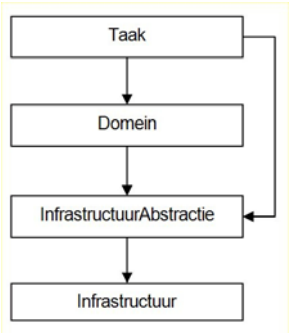
Gelaagde architecturen

Deel 2: Ontwerpen van een Logisch Lagenmodel

Bij het ontwerpen van een goed lagenmodel moet een software architect vele ontwerpkeuzen maken. Medewerkers van de Hogeschool Utrecht hebben een heuristiek opgesteld, waarbij deze ontwerpkeuzen expliciet gemaakt zijn en in stappen apart worden beschreven en geïllustreerd. En wel zo dat het lagenmodel enerzijds tegemoet komt aan de kwaliteitsdoelen van de klant en anderzijds duidelijk en eenduidig te gebruiken is voor de ontwikkelaars. Dit artikel beschrijft de stappen om tot een logisch lagenmodel te komen. Daarbij wordt gebruik gemaakt van het 'Logica in Lagen' referentiemodel dat in het vorige deel in deze serie is beschreven.

Ondanks dat veel vakliteratuur wel iets zegt over lagenmodellen, wordt maar zelden besproken hoe je als architect tot een lagenmodel kan komen dat past bij de kwaliteitseisen die aan het systeem zijn gesteld. Vaak wordt een specifiek model als oplossing aangedragen en uitgelegd. Buschmann (1), Larman (4) en de Microsoft Application Architecture Guide (5) vormen de uitzonderingen en beschrijven wel stappen en/of ontwerpoverwegingen. Niet compleet, maar beslist interessant. De methode die in dit en het volgende artikel wordt gepresenteerd, bundelt de kennis die in deze en andere bronnen te vinden is, verwijst daarnaar en vult die aan.
Dit artikel beschrijft een aanpak voor het opstellen van een logisch lagenmodel. Hierbij staat de vraag centraal: Hoe moet de totale systeemfunctionaliteit in lagen worden opgedeeld om aan de gestelde kwaliteitseisen te voldoen? Kenmerkend is het denken vanuit de functionaliteit en de kwaliteitseisen en om door middel van abstractie tot een logische ordening komen. Dit in tegenstelling tot een fysiek lagenmodel, waarbij vooral wordt gedacht vanuit de mogelijkheden van de techniek. Het ontwerpen van het fysieke lagenmodel komt later in deze serie aan de orde.



Figuur 1: Een logisch lagenmodel. De Taak-laag bevat de presentatie en taakspecifieke logica en de Domein-laag de domeingenerieke logica conform het 'Logica in Lagen' referentiemodel.

Heuristiek bij het ontwerpen

Het ontwerpproces om tot een logisch lagenmodel te komen is opgesplitst in vijf stappen, die zich met verschillende ontwerpvragen bezighouden. De stappen worden hier sequentieel beschreven, maar in de praktijk kunnen ze ook naast elkaar of iteratief worden uitgevoerd.

Stap 1: Bepaal de kwaliteitsdoelen

Deze eerste stap houdt zich bezig met de vraag waarom we een gelaagde architectuur nodig hebben. Het antwoord op deze vraag zou moeten bestaan uit een aantal kwaliteitsdoelen, die met een gelaagde architectuur gerealiseerd kunnen worden. Traditionele analyse en ontwerp richten zich vooral op de functionele eisen en om die te realiseren is een lagenmodel niet beslist nodig. Software architectuur richt zich juist op het identificeren van de niet-functionele eisen en het vinden van oplossingen voor deze (kwaliteits)eisen. Een lagenmodel kan zo'n oplossing zijn.

Als het goed is, worden in de eerste fasen van het project met het inventariseren van de requirements ook de niet-functionele eisen afgestemd met de opdrachtgever en de gebruikersorganisatie. Zie hiervoor bijvoorbeeld de Inception fase van OpenUP (6) en het extended ISO-model voor softwarekwaliteit (8). Bepaal vervolgens welke van deze eisen goed met een gelaagde architectuur bereikt kunnen worden. Kandidaten zijn onder andere eisen die te vertalen zijn naar de volgende (ISO 9126) kwaliteitsattributen: Onderhoudbaarheid, analyseerbaarheid, uitbreidbaarheid, herbruikbaarheid, juistheid en portabiliteit, vervangbaarheid.

Stap 2: Onderken logische lagen

Deze stap houdt zich bezig met de vraag welke functionaliteiten moeten worden gescheiden in aparte lagen en waarom. Het 'Logica In Lagen' referentiemodel, dat in het vorige artikel is gepresenteerd, kan goed als hulpmiddel worden gebruikt bij deze stap. Het onderscheidt verschillende soorten logica, definieert die en geeft voorbeelden van concrete functionaliteiten. De typen functionaliteit kunnen op vele manieren tot lagen worden gegroepeerd . In de voorbeelden in dit artikel worden voor de eenvoud gehele lagen uit het Logica In Lagen(LIL) -referentiemodel (7) gerefereerd, maar een gedetailleerder onderscheid kan in de praktijk nodig zijn.

Bepaal welke lagen nodig zijn om de kwaliteitsdoelen te realiseren

Het uitgangspunt voor deze stap is de set kwaliteitsdoelen, geselecteerd in de vorige stap. Bepaal nu per kwaliteitsdoel of een scheiding in lagen bijdraagt aan het realiseren van dit doel. En welke functionaliteiten dan precies gescheiden moeten worden.

In tabel 1 staan voorbeelden van (naar de ISO 9126 vertaalde) kwaliteitsdoelen en hoe die kunnen worden gerealiseerd door functionaliteiten over verschillende lagen te verdelen. De functionaliteiten zijn conform de termen van het LIL-referentiemodel beschreven.

Orden de lagen en definieer ze

Bepaal op basis van de vorige stap welke lagen er moeten komen en wat hun hiërarchisch niveau is. Bepaal per laag ook precies wat wel en niet tot de inhoud behoort.

Neem daarbij de volgende ontwerpregels in acht:

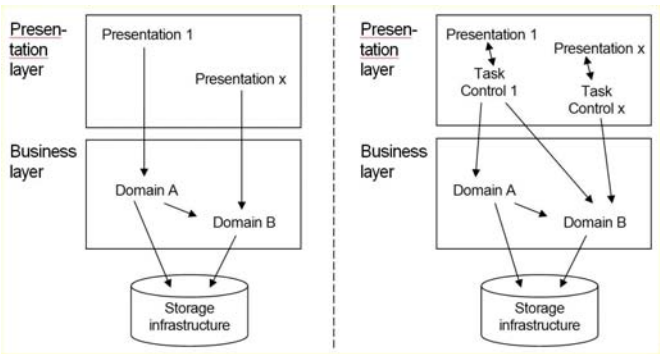
- Hoe minder lagen, hoe beter! Want lagen bieden ook nadelen.
- Ontwerpers en ontwikkelaars moeten iedere extra laag snappen en implementeren, wat tot extra ontwikkeltijd kan leiden. Verder schrijft Larman (4) dat extra lagen bij sommige systemen performance problemen kunnen opleveren.
- Iedere laag moet in principe de standaard communicatieregels respecteren.
- Als de onderdelen in twee kandidaat-lagen zonder beperking met elkaar moeten kunnen communiceren, dan gaat het niet om twee lagen, maar om twee partities binnen een laag.
- Een lagenmodel is een hiërarchische, verticale ordening. Binnen een laag kunnen groepen functionaliteit naast elkaar worden geplaatst. Maar structuren die horizontaal naast de lagen worden geplaatst horen niet in een echt lagenmodel thuis! Vaak gaat het dan om andere deelsystemen, componenten of aspecten.
- Verschillende deelsystemen kunnen verschillende lagenmodellen krijgen.

Als voorbeeld staan in figuur 2 twee verschillende lagenmodellen getekend, die erg op elkaar lijken, maar verschillende doelen dienen. In het linker lagenmodel bevat de Presentation layer de presentatielogica en de Business layer de taakspecifieke en de domeingenrieke logica. Als die twee soorten logica in de Business layer fysiek niet worden gescheiden, dan zullen

Kwaliteitsdoel	Functionaliteiten scheiden
1. 'Herbruikbaarheid' van taakspecifieke logica realiseren. Waardoor verschillende presentatie technologieën gebruikt kunnen worden voor dezelfde use case.	Presentatie logica scheiden van Taakspecifieke logica
2. 'Onderhoudbaarheid - analyseerbaarheid en wijzigbaarheid' verbeteren. Door taakspecifieke logica niet te vermengen met presentatie logica.	
3. 'Herbruikbaarheid' van domeingenerieke logica realiseren. Waardoor nieuwe functies (use cases, kanalen) snel ontwikkeld kunnen worden.	Presentatie logica en Taakspecifieke logica scheiden van Domeingenerieke logica
4. 'Functionaliteit - juistheid' van de opgeslagen gegevens garanderen. Door domeingenerieke logica te centraliseren.	
5. 'Onderhoudbaarheid - analyseerbaarheid en wijzigbaarheid' verbeteren. Door domeingenerieke logica op één plek te concentreren en niet te vermengen met taakspecifieke functionaliteit.	
6. 'Onderhoudbaarheid - stabiliteit' verbeteren. Door onafhankelijkheid van wijzigingen in database-structuur	Presentatie logica en Taakspecifieke logica scheiden van Domeingenerieke logica
7. 'Onderhoudbaarheid – analyseerbaarheid' verbeteren. Door het ontbreken van database en andere infrastructuur gerelateerde zaken in domein-klassen.	
8. 'Beveiligbaarheid' verbeteren. Door overal gebruik te maken van dezelfde security-services uit de infrastructuur laag.	
9. 'Portabiliteit – vervangbaarheid' verbeteren. Waardoor eenvoudiger van DBMS, applicatieserver, randapparatuur, etcetera gewisseld kan worden.	Infrastructuur logica scheiden van Infrastructuur logica

Tabel 1: Voorbeelden van kwaliteitsdoelen die gerealiseerd kunnen worden door bepaalde typen logica te scheiden. De kwaliteitsdoelen tussen '' zijn uitgedrukt in ISO 9126 termen.

de kwaliteitsdoelen 1, 2 en 4 uit tabel 1 wel worden gerealiseerd, maar 3 en 5 niet.
In het rechter lagenmodel bevat de Presentation layer de presentatie en taakspecifieke logica en de Business layer alleen de domeingenerieke logica. Als de twee soorten logica in de Presentation layer fysiek niet worden gescheiden, dan worden de kwaliteitsdoelen 3, 4 en 5 uit tabel 1 wel gerealiseerd, maar 1 en 2 niet.
Figuur 2 maakt dus ook duidelijk dat lagen met alleen een naam specificeren makkelijk tot interpretatiefouten kan leiden. Definieer de inhoud van de lagen dus goed (en maak daarbij gebruik van het LIL-referentiemodel).



Figuur 2: Twee verschillende lagenmodellen met twee lagen.

Stap 3: Bepaal de communicatieregels tussen de lagen
Deze stap houdt zich bezig met de vraag welke communicatieregels moeten worden gevolgd en welke uitzonderingen zijn toegestaan. En waarom?
De regels die de basis voor het lagenmodel vormen, zoals beschreven door Buschmann (1), zijn:

1. Functie aanroepen mogen alleen van hoger gelegen lagen naar lager gelegen lagen.
   • Van beneden naar boven is dus niet toegestaan.
2. Bij communicatie tussen de lagen mag geen laag worden overgeslagen (strict layered).

Als eerste ontwerpregel bij deze stap geldt: Neem deze communicatieregels zeer serieus.
Want als de onderdelen in verschillende lagen zonder beperking met elkaar mogen communiceren, dan kunnen veel kwaliteitsdoelen niet gerealiseerd worden.

Bepaal welke gebodsregels gelden en waarom
Ga per kwaliteitsdoel na welke gebodsregels gevolgd moeten worden om het doel te realiseren. De eerste vier regels in tabel 2 komen voort uit deze stap.

Bepaal welke uitzonderingsregels gelden en waarom
Een uitzondering op de regels moet worden toegestaan als de gevolgen van een regel te nadelig uitpakt voor een ander kwaliteitsdoel. Regel 5 en 6 uit tabel 2 illustreren deze stap.
Hou wel vast aan de volgende ontwerpregel: Als een uitzondering op de standaardregels wordt toegestaan, dan moet duidelijk worden beschreven in welke situaties dit is toegestaan en wat daar de reden voor is. En welke maatregelen moeten worden genomen om de nadelen op te vangen. Zo zou bij regel 5 moeten worden gezocht om afhankelijkheden van een onderliggende laag naar de presentation layer te minimaliseren. En bij regel 6 hoe onafhankelijkheid van wijzigingen in de databasestructuur toch zo goed mogelijk kan worden bereikt. Patterns kunnen daarbij helpen. In het volgende artikel komen we daar verder op terug.

Stap 4: Toets de geschiktheid van het lagenmodel
Deze stap houdt zich bezig met de volgende vragen:

1. Kan de software wel gaan werken onder deze architectuur?
2. Worden de beoogde kwaliteitsdoelen wel echt behaald met dit lagenmodel?

De eerste vraag kan worden beantwoord door (delen van) significante requirements/use cases te selecteren en daarmee te toetsen of die kunnen werken met dit lagenmodel. Dat kan

Kwaliteitsdoel	Regel
'Herbruikbaarheid' van de functionaliteit in een lager gelegen laag.	1. Alleen aanroepen van hoger naar lager gelegen lagen (strict layered model)
'Herbruikbaarheid' van de functionaliteit in een lager gelegen laag.	2. Sla geen laag of lagen over.
	3. Verberg de implementatie
'Functionaliteit - Juistheid'	4. Sla de laag die deze functionaliteit bevat niet over (bij invoer, verwijdering of wijziging van gegevens).
• Controles worden altijd en op dezelfde manier uitgevoerd	
• Berekeningen worden op dezelfde manier uitgevoerd	
'Bruikbaarheid'	5. Berichten naar de presentation layer zijn toegestaan in de volgende situaties …
• Synchronisatie van user interfaces	
'Bruikbaarheid'	6. In de volgende situaties mag en laag worden overgeslagen (relaxed layered model): bij read-only acties, …

Tabel 2: Voorbeelden van kwaliteitsdoelen en hoe die gerealiseerd kunnen worden door een communicatieregel.

modelgedreven, in de vorm van een ontwerp. Bijvoorbeeld zoals Larman (4) dat doet door de bij de use case betrokken softwareklassen volgens het lagenmodel ieder aan een laag toe te wijzen. En vast te stellen dat dat goed past of ergens knelt. En aanvullend hierop met een sequence diagram (zie figuur 3) ontwerpen hoe de use case kan worden gerealiseerd en of de communicatieregels zonder problemen kunnen worden gevolgd.
Ook kan een werkend prototype worden gebouwd, waarbij het lagenmodel op soortgelijke wijze kan worden getoetst.

Toetsen of de beoogde kwaliteitsdoelen wel echt behaald zullen worden is lastiger, maar wel te doen. In navolging van de ATAM methode (2) moeten de in stap 1 geselecteerde kwaliteitsdoelen weer naar voren worden gehaald. Vervolgens moet per kwaliteitsdoel een scenario worden verzonnen en gesimuleerd, waarbij aan het licht moet komen of het lagenmodel (of de totale architectuur) nu wel of niet voldoet voor dit kwaliteitsdoel. Voorbeelden zijn:

- Voeg een nieuwe use case toe en ontwerp die. Wordt het vereiste niveau van herbruikbaarheid en onderhoudbaarheid bereikt?
- Wijzig bestaande functionaliteit die in meerdere use cases voorkomt. Blijven de wijzigingen tot één laag beperkt? Wordt het vereiste niveau van onderhoudbaarheid bereikt?
- Vervang de database, het operating system of bijvoorbeeld de security oplossing door een andere versie of ander product. Is de vereiste mate van portabiliteit bereikt?

Stap 5 Documenteer het lagenmodel

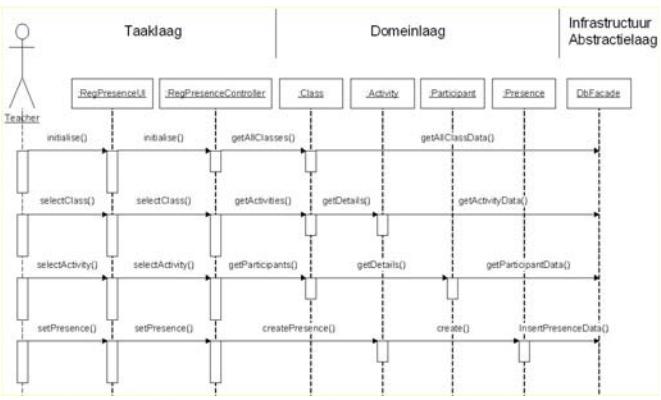
Documenteer het lagenmodel voldoende nauwkeurig, zodat het bespreekbaar en controleerbaar is en de ontwikkelaars hun ontwerpkeuzen zonder interpretatieproblemen kunnen nemen.

- Beschrijf welke lagen onderscheiden worden en geef de lagen passende namen.
- Beschrijf het hiërarchisch niveau van iedere laag.
- Beschrijf welke communicatieregels tussen de lagen gelden. Zo nodig per laag (3).
- Beschrijf waarom de regel geldt; de relatie met de kwaliteitsdoelen.
- Beschrijf in het geval van een uitzonderingsregel in welke situaties die geldig is en wat er gedaan moet worden om het ontstane nadeel op te vangen.
  Wanneer verschillende deelsystemen ieder een eigen lagenmodel krijgen, dan moeten de communicatieregels tussen de lagen van deze deelsystemen ook duidelijk beschreven worden.
- Maak een grafische weergave van de hoofdlijnen van het lagenmodel.
  Hier zijn meerdere mogelijkheden voor (3).
- Beschrijf welke kwaliteitsdoelen door deze indeling in lagen en met deze regels gerealiseerd kunnen worden. En hoe die gerealiseerd kunnen worden.

Voor een bijlage met een compleet uitgewerkt voorbeeld, parallel aan de stappen, zie: www.onderzoek.hu.nl/publicaties en zoek vervolgens op auteur Pruijt of Wiersema.

Referenties

1. *Buschmann, Frank et all - Pattern-Oriented Software Architecture: A System of Patterns, John Wiley, 1996*
2. *Clements, Paul et al - Evaluating Software Architectures - Software Engineering Institute, Addison Wesley Publishing, 2002*
3. *Clements, Paul, et al - Documenting Software Architectures - Software Engineering Institute, Addison Wesley Publishing, 2003*
4. *Larman, Craig - Applying UML and Patterns - Pearson Education, 2005*
5. *Microsoft Application Architecture Guide - Microsoft Press, 2009 - ISBN: 9780735627109*
6. *OpenUP - The Eclipse Foundation - www.eclipse.org/epf/*
7. *Pruijt, Leo - Meer inzicht in gelaagde architecturen; Deel 1: Uitleg, terminologie en methoden - Release, December 2010*
8. *Zeist, Bob van; et al - Kwaliteit van softwareproducten - Kluwer Bedrijfsinformatie, 1996*



Figuur 3: Sequence diagram bij stap 4 voor de significante use case 'Registreer Aanwezigheid/ Register Presence'. Het gerelateerde lagenmodel is in figuur 1 afgebeeld.

Leo Pruijt is docent aan de Hogeschool Utrecht en verbonden aan het Lectoraat 'Architectuur van Digitale Informatiesystemen'.

Wiebe Wiersema is lector bij het Lectoraat 'Architectuur van Digitale Informatiesystemen' aan de Hogeschool Utrecht en Vakgroepleider Solution Architectuur bij Capgemini.