**Voorbeeld ontwerpen klassediagrammen (Hfdst. 3)**

Achtergrondinformatie

Hans en Jacqueline gaan samenwonen. Ze hebben een verdieping van een Amsterdams grachtenpand kunnen kopen, omdat ze beiden goed verdienen. Jacqueline is piloot bij een luchtvaartmaatschappij en vliegt vanaf Schiphol. Hans is arts in een academisch ziekenhuis in Amsterdam. Beiden hebben vanwege hun werk een zeer wisselend weekschema. Ze voorzien dat er hierdoor problemen zullen ontstaan met het doen van inkopen voor de dagelijkse maaltijd. Daarom willen ze een computersysteem laten maken dat draait op hun nieuwste aanwinst: een supersnelle pc. Ze hebben zelfs al een naam voor het systeem gedacht. Het gaat EasyShop heten.

De informatici die het systeem zullen bouwen komen een avond praten over hoe het er uit moet komen te zien. Globaal komt het hierop neer: (=context)

*Hans en Jacqueline moeten onafhankelijk van elkaar het systeem kunnen gebruiken. Ze willen per week een scherm zien waarop ze kunnen aangeven of ze wel of niet aanwezig zijn bij het diner, lunch en/of ontbijt. Bovendien moet aangegeven kunnen worden of er gasten verwacht worden. Het is de bedoeling dat elke huisgenoot alleen voor zichzelf de agenda invult.*

*Per dag wordt bovendien aangegeven wie er kookt. Deze persoon zal uit een aantal recepten kiezen wat er die dag gegeten wordt.*

*Op basis van de gegeven recepten plus informatie over de meest geprefereerde lunch en ontbijt van zowel Hans als Jacqueline zal het systeem één keer per week een boodschappenlijst maken. Hans of Jacqueline zal deze actie initiëren, want aanwezigheid van een van hen is vereist voor de bezorging. Op dagen die niet ingevuld zijn, wordt ervan uitgegaan dat er niemand aanwezig zal zijn.*

*Deze boodschappenlijst zal per fax naar de lokale supermarkt gestuurd worden, waarna de supermarkt de gevraagde boodschappen thuisbrengt. Het systeem bewaart een lijst met faxnummers van lokale supermarkten met een bezorgdienst. De gebruiker kan uit deze lijst kiezen.*

*Het systeem zal de voorraad van het huishouden bijhouden. Wanneer de boodschappen bezorgd worden zal de gebruiker deze met de juiste hoeveelheden invoeren in het systeem. Wanneer een dag verstreken is, dan gaat het systeem ervan uit dat de maaltijden die dag genuttigd zijn.*

*Hans en Jacqueline zijn allebei geen ster in de keuken, daarom willen ze een kookboek met een groot aantal eenvoudige recepten in het systeem onderbrengen. Dit kookboek moet onafhankelijk van de agenda in te zien zijn, bijvoorbeeld op het moment dat gekookt moet worden.*

**Opdrachten**

1. Identificeer alle mogelijke kandidaatklassen. Ga dus alle zelfstandige naamwoorden onderstrepen.
2. Ga hierna alle zelfstandige naamwoorden goed bekijken en geef aan welke beslissing je hebt genomen m.b.t. geschiktheid. (Denk er aan ze op alfabetische volgorde te zetten.)

Stel bij elke kandidaatklasse de volgende vragen:

* Speelt de kandidaatklasse een zelfstandige rol in het probleemdomein?
* Wil je iets van de kandidaat weten of wil je dat hij iets voor je doet?
* Heeft de kandidaatklasse een duidelijke verantwoordelijkheid in het systeem?

Wanneer je deze vragen met ‘ja’ beantwoordt, dan is de kandidaat een klasse.

Behalve het positief beantwoorden van deze vragen kun je ook kijken waarom een kandidaat géén klasse zou zijn. Bijv. redundant met een andere klasse, irrelevant, vaag, niet kwantificeerbaar, attributen, operaties,

Houd hierbij het volgende schema aan waarin al een paar voorbeelden zijn gezet om het te verduidelijken:

|  |  |
| --- | --- |
| Kandidaatklasse | Beslissing m.b.t. geschiktheid |
| Actie | Te vaag |
|  |  |
| Bedoeling | Irrelevant |
|  |  |
| Bezorging | Irrelevant |
|  |  |
|  |  |
| Dag | Klasse |

Uitwerking:

1.

*Hans en Jacqueline moeten onafhankelijk van elkaar het* ***systeem*** *kunnen gebruiken. Ze willen per* ***week*** *een* ***scherm*** *zien waarop ze kunnen aangeven of ze wel of niet aanwezig zijn bij het* ***diner****,* ***lunch*** *en/of* ***ontbijt****. Bovendien moet aangegeven kunnen worden of er* ***gasten*** *verwacht worden. Het is de* ***bedoeling*** *dat elke* ***huisgenoot*** *alleen voor zichzelf de* ***agenda*** *invult.*

*Per* ***dag*** *wordt bovendien aangegeven wie er kookt. Deze* ***persoon*** *zal uit een aantal* ***recepten*** *kiezen wat er die dag gegeten wordt.*

*Op basis van de gegeven recepten plus* ***informatie*** *over de meest geprefereerde lunch en ontbijt van zowel Hans als Jacqueline zal het systeem één keer per week een* ***boodschappenlijst*** *maken. Hans of Jacqueline zal deze* ***actie*** *initiëren, want aanwezigheid van een van hen is vereist voor de* ***bezorging****. Op dagen die niet ingevuld zijn, wordt ervan uitgegaan dat er niemand aanwezig zal zijn.*

*Deze boodschappenlijst zal per* ***fax*** *naar de lokale* ***supermarkt*** *gestuurd worden, waarna de* ***supermarkt*** *de gevraagde* ***boodschappen*** *thuisbrengt. Het systeem bewaart een lijst met* ***faxnummers*** *van lokale supermarkten met een* ***bezorgdienst****. De* ***gebruiker*** *kan uit deze lijst kiezen.*

*Het systeem zal de* ***voorraad*** *van het* ***huishouden*** *bijhouden. Wanneer de boodschappen bezorgd worden zal de gebruiker deze met de juiste* ***hoeveelheden*** *invoeren in het systeem. Wanneer een dag verstreken is, dan gaat het systeem er vanuit dat de* ***maaltijden*** *die dag genuttigd zijn.*

*Hans en Jacqueline zijn allebei geen* ***ster*** *in de* ***keuken****, daarom willen ze een* ***kookboek*** *met een groot aantal eenvoudige recepten in het systeem onderbrengen. Dit kookboek moet onafhankelijk van de agenda in te zien zijn, bijvoorbeeld op het moment dat gekookt moet worden.*

2.

Resultaten van klasseselectie

Kandidaatklasse Beslissing m.b.t. geschiktheid

Actie te vaag

Agenda klasse

Bedoeling irrelevant

Bezorgdienst irrelevant

Bezorging irrelevant

Boodschap klasse

Boodschappenlijst associatie (Is lijst van meerdere instanties van Boodschap)

Dag klasse

Diner klasse (verantwoordelijkheid is: bijhouden wie de kok is.)

Fax implementatie

Faxnummer attribuut

Gast redundant met Persoon

Gebruiker redundant met Persoon

Hoeveelheid te vaag, attribuut

Huisgenoot redundant met Persoon

Huishouden irrelevant

Informatie te vaag

Keuken irrelevant

Kookboek klasse

Lunch klasse

Maaltijd klasse

Ontbijt klasse

Persoon klasse

Recept klasse

Scherm implementatie

Ster irrelevant

Supermarkt klasse

Systeem te vaag

Voorraad attribuut

Week klasse

**Maak een modeldictionary**

In deze stap moeten alle klassen eenduidig gedefinieerd worden. Vaak kost dit enige moeite. De klasse Agenda is zo’n probleemgeval. Wat betekent een dergelijke klasse voor ons systeem? Welke verantwoordelijkheid kunnen we eraan toekennen? Voor ons systeem is een agenda een verzameling van weken, dagen en maaltijden, waaraan informatie gekoppeld is over de aanwezigheid van personen. Het lijkt dat we deze agenda al tot in detail hebben gemodelleerd door de klassen Week, Dag, Maaltijd en Persoon op te nemen. Het is alleen wel zaak de juiste associaties tussen de klassen te leggen. De conclusie is dus dat de klasse Agenda uit het systeem verdwijnt.

**Modeldictionary**

Klasse Omschrijving

Dag …

Diner Een maaltijd waarbij warm voedsel genuttigd wordt dat wordt bereid volgens een *Recept*. *Diner* is verantwoordelijk voor het bijhouden van de aanwezigheid van *Personen* en voor het bijhouden van gegevens over de kok.

Kookboek …

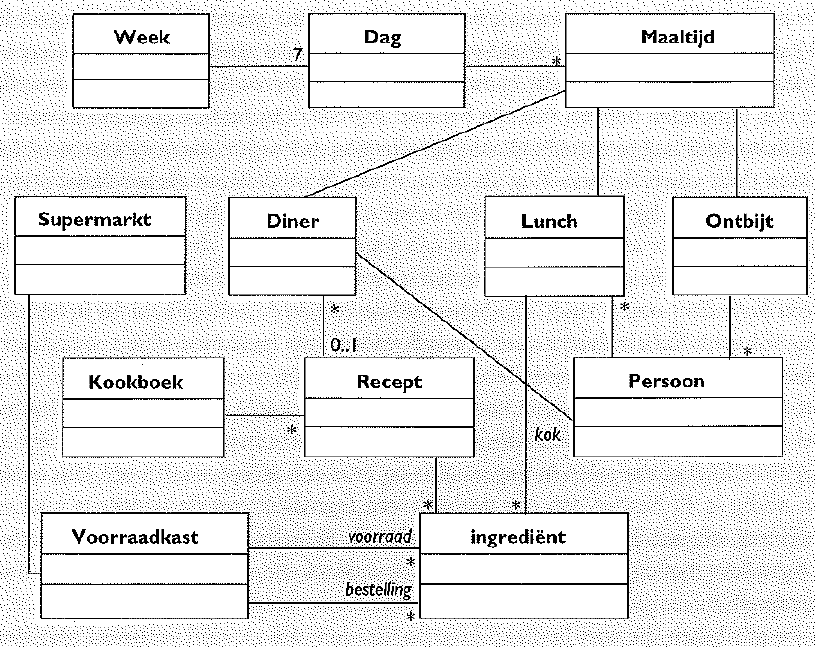
|  |  |
| --- | --- |
| Klasse | Omschrijving |
| Dag | Afgeronde tijdseenheid waarbinnen een van tevoren bepaald aantal maaltijden valt. Elke maaltijd is ofwel een *Ontbijt*, een *Lunch*, ofwel een *Diner* |
| Diner | Een maaltijd waarbij warm voedsel genuttigd wordt dat wordt bereid volgens een *Recept*. *Diner* is verantwoordelijk voor het bijhouden van de aanwezigheid van *Personen* en voor het bijhouden van gegevens over de kok. |
| Ingrediënt | Voedselbestanddeel dat gebruikt kan worden bij de bereiding van een *Recept* en/of het nuttigen van een *Diner*, *Lunch* of *Ontbijt* |
| Kookboek | Verzameling Recepten. Is verantwoordelijk voor zoekmethoden binnen de verzameling |
| Lunch | Maaltijd waarbij voedsel genuttigd wordt dat niet wordt bereid volgens een *Recept*. |
| Maaltijd | Moment in tijd waarop voedsel genuttigd wordt. |
| Ontbijt | Maaltijd waarbij voedsel genuttigd wordt dat niet wordt bereid volgens een *Recept*. |
| Persoon | Representatie binnen het systeem van een reëel persoon. |
| Recept | Wijze om van Ingrediënten warm voedsel te maken. |
| Supermarkt | Representatie binnen het systeem van een reële supermarkt |
| Voorraadkast | Beheerder van de voorraad. Is verantwoordelijk voor het bijhouden van de hoeveelheden van de Ingrediënten die in voorraad zijn en voor het opstellen en versturen van een bestelling. |
| Week | Samenvoeging van zeven opeenvolgende Dagen, te weten zondag tot en met zaterdag. Is verantwoordelijk voor het presenteren van de agenda aan de gebruiker. |

Je ziet dat hier Ingrediënt en Voorraadkast aan toegevoegd zijn. Ingrediënt is in plaats gekomen van Boodschap. De voorraadkast is verantwoordelijk voor alle zaken die betrekking hebben op Ingrediënten, het bestellen ervan, nagaan of er voldoende voorraad van in huis is, enz.

(Waarschijnlijk is dit een *singleton* klasse, dat wil zeggen dat er maar één instantie van in het systeem zal zijn.)

Hierna identificeren we alle mogelijke kandidaat-associaties. In principe zijn dit alle combinaties van twee klassen uit ons model. Een aantal associaties wordt ons direct aangereikt vanuit de modeldictionary. Een dag heeft een relatie met een aantal Maaltijden, een Week met zeven Dagen, een Recept met een Diner en meerdere Ingrediënten, enz.

We komen dan tot een eerste versie van het klassediagram:



Nu moeten de attributen worden toegekend aan één of meer klassen. Eenzelfde attribuut kan aan meerdere klassen worden toegekend.

Het nieuwe attribuut Naam kennen we toe aan Dag, Recept, Persoon en Maaltijd.

Het nieuwe attribuut Type kennen we toe aan de klasse Ingrediënt.

Prijs is een attribuut bij Ingrediënt. Vraag is of dit geregistreerd moet worden. Wordt nl. buiten het systeem om bepaald en is bovendien afhankelijk van de keuze van de supermarkt.

Kooktijd is een attribuut bij Recept, maar de naam Bereidingstijd geeft beter weer wat er mee bedoeld wordt.

Bereidingswijze is duidelijk een attribuut bij Recept.

Voorraadkast moet ook bijhouden hoeveel van een bepaald Ingrediënt in bestelling is. We kunnen dit modelleren door nog een attribuut aan Ingrediënt toe te voegen, maar we kunnen dit ook oplossen door zowel de bestelling als de voorraad te modelleren als een associatie van Voorraadkast met Ingrediënt. Ingrediënt bevat nu alleen een attribuut Hoeveelheid. Als Voorraadkast via de associatie Voorraad een Ingrediënt-object vindt, dan betekent Hoeveelheid de hoeveelheid die nog in voorraad is en als Voorraadkast via de associatie Bestelling een Ingrediënt-object vindt, dan betekent Hoeveelheid de hoeveelheid die in bestelling is. Evenzo geldt dat als Recept naar hoeveelheid van een Ingrediënt-object vraagt het de betekenis heeft van hoeveelheid die nodig is in dat recept.

Als een soort ingrediënt zowel in voorraad is, als in recepten gebruikt wordt, dan heeft zowel de Voorraadkast als ieder Recept zijn eigen aparte instantie van Ingrediënt. Stel dat er twee kilo aardappelen in voorraad is en er is een recept waarin één kilo aardappelen nodig is. Er bestaat dan een instantie van Ingrediënt met de waarde “2 kilo” voor Hoeveelheid, die een link met Voorraadkast heeft. Er bestaat ook een tweede instantie van Ingrediënt met de waarde “1 kilo” voor Hoeveelheid, die een link met het betreffende Recept heeft.

Verder voegen we nog wat extra attributen toe zoals Moeilijkheidsgraad bij Recept, Nummer bij Week, Naam bij Maaltijd en BestellingVerzondenOp bij Voorraadkast. Omdat we verschillende soorten van personen hebben, voegen we een attribuut Type toe aan Persoon.

Op dit moment laten we het hierbij. Andere attributen worden meestal snel gevonden in volgende iteratieslagen.

We komen dan tot de tweede versie van het klassediagram:

