van verschillende auteurs. Er zijn auteurs die één boek hebben geschreven, maar er zijn ook auteurs die meerdere boeken hebben geschreven. Ook komt het voor dat een collectief van schrijvers aan één boek heeft gewerkt.
Brinkman Uitgeverij werkt met verschillende drukkerijen, afhankelijk van de aard van het boek. Ieder boek wordt echter bij maar één drukkerij gedrukt. Een redacteur van Brinkman Uitgeverij werkt met meerdere auteurs. De redacteur verzorgt voor elk van hen de redactie en de productie.
Maak hiervan een klassendiagram.

15 Tennisclub TopSpin
Bij tennisclub TopSpin wordt een overzicht bijgehouden van de leden. Van alle leden zijn persoonlijke gegevens bekend als naam en adresgegevens. Een lid heeft een uniek lidnummer. Verder is bekend wat voor soort speler het lid is. Een lid kan junior, senior of recreatiespeler zijn. Ook is de jaarlijks contributie vastgesteld, die per soort speler vast staat. Ten slotte is bekend voor welk team een speler uitkomt in de competitie. Van een team wordt de klasse genoteerd. Recreanten zijn niet ingedeeld in een team, want zij spelen geen competitie.
Maak hiervan een klassendiagram.

16 Muziek Stimulatie Square Panda



Muziek Stimulatie Square Panda wil informatie opslaan over de muzikanten die optreden op alle uitgebrachte albums. De volgende gegevens zijn van belang:

- Van elke muzikant die opnames maakt bij Square Panda is het burgerservicenummer, de naam, het adres en het telefoonnummer bekend.
- Er wordt bijgehouden welke instrumenten worden gebruikt.
- Elk instrument heeft een naam (gitaar, synthesizer, fluit enzovoort) en een type (dat kan alleen blaas, snaar of slag zijn). Als het instrument een blaasinstrument is, moet de toonsoort (bijvoorbeeld C, Bes, Es) worden bijgehouden.
- Elk album dat wordt opgenomen onder het Square-Panda-label heeft een titel, een copyright-datum een formaat (cd of dvd) en een album-id.
- Elk nummer dat wordt opgenomen bij Square Panda heeft een (unieke) titel en een auteur.
- Elke muzikant kan één of meer instrumenten bespelen en een instrument kan door meerdere muzikanten bespeeld worden.
- Op elk album staat een aantal nummers, maar geen enkel nummer mag op meerdere albums voorkomen.
- Elk nummer wordt gespeeld door één of meer muzikanten en een muzikant kan één of meer nummers spelen. Hierbij wil Square Panda bijhouden welk instrument de muzikant op dit specifieke nummer speelt (dat kan er slechts één zijn).
- Op elk album is er precies één muzikant die optreedt als producer. Een muzikant kan uiteraard meerdere albums producen.

Stel aan de hand van bovengenoemde gegevens een klassendiagram op waarin dit alles wordt weergegeven.

5.9 Functioneel-ontwerp-rapport

Het functioneel-ontwerp-rapport is een document dat bedoeld is voor de klant/opdrachtgever. Bij het schrijven van dit rapport moet je hier ook constant rekening mee houden. Zorg dat alles wat je opschrijft begrepen kan worden!

Indeling functioneel-ontwerp-rapport

Voorblad
Voorwoord
Inhoudsopgave
Samenvatting
Requirements

Analyse huidige situatie
    Informatieverwerking
    Applicaties
    Infrastructuur
Analyse gewenste situatie
    Informatieverwerking
    Applicaties
    Infrastructuur
Consequenties
    Organisatorische consequenties
    Technische consequenties
Kosten
Planning

Voorwoord

In het voorwoord schrijf je waarom je dit document hebt gemaakt. Je beschrijft het bedrijf waarin je werkt, het bedrijf van de klant en het doel van het project voor je klant. Zijn er nog andere zaken die genoemd moeten worden, maar die buiten het document vallen, dan is dit de plaats om ze alsnog te vermelden. Hierbij kun je denken aan de dank die je wilt uitspreken voor de hulp die je gekregen hebt van iemand.

Inhoudsopgave

De inhoudsopgave wordt zelf niet genoemd in de inhoudsopgave maar is wel een belangrijk onderdeel. Elke tekstverwerker kent de optie automatisch genereren, natuurlijk gebruik je die.

Samenvatting

Op basis van een (management)samenvatting moet de projectmanager en/of opdrachtgever besluiten of het project kan doorgaan. De samenvatting is compact en bevat concrete feiten die belangrijk zijn voor de besluitvorming.

Requirements

De MoSCoW-methode is het belangrijkste instrument voor dit hoofdstuk. Je kunt de resultaten het best conform deze methode rangschikken.

- Must haves
- Should haves
- Could haves
- Won't haves

Analyse huidige situatie

Om het verschil te kunnen zien tussen de situatie zoals die nu is en de toekomstige situatie is het belangrijk de huidige situatie nauwkeurig in kaart te brengen.

Informatieverwerking

Hier beschrijf je de bestaande situatie. Je geeft de bedrijfsstructuur weer met een organogram. Tevens geef je met behulp van een use-case-diagram de functionaliteit van de verschillende processen weer. De verschillende use cases worden hier uitgewerkt. Zo mogelijk geef je de relatie tot de verschillende bedrijfsonderdelen weer.

Applicaties

Je geeft hier een opsomming van applicaties die op dit moment worden gebruikt, bij voorkeur gekoppeld aan de daaraan verbonden bedrijfsonderdelen en use cases.

Infrastructuur

Als er sprake is van een netwerkinfrastructuur moet deze hier grafisch worden weergegeven.

Analyse gewenste situatie

Nadat de bestaande situatie in kaart is gebracht is het zaak om de gewenste situatie duidelijk te krijgen. Lastig is hierbij dat de gebruikers vaak niet exact weten wat ze willen of niet weten wat mogelijk is. De MoSCoW-methode kan hierbij een belangrijk hulpmiddel zijn.

Informatieverwerking

Hier beschrijf je de nieuwe situatie, bij voorkeur met behulp van een use-case-diagram. Vervolgens werk je de verschillende use cases uit.

Applicaties

Hier geef je een opsomming van de applicaties die op dit moment worden gebruikt, bij voorkeur gekoppeld aan de daaraan verbonden use cases van de nieuwe situatie. Het kan zijn dat je hier wireframes opneemt van de nieuwe applicatie/situatie. Het kan ook dat er de voorkeur aan wordt gegeven de wireframes in het technisch ontwerp op te nemen.

Infrastructuur

Als er sprake is van een wijziging in de netwerkinfrastructuur moet deze hier grafisch worden weergegeven.

Consequenties

Als de nieuwe situatie veranderingen tot gevolg heeft dan is het belangrijk deze vooraf te benoemen.

Organisatorische consequenties
Hier moet alles worden beschreven dat invloed heeft op de organisatie. Als bedrijfsprocessen anders gaan verlopen, dan zou er wel eens een training gegeven moeten worden. De tijd die hiervoor moet worden vrijgemaakt zal benoemd moeten worden. Het zou zelfs zo kunnen zijn dat personeelsleden hun huidige werk verliezen of er ander werk bij krijgen. Deze personeelsleden zullen zo nodig moeten worden omgeschoold.

Technische consequenties
Alle wijzigingen aan de infrastructuur, alle materialen/apparatuur die extra moeten worden aangeschaft moeten hier worden vermeld.

Kosten
In deze fase is het alleen mogelijk een globale beschrijving van de kosten te maken. Het is onmogelijk om nu al voor het hele project alles tot op de euro te specificeren. Toch is het belangrijk de kosten voor hardware, software en voor ontwikkeling te noemen. De uren voor de fase ontwerp zijn natuurlijk wel gespecificeerd.

Planning
Hier neem je de bijgestelde planning uit het Projectplan over.