Les F-02 UML

In deze lesbrief wordt globaal beschreven wat **Unified Modeling Language (UML)** inhoudt. UML is een **modelleertaal**. Dat wil zeggen dat je daarmee de objecten binnen een (informatie)systeem modelmatig kunt beschrijven.

In module E heb je al kennis gemaakt met het modelleren van systemen aan de hand van contextdiagrammen, dataflowdiagrammen, strokendiagrammen en ERD's. De modelleertaal UML is specifiek bedoeld voor situaties waarin object georiënteerd wordt gewerkt.

Doel van deze lesbrief is om kennis en inzicht te verschaffen in wat UML inhoudt. Daartoe worden enkele basisbegrippen toegelicht. De kennis die je in de lesbrieven over UML opdoet kan je ook toepassen door de gratis UML software uit te proberen (zie download).

1.1 Modelleren

Modelleren is het maken van modellen om daarmee de realiteit in kaart te brengen.
Een **model** wordt dan ook wel gedefinieerd als een vereenvoudiging van de werkelijkheid.

Een architect zal met tekeningen en maquettes de realiteit modelleren die hij voor ogen heeft. Let wel: die realiteit hoeft dus (nog) niet te bestaan! De maquettes en tekeningen dienen ervoor om met alle partijen (aannemers, kopers) te kunnen communiceren.
Ook een gamedesigner zal zijn ideeën op een overzichtelijke manier willen weergeven. Hij is tenslotte de architect van zijn game! Maar ook de gamedesigner zal met veel partijen over zijn ideeën moeten communiceren, bijvoorbeeld met de bedrijfsleiding of met programmeurs.

Door te modelleren krijg je een beter begrip van de realiteit die je modelleert. Als je een simpel systeem gaat bouwen is heb je vaak voldoende overzicht. In veel situaties waarin eenvoudige systemen worden gebouwd wordt de fase van het modelleren dan ook wel overgeslagen.
Zodra een systeem complexer wordt krijg je te maken met meer eisen en wensen. Het kan dan handig zijn om te gaan modelleren. Doordat modellen simpeler zijn dan de werkelijkheid kunnen zijn deze **sneller te overzien**. Daarnaast kunnen we modellen ook in andere, **begrijpelijkere verschijningsvormen** maken (zoals een tekening op papier i.p.v. software-broncode). Tenslotte zijn de **productiekosten** van modellen doorgaans lager zodat we deze sneller kunnen ontwikkelen, aanpassen en overnieuw kunnen beginnen.

1.2 UML

UML is de afkorting van Unified Modeling Language. UML is dus een universele modeleertaal. UML komt uit de software ontwikkeling en is ontwikkeld door Grady Booch, James Rumbaugh en Ivar Jacobson. Deze drie (ook wel de drie amigo's genoemd) hadden voor UML elk een eigen taal, respectievelijk Booch95, Object Modelling Technique (OMT) en Object Oriented Software Engineering (OOSE). Zij hebben hun drie eigen talen gecombineerd en gestandaardiseerd in UML.

Van UML zijn verschillende versies verschenen. In deze lesbrief beperken we ons voornamelijk tot UML 1.0.

Wat kan je met UML?

- **visualiseren**: een model kan je met UML zichtbaar maken en er dus een voorstelling van maken.
- **specificeren**: je kan van modellen zeggen wat ze kunnen en hoe ze zich gedragen.
- **construeren**: je kan de opbouw van modellen bepalen.
- **documenteren**: je maakt het mogelijk om wat je met UML ontwikkeld te bewaren en later te bekijken.

1.3 Use Case Diagrammen

Geen enkel systeem is geïsoleerd. Er is altijd iets dat interacteert met een systeem. Dit kan zowel een menselijk 'iets' zijn als een automatisch/geautomatiseerd 'iets'. Dit 'iets' noemen we een **actor**. Een actor is dus een mens, een proces of ander systeem dat interactie heeft met het systeem.

Een actor heeft een verwachting bij de werking van het systeem. De actor wil/kan het systeem op een bepaalde manier gebruiken. Deze interactie en wensen, inclusief varianten en uitzonderingssituaties kan worden vastgelegd in een **use case**.

Een use case kan in verschillende situaties en omstandigheden worden uitgevoerd. We spreken dan van een **scenario**.

Als voorbeeld bekijken we het systeem "pinautomaat". Er zijn verschillende actoren actief rondom dit systeem. Zo is er de klant die geld wil pinnen of zijn saldo wil inzien, de bankmedewerker die de automaat wil bijvullen of de onderhoudsmedewerker die een onderhoudsbeurt of reparatie wil uitvoeren.

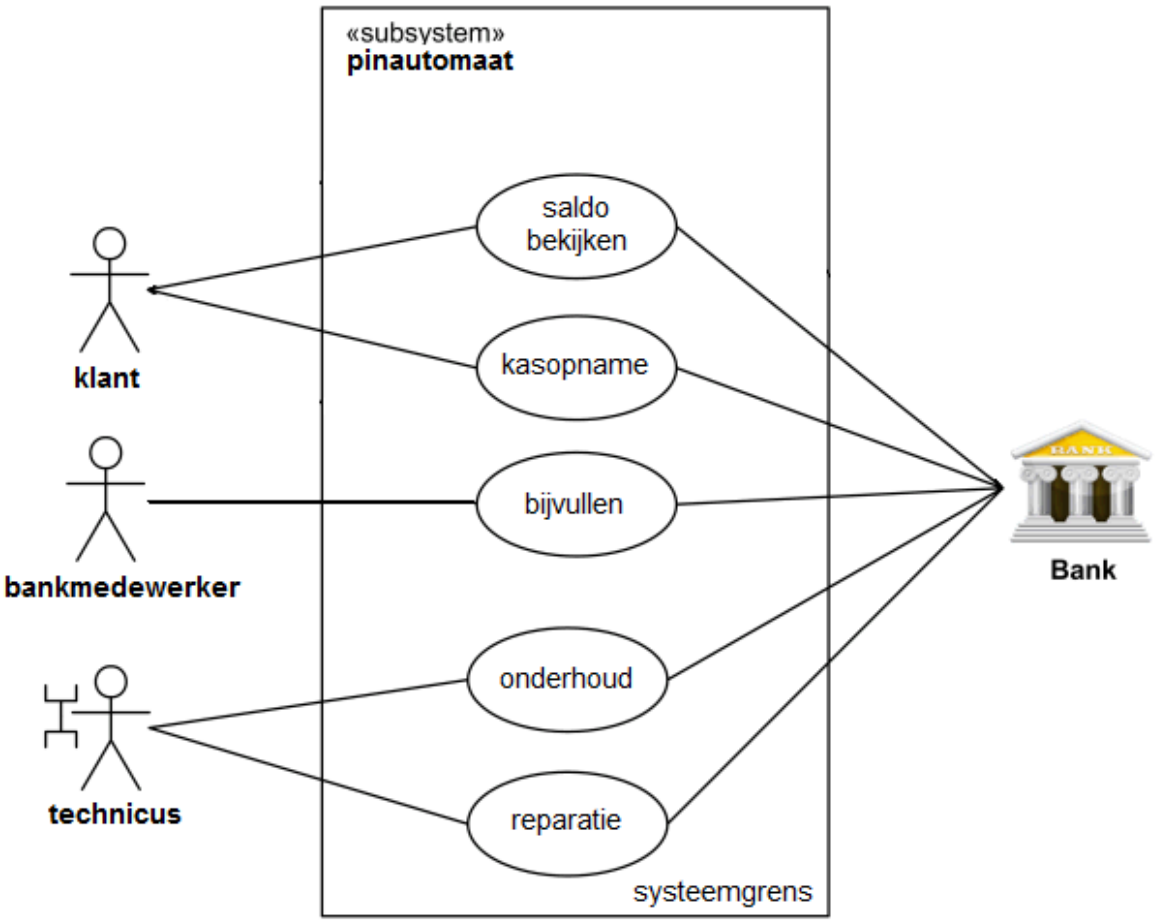
We onderscheiden dus:

actoren:	klant, bankmedewerker, onderhoudsmedewerker
use cases:	saldo bekijken, pinnen (kasopname), bijvullen, onderhoud en reparatie

Bij use cases maken we gebruik van **use case diagrammen** voor het overzicht en doen we de werkelijke beschrijving in zogenaamde **use case templates**, ook wel use case beschrijvingen genoemd. Samen vormen zij het beschrijvend (UML) model.

Het doel van de use case diagrammen en templates is om voorafgaand aan het systeemontwikkelingsproces te kunnen overleggen over het te ontwikkelen systeem (met opdrachtgevers en projectteam) en achteraf documentatie (een functionele beschrijving) mee te leveren als het product wordt opgeleverd.

Op de volgende pagina tref je het use case diagram aan van het systeem "pinautomaat".



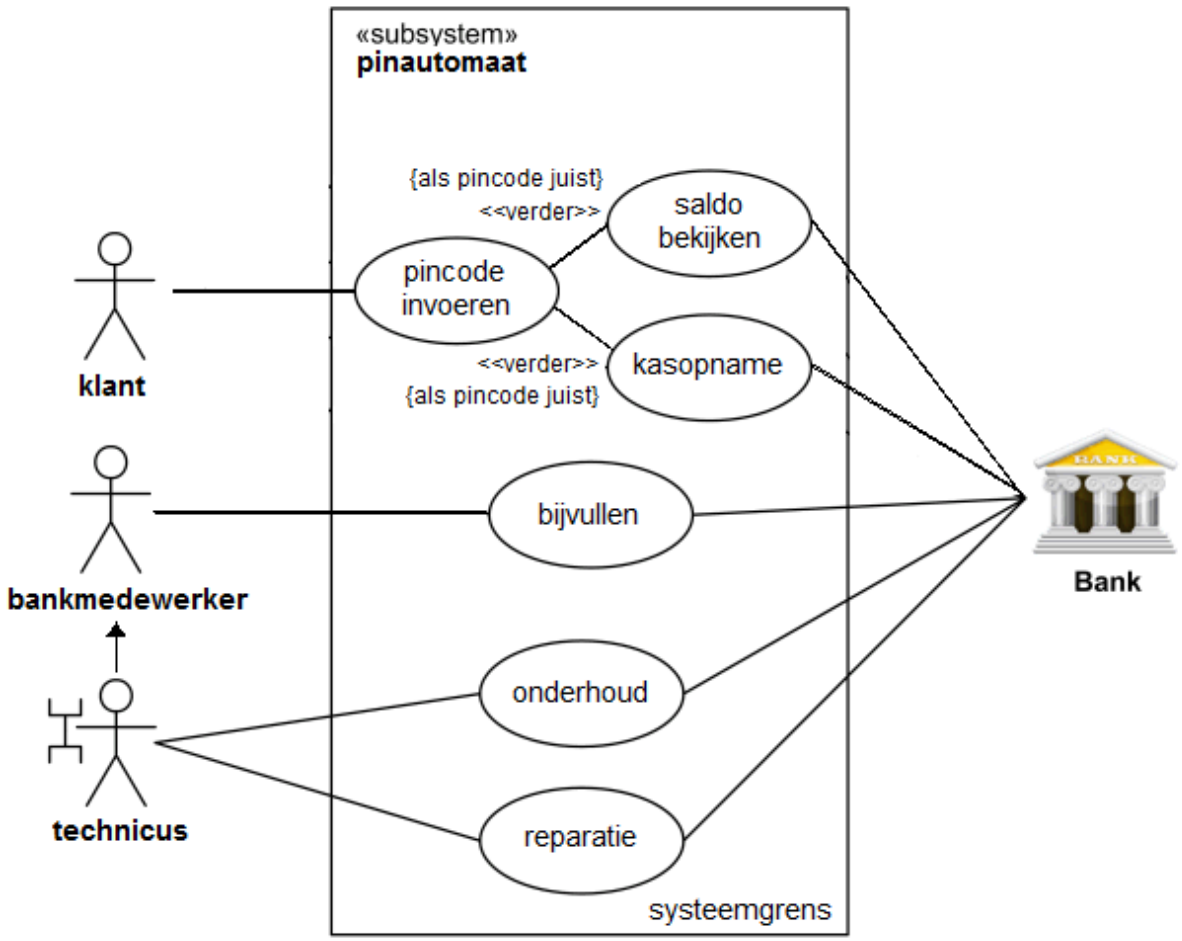
Elk use case diagram heeft een naam. Die zetten we links boven in een hokje.

Actoren worden in een diagram afgebeeld als een poppetje. De use case plaatsen we in een ellips. Als een actor een relatie heeft met een use case geven we dit aan met een lijn.

Om alle use cases van een systeem zetten we een lijn. Deze lijn geeft de **systeemgrens** aan. Deze grens heeft als doel om aan te geven dat de use cases bij een specifiek systeem horen. Soms worden namelijk in één diagram meerdere systemen opgenomen.

Op basis van het bovenstaande use case diagram zou je je kunnen afvragen of de handeling “pincode invoeren” niet ontbreekt. Aan de ene kant kan je zeggen dat het invoeren van de pincode een onderdeel vormt van een kasopname of van het bekijken van je saldo . Aan de andere kant zou je kunnen zeggen dat het invoeren van de pincode een op zichzelf staande handeling is.

Op de volgende pagina tref je een use case diagram aan waarin het invoeren van de pincode als een aparte handeling is opgenomen.



Dit diagram is een verfijning van het vorige model. Je ziet bij twee lijnen met een pijl ("saldo bekijken" en "kasopname") de tekst <<verder>> staan. De Nederlandse term <<verder>> wordt weinig gebruikt; vaker wordt gebruik gemaakt van de termen <<extend>> en <<include>>. Hiermee kan je zeggen dat de ene use case onderdeel vormt of een **uitbreiding** is op de andere. In dit geval is "kasopname" een uitbreiding van "pincode invoeren".

Aan een uitbreiding kan ook een **voorwaarde** of **conditie** worden toegevoegd. Deze staat tussen accolades vermeld. In dit geval is de voorwaarde voor de uitbreiding "kasopname" op de use case "pincode invoeren" de voorwaarde "{als pincode juist}".

Niet alleen use cases kunnen uitbreidingen zijn van andere use cases. Ook actoren kunnen uitbreidingen zijn van andere actoren. Zo kan bijvoorbeeld een technicus in dienst zijn van de bank en bankmedewerker zijn en in die hoedanigheid ook kunnen "bijvulllen". In dat geval staat er een pijl van "technicus" naar "bankmedewerker".

OPDRACHT

We bekijken het systeem Magister, veel gebruikt voor cijferregistratie op scholen. Er zijn verschillende actoren bij dit systeem. Allereerst zijn er leerlingen. Zij willen hun cijfers kunnen zien. Mentoren willen eveneens cijfers zien (maar dan van hun leerlingen). Docenten willen graag cijfers kunnen invoeren. We hebben dus zojuist 3 actoren ontdekt en 2 use cases, namelijk:

Actoren: Leerling, Mentor, Docent
Use cases: Cijfers zien, Cijfers invoeren

Eén iemand kan ook verschillende rollen hebben. Zo kan de heer Lans docent zijn en mentor. Omdat een docent anders met het systeem omgaat dat een mentor maken we toch twee verschillende actoren.

Teken een Use Case Diagram bij dit systeem.

1.4 Use Case Templates

Een **use case template** is een beschrijving van de use case. Deze moet zo precies en volledig mogelijk zijn. Een use case template bestaat minimaal uit:

- de naam en andere unieke referentie voor de use case
- beschrijving
- lijst van betrokken actoren
- condities, voorwaarden waaraan moet worden voldaan om de use case succesvol te doorlopen
- de stappen die de gebruiker uitvoert
- uitzonderingen (optioneel), wat gebeurt er als er iets mis gaat

Hieronder staat een voorbeeld, de use case template “kasopname”

Use case	101
naam	Kasopname
beschrijving	actor voert pincode in, als deze juist is kan de gewenste kasopname worden opgegeven en worden uitgevoerd
actoren	Klant
condities	- klant moet bekend zijn in het systeem - klant moet pincode goed hebben ingevoerd - klant moet gewenste kasopname opgeven - klant moet voldoende saldo hebben voor gewenste kasopname - systeem moet kasopname kunnen uitkeren
stappen	1) klant toetst pincode in 2) systeem controleert pincode 3) klant kiest gewenste bedrag 4) systeem controleert saldo en voorraad biljetten 5) systeem keert bedrag uit
uitzonderingen	pincode is niet juist: er volgt een foutmelding saldo is ontoereikend: er volgt een foutmelding biljettenvoorraad is ontoereikend: er volgt een foutmelding

1.5 Klassendiagrammen

Je hebt nu geleerd hoe je met behulp van use case diagrammen en use case templates een model kunt maken voor een systeem.

Aan de hand van de use case diagrammen en use case templates kunnen nu de klassen in het systeem beschreven aan de hand van hun eigenschappen en methoden.

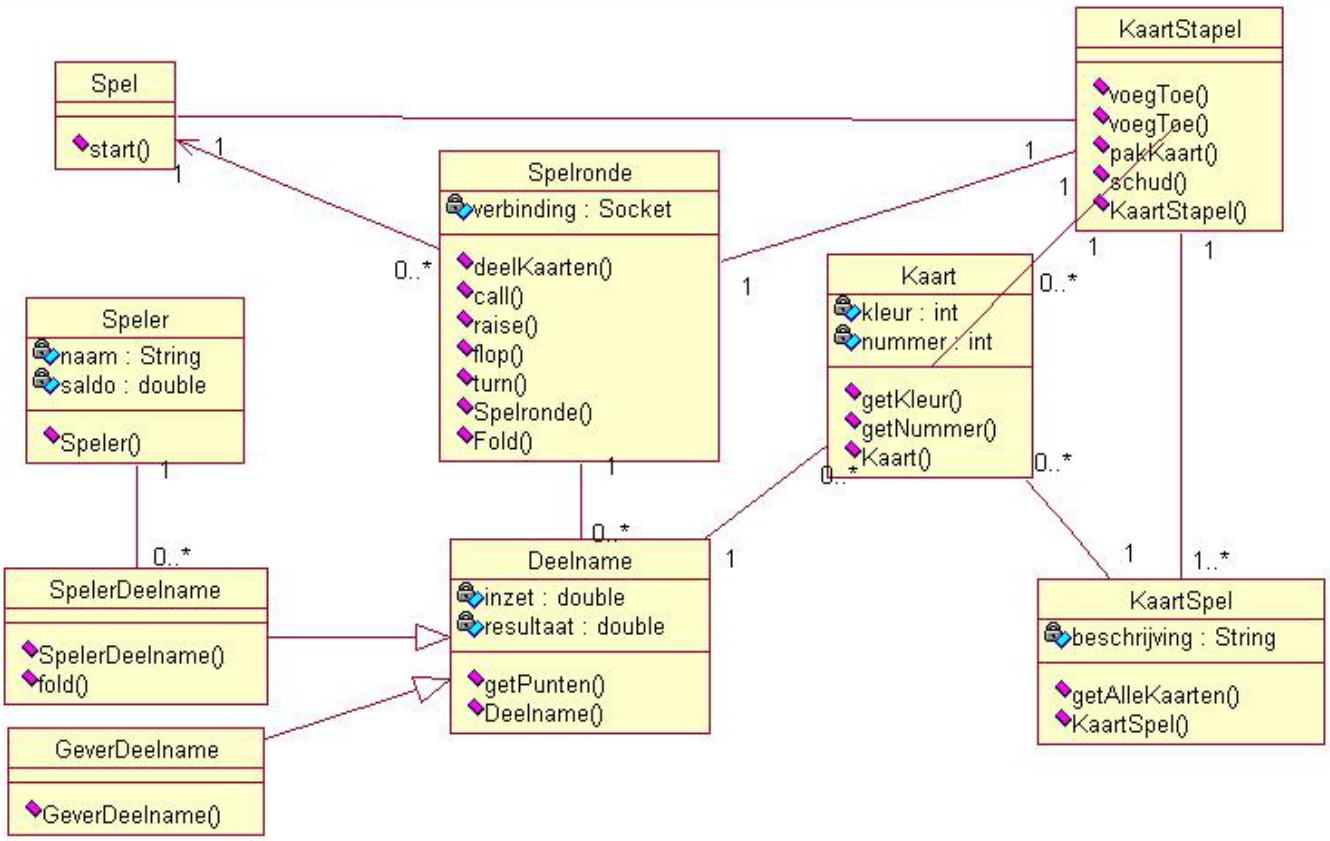
klant
+ rekeningnummer: int + pincode: int + pincodejuist: boolean + voldoendesaldo: boolean
+ ControleerPincode(): boolean + HaalRekeninggegevensop() + ControleerSaldo():boolean

In het eerste deel zetten we de naam van de **klasse**.
In het tweede deel staan de **eigenschappen** (**attributen**) opgesomd.
De notatie die daarbij gebruikt wordt is als volgt:

- de naam van de eigenschap, gevolgd door een dubbele punt
- het datatype van de eigenschap

In het derde deel staan de **methoden** (**operaties**) opgesomd.
De notatie die daarbij gebruikt wordt is als volgt:

- de naam van de methode
- tussen de haakjes staan de eventuele parameters (argumenten) die kunnen worden meegegeven aan de operatie, gevolgd door een dubbele punt
- het datatype wat de methode terug geeft.



Hierboven zie je een klassendiagram bij het kaartspel “Blackjack”. De **pijlen** in het diagram geven aan welke (abstracte) klassen eigenschappen en methoden overerven van een andere klasse. Als twee klassen verbonden zijn door een **lijn** is er sprake van een relatie tussen de twee klassen.

De getallen bij een lijn geven aan om welk type relatie het gaat, hoe vaak een klasse in die relatie kan voorkomen. Zo zien we bijvoorbeeld dat er één of meerdere kaartspellen kunnen voorkomen in één kaartstapel. We spreken van de **multipliciteit** van de relatie.

0…1	geen of maximaal één keer
1	altijd één keer
0…*	geen of meerdere keren
*	meerdere keren (dus niet geen)
1…*	een of meerdere keren

1.6 Samenvatting

Een **model** is een vereenvoudigde weergave van de realiteit. Modellen zorgen ervoor dat de beschreven realiteit snel en overzichtelijk kan worden overzien.

UML (**Unified Modeling Language**) is een modelleertaal die wordt gebruikt in de systeemontwikkeling. Bij deze modelleertaal wordt de **interactie** tussen **actor** en **systeem** beschreven in **use cases**. Deze use cases worden grafisch weergegeven in een **use case diagram** en tekstueel beschreven in **use case templates**.

Aan de hand van use case diagrammen en use case templates kunnen de klassen in het systeem worden beschreven met **klassendiagrammen**.