De zin en onzin van een het lagen model

De Law of Demeter in de praktijk

Erik Philippus
Improvemen**T**
erik@agile-architecting.com

Abstractie ligt aan de basis van architectuur. Kenmerkend voor abstraheren is het verbergen van bepaalde details van een systeem door middel van encapsulatie, waardoor we in staat zijn om gewenste eigenschappen van een systeem te identificeren, te specificeren en te borgen. Abstraheren is daarnaast een noodzakelijk proces om vanuit individuele gevallen tot algemeen geldende principes te komen. Een van oudsher veel toegepast principe is de gelaagde architectuur of *multi-tier architecture*, waarbij systeem gedrag wordt verdeeld over verschillende niveaus van abstractie. Deze gelaagdheid is een structurerend mechanisme op logisch niveau, maar is niet zelden tevens een representatie van (een deel van) de fysieke infrastructuur. Een bekend voorbeeld hiervan is de z.g. *Hardware Abstraction Layer* (HAL), bedoeld om verschillen in hardware af te schermen van de software. De meeste *operating systems* hebben een HAL ingebouwd, zodat dezelfde *kernel code* kan worden gebruikt voor verschillende hardware configuraties. Maar ook het software systeem zelf is vaak gepartitioneerd in diverse lagen, als het goed is ieder met een specifieke functionaliteit en eigen verantwoordelijkheid.

Deze opdeling in verschillende lagen is eigenlijk een veralgemenisering van het streven naar *loose coupling* en *high cohesion*, bedoeld om software modules op een hoger niveau van inzetbaarheid en uitbreidbaarheid te brengen. De heuristiek die hierbij wordt toegepast, is de z.g. *Law of Demeter*, vernoemd naar de griekse godin van de landbouw. Kort samengevat zegt deze regel: "praat alleen met je directe buren". De naamgeving van deze regel heeft waarschijnlijk te maken met het gegeven dat Demeter ook de godin was van de burgelijke orde en wetten, waarvan de invoering door de landbouw werd bewerkstelligd, toen de mensen hun zwervende leefwijze opgaven en zich gingen groeperen in vaste woonplaatsen. In ieder geval is de achterliggende gedachte dat een software module minimale kennis dient te bezitten omtrent de structuur of eigenschappen van andere elementen. Toepassing van de *Law of Demeter* in een gelaagde architectuur betekent dat software in een bepaalde laag alleen toegang heeft tot de daaronder gelegen laag. Om de doelstelling van hogere adaptiviteit en onderhoudbaarheid waar te maken, dient "*layer skipping*" zoveel mogelijk worden voorkomen.

Tot zover de theorie. Want in de praktijk is het niet al goud wat blinkt. In sommige gevallen dwingt de strikte toepassing van dit architectuur principe tot implementatie van een aanzienlijk aantal "*wrappers*", soms ook omschreven als *Demeter Transmogrifiers.* Dit zijn in feite 'doorgeefluiken' voor het uitwisselen van boodschappen tussen software modules uit niet naast elkaar gelegen lagen. Dit maakt een systeem niet alleen moeilijker te doorgronden, maar kan ook een desastreus effect hebben op de *performance*. Ik ben zelf eens een keten van tientallen doorgeefluiken tegengekomen bij de analyse van een ondermaats presterend software systeem. En als dan ook nog blijkt dat de boodschappen ergens onderweg in zo'n keten (ongedocumenteerde) modificaties ondergaan, is de misère compleet.

Een te strak opgelegd lagen model kan leiden tot blinde vlekken of zelfs hokjesgeest bij ontwikkelaars, met als gevolg een gebrek aan focus op de werkelijk belangrijke, overkoepelende eigenschappen van het systeem. Een fundamentele tekortkoming van het lagen model is namelijk dat het op gespannen voet staat met de z.g. *cross-cutting concerns*: functies die orthogonaal staan op de verantwoordelijkheden van de afzonderlijke lagen, zoals b.v. logging en beveiliging. Juist deze generieke handelingen kunnen de op papier zo fraaie opdeling in lagen behoorlijk compromiteren. Naast diverse *design patterns*, is *Aspect Oriented Development* aangedragen als oplossing voor dit probleem, maar deze techniek lijkt nog steeds niet volwassen en heeft ook weer zo zijn nadelen.

Er bestaat helaas geen simpele, universeel toepasbare oplossing voor dit heikele probleem. Vaak zijn de *cross-cutting concerns* gerelateerd aan kwaliteitseisen, die per definitie op meta-niveau door de architectuur dienen te worden geaddresseerd. En voor de systemische functionaliteit die op geen enkele manier past bij de algemene decompositie, zal in een zo vroeg mogelijk stadium een goed gedocumenteerde en gecontroleerde uitzondering op de regel gemaakt moeten worden. Demeter zal zich echt niet omdraaien in haar graf als er in een paar gevallen op een weloverwogen manier wordt afgeweken van de mooie lagen architectuur.

Erik Philippus
oktober 2009

Verschenen in de architectencolumn in Bits & Chips, 11e jaargang nr 14/15