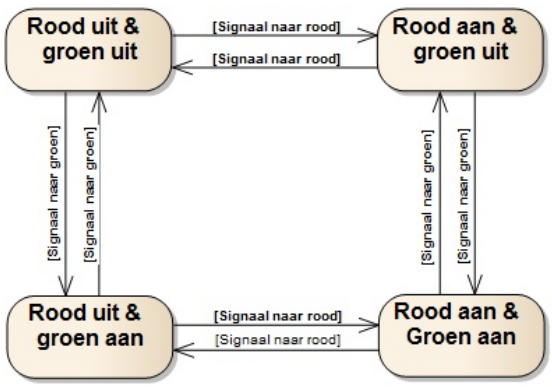
Use Case 3.0

Mensen onderscheiden zich van de meeste andere diersoorten doordat ze gereedschappen zijn gaan gebruiken om hen te ondersteunen bij het verrichten van specifieke taken. In de loop van de geschiedenis zijn die gereedschappen steeds geavanceerder geworden. We zijn machines gaan maken die complete taken van ons overnemen. Software systemen zijn een bijzonder soort machines. Als we een software systeem gebruiken bij het uitvoeren van een taak, leggen we een aantal verantwoordelijkheden bij het systeem neer. We laten het systeem acteren in onze wereld. Toestanden van onze wereld krijgen betekenis voor het systeem en toestanden van het systeem krijgen betekenis in onze wereld. Je kunt daardoor het systeem niet meer helemaal los zien van de wereld waarin het acteert. Een fundamenteel probleem daarbij is dat het systeem zich houdt aan logische wetten, terwijl de wereld waarin het moet acteren zich niet aan formele regels lijkt te houden.

door Reinoud de Leve

Langs het kanaal bij mij achter is de weg voor een gedeelte opgebroken. Bij de wegwerkzaamheden zijn twee verkeerslichten[1] geplaatst. De doorgang is te smal om twee auto's elkaar te laten passeren. De verkeerslichten moeten er voor zorgen dat het verkeer uit beide richtingen om beurten wordt doorgelaten. Daarmee acteren de verkeerslichten in onze wereld. Zij hebben een verantwoordelijkheid gekregen bij het regelen van het verkeer.
Het zijn eenvoudige verkeerslichten met alleen een groen en rood licht. Er is een systeem dat de lampen aanstuurt. Als het systeem een signaal naar de rode lamp stuurt, gaat deze branden. Stuurt het systeem opnieuw een signaal naar de rode lamp dan gaat deze weer uit. De groene lamp werkt op precies dezelfde wijze. De rode lamp kent twee statussen: Aan en Uit. De groene lamp kent ook twee statussen. Samen kennen ze vier statussen.

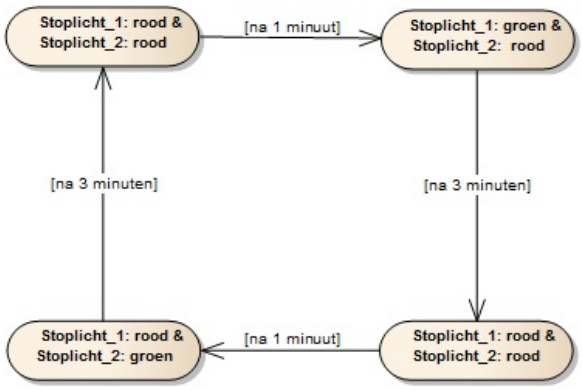


Alle statussen zijn voor het systeem even plausibel, maar binnen onze wereld heeft de status "Rood aan & Groen aan" geen betekenis. Een automobilist weet als beide lampen branden niet goed of hij moet stoppen of mag doorrijden. De betekenis van de lampen is iets buiten het systeem. Het is iets van onze wereld, maar het legt wel beperkingen op aan de werking van het systeem. Wij zullen bijvoorbeeld eisen dat het systeem de lampen steeds in combinatie aanstuurt, zodat als de rode lamp aan gaat, de groene lamp uit gaat en omgekeerd.

> Use Cases nemen een wonderlijke positie in tussen het beschrijven van wat je met een systeem wilt bereiken en hoe je dat met het systeem wilt realiseren.

Het systeem stuurt niet één maar twee verkeerslichten aan. Het moet er voor zorgen dat steeds maar één van beide rijrichtingen een groen licht heeft. Daarnaast moet het verkeer steeds de tijd krijgen om de rijbaan vrij te maken voordat het verkeerslicht voor het verkeer in tegengestelde richting op groen wordt gezet. We spreken af dat de rijrichtingen om beurten 3 minuten lang een groen licht krijgen en dat het verkeer steeds 1 minuut de tijd krijgt om de rijbaan vrij te maken. Schematisch ziet

het programma voor het systeem dat de verkeerslichten aanstuurt er dan als volgt uit.



Tot zover heeft het systeem nog maar weinig interactie met het verkeer dat het aanstuurt. Het systeem opereert voornamelijk in zijn eigen wereldje. Een beetje autistisch volgt het zijn eigen regeltjes. Als het volgens die regels moet, laat het auto's voor een rood licht wachten, terwijl er helemaal geen verkeer uit de tegengestelde richting komt. Het systeem regelt wel het verkeer, maar neemt daar nog nauwelijks verantwoordelijkheid voor.
We zullen daarom het systeem uitbreiden met sensoren, zodat het systeem kan reageren op het verkeer.
Voortaan geeft een naderende auto een signaal aan het systeem dat hij de wegafzetting wil passeren. Als de rijbaan vrij is, geeft het systeem een groen licht aan de naderende auto. Als de rijbaan in gebruik is door verkeer uit de tegengestelde richting, wacht het systeem met het geven van het groene licht totdat de rijbaan weer vrij is.
We zouden het gedrag van het systeem kunnen beschrijven met een use case. Use cases zijn het middel om de interactie tussen een systeem en zijn actoren te beschrijven. Onze use case zou bijvoorbeeld kunnen heten: "Passeren van een wegafzetting". Dat is immers het doel van de naderende automobilist. De use case begint als de auto de wegafzetting nadert en eindigt als de auto de wegafzetting heeft gepasseerd. De eerste stap zal iets kunnen zijn als:

> De auto meldt dat hij de wegafzetting wil passeren.

Dat gebeurt - weten we - met een sensor. Dat hoeven we nog niet te vermelden. De use case is immers oplossingsonafhankelijk. De laatste stap wordt waarschijnlijk iets als:

> De auto meldt dat hij de wegafzetting heeft gepasseerd.

Het is echter maar de vraag of deze laatste stap voor het systeem wel nodig is. Het systeem kan namelijk helemaal niet vaststellen of de auto uit de laatste stap wel dezelfde is als de auto uit de eerste. Het enige wat het systeem zou kunnen vaststellen, is dat het aantal auto's in beide stappen gelijk is. Als dat zo is, zou het systeem kunnen concluderen dat de rijbaan vrij is. Op

het eerste gezicht lijkt dat een handige oplossing. In de praktijk blijkt het echter helemaal geen goede strategie te zijn. Er zijn vissers die halverwege de wegafzetting hun auto aan de kant zetten om te gaan vissen. Ook wordt er een enkele keer een auto met pech opgehaald door een oplegger van de ANWB. Het tellen van het aantal auto's, leidt er daardoor toe dat het licht heel vaak onnodig op rood blijft staan. Uiteindelijk werkt het nog het beste als het systeem elke auto een minuut de tijd geeft om de wegafzetting te passeren. Net als het systeem eerst ook deed toen het nog geen sensoren had.
Wat geldt voor de laatste stap, geldt ook voor alle tussenliggende stappen. We kunnen heel goed een aantal stappen verzinnen waarmee het systeem de verkeerslichten zou kunnen bedienen. In de meeste gevallen zijn deze stappen voor het systeem helemaal niet de meest efficiënte manier om het probleem aan te pakken. Er zijn voor het probleem van de verkeerslichten verschillende scenario's te verzinnen die allemaal hetzelfde resultaat hebben. De meeste verkeerslichten die ik ken, staan standaard op rood en springen op groen als een auto nadert. Het kan ook andersom. Het licht zou ook standaard op groen kunnen staan en op rood springen als er vanaf de tegengestelde richting verkeer nadert. Misschien is dat technisch wel een eenvoudigere oplossing. Ik verwacht het niet, maar het zou kunnen.
Use cases leunen heel zwaar op scenario's: een reeks volgordelijke stappen. Use cases nemen daardoor een

wonderlijke positie in tussen het beschrijven van wat je met een systeem wilt bereiken en hoe je dat met het systeem wilt realiseren. De volgordelijkheid van de stappen is meer een onderdeel van de oplossing dan van het probleem zelf. Bijna iedereen die met use cases werkt, is weleens in de val gelopen dat hij de scenario's veel te gedetailleerd beschreef. Dat is geen zwakte van de analist, maar een zwakte van de methode. De methode verleidt ons daartoe. Je beschrijft het systeem alsof het er al is. Je probeert de interactie tussen de actoren en het systeem zo kloppend mogelijk te beschrijven. Daarvoor moet je haast wel premature aannames doen over de werking van het systeem. Veel analisten richten zich zo sterk op het beschrijven van de interactie tussen de actor(en) en het systeem, dat ze lijken te vergeten dat het systeem meer doet dan communiceren met actoren. Het systeem heeft een verantwoordelijkheid bij het uitvoeren van een taak. Daaruit vloeien activiteiten voort die niet allemaal direct zichtbaar hoeven te zijn in de interactie tussen het systeem en zijn actoren.
Je kunt mensen opnieuw opvoeden, nogmaals vertellen hoe je wel en hoe je niet use cases moet schrijven en hopen dat ze de fouten niet meer zullen maken, of je kunt hen een alternatief bieden. Essential use cases is zo'n alternatief. Essential use cases bouwen voort op het oorspronkelijke idee achter use cases. Er is dan ook een grote verwantschap tussen beide benaderingen. Het belangrijkste verschil tussen traditionele use cases en essential use cases is, dat traditionele use cases uitgaan van "stimulus response" en essential use cases van "responsibility". In de essential use case schrijven we niet op hoe de actor communiceert met het systeem over het uitvoeren van een specifieke taak, maar wat de verantwoordelijkheden (responsibilities) zijn van de actor(en) en het systeem bij het uitvoeren van de taak. Rebecca Wirfs-Brock voerde een template in voor het schrijven van use cases met twee kolommen: één voor de gebruiker en één voor het systeem. Deze template is minder populair dan de template waarin we de acties van de gebruiker en het systeem als elkaar opvolgende stappen onder elkaar zetten in een genummerde lijst, maar leent zich bijzonder goed voor essential use cases. Bij essential use cases plaatsen we de verantwoordelijkheden van de gebruiker in de kolom van de actor en de verantwoordelijkheden van het systeem in de kolom van het systeem. We hoeven daarbij nog niet over volgordelijkheid na te denken.

UC01: Passeren van een wegafzetting	
Actor	Systeem
Kenbaar maken dat hij de wegafzetting wil passeren.	Doorlaten als er geen verkeer op de rijbaan is uit tegengestelde richting.
	Tegenhouden als er wel verkeer op de rijbaan is uit tegengestelde richting.

Een essential use case past beter in een Agile werkwijze dan een traditionele use case. Hij beschrijft een taak die de actoren met het systeem willen realiseren. De taak heeft een concreet doel dat waarde heeft voor de actoren. De essential use case beschrijft welke verantwoordelijkheden de stakeholders bij het systeem willen neerleggen en welke bij de actor(en) en bevat geen premature keuzes over hoe het systeem die verantwoordelijkheden gaat realiseren. Daarmee is de essential use case een uitstekend startpunt voor de refinement sessies. De ontwikkelaars zullen in ons voorbeeld van de Product Owner willen weten hoe het systeem kan vaststellen of er verkeer uit de tegengestelde richting komt. De Product Owner moet deze requirement nu zo verwoorden dat die gebruik maakt van verschijnselen uit zijn eigen wereld die het systeem ook kan waarnemen.
Sommige mensen zullen tegenwerpen dat er in onze wereld nu eenmaal dingen zijn die in een bepaalde volgorde moeten gebeuren. Dat is misschien in een enkel geval waar. In de meeste gevallen ervaren we volgordelijkheid als noodzakelijk, omdat we het nooit anders hebben gedaan. We moeten ook niet vergeten dat systemen andere mogelijkheden en beperkingen hebben dan mensen. De volgordelijkheid die voor ons noodzakelijk is, is voor het systeem mogelijk volledig overbodig. Het systeem zou bijvoorbeeld dingen parallel kunnen doen, die voor ons onmogelijk zijn. Bovendien, als volgordelijkheid echt noodzakelijk is, dan vertaalt dat zich naar een verantwoordelijkheid. Het is bijvoorbeeld de verantwoordelijkheid van een systeem om het saldo pas uit te betalen nadat het verzoek daartoe is goedgekeurd.
Essential use cases is geen nieuw idee. Constantine en Lockwood schreven erover in hun boek "Software for use". Het is jammer dat het een beetje in de zijlijn is gebleven. Het verdient beslist meer aandacht. Misschien is het iets voor Use Case 3.0.

1) Michael Jackson: The real world

Reinoud de Leve

Reinoud de Leve is senior requirements engineer bij Atos. Hij heeft al meer dan 15 jaar ervaring met onder andere het gebruik van use cases. Hij werkte voor een groot aantal bedrijven met name binnen de financiele sector. Reinoud is redacteur van het DREAMagazine.