

De Dienende Architect

Het gedrag van een architect in een Agile omgeving

Viktor Grgić

Architecten produceren veel waste (ook wel documenten die mensen niet lezen) en zijn vaak directief; meestal tot grote frustraties van software bouwers en het management. Daarbovenop komt een focus op eigen besluiten, architectuurdocumenten, modellen en platen. Deze leveren vaak onvoldoende of op zijn minst onduidelijke waarde. Zonder een afbreuk te doen aan modellen en documentatie die belangrijk zijn, zouden architecten zich echter vooral dienstbaar naar software engineers moeten opstellen en concrete vragen van belanghebbenden beantwoorden. En ze zouden ook het creatieve proces moeten faciliteren in plaats van alles zelf te gaan bedenken. Hierdoor is de waarde van een architect zichtbaarder en door de betrokkenheid van anderen creëert de architect zelf een groter draagvlak voor de ontstane architectuur. Dit artikel geeft weer wat de dienende architect kenmerkt, de verschillen met een klassieke architect zijn en de voor- en nadelen voor de omgeving.

Kenmerken van de dienende architect

De kenmerken zijn geworteld in de opstelling van een dienende leider wat inmiddels een bekend begrip is. 'Door anderen te dienen, dien je uiteindelijk jezelf' is het principe. Een verband met dienende leiderschap is niet alleen via het woord "dienend", maar ook leiderschap. Een ervaren architect heeft tegenwoordig sterk ontwikkelde leiderschapskwaliteiten.

De Directieve Architect	De Dienende Architect
Aandacht voor architectuur zelf	Aandacht voor belanghebbenden
Architectuur is een product geleverd door een of meer architecten	Architectuur is het resultaat na samenwerking met bouwers
Architectuurraamwerk en modelleertechniek staan centraal in communicatie	<i>Common sense</i> en eenvoud zijn het belangrijkste in communicatie
Architecten zijn bewakers en zorgen ervoor dat iedereen de uitgestippelde lijn volgt	Architectuurbewaking komt voort uit een gevoel van verantwoordelijkheid van zowel bouwers als architecten
Bepaalt alle kaders waarbinnen engineers zich mogen begeven.	Definieert en draagt één visie uit
Bedenkt en bewaakt de architectuur	Initieert, faciliteert en coördineert het proces

De dienende architect geeft aandacht aan belanghebbenden en hun belangen en zorgen

Iedere zin of een plaatje in een architectuurdocument is gemaakt om te worden begrepen door de lezer. Desnoods zou de architect speciaal voor iedere lezer een eigen "Jip en Janneke" plaatje moeten maken, zodat iedereen begrijpt. Een architectuur is geschreven op basis van vragen van belanghebbenden.

Als niemand om bepaalde informatie vraagt, dan heeft het geen nut om die te beschrijven. Het feit dat je als architect weet dat er een vraag gaat komen, is nog geen reden om die nu al te gaan beantwoorden. Het is namelijk nooit bekend wat die vraag exact is en op welke manier de belanghebbende zijn antwoord wenst te hebben. Met andere woorden, net op tijd, net genoeg besluiten en communiceren. Deze manier van omgang met architectuur wordt vaak Emerging Architecture [Gartner, 2009] genoemd.

Het nadeel van deze manier van werken is dat de architect een jacht op vragen moet openen. De belanghebbenden komen niet altijd uit zichzelf met vragen en als die er zijn, dan moet praktisch altijd doorvraging plaatsvinden. Het is een proces waarin de architect in de hoofd van de belanghebbende kruipt.

De belanghebbende heeft de voorkeur aan een face-to-face uitleg eventueel ondersteund door een whiteboard of een praatplaat. Dit reduceert niet alleen waste in de vorm van documenten, maar het geeft de belanghebbende het gevoel dat zij begrepen wordt in plaats van dat zij zich in de gedachtegang van architect moet verplaatsen.

Dit vergt wel van de architect enige vaardigheid en improvisatie. Een plaatje moet namelijk, soms op het moment dat een vraag wordt gesteld, spontaan getekend worden. Ook zegt de belanghebbende vaak dat een tot in detail uitgewerkt en goedgekeurd architectuurdocument nodig is. De schijnzekerheid van een document moet dan doorbroken worden door het inlevingsvermogen en de sluitende en volledige beantwoording van vragen. Het resultaat is een groeiend vertrouwen, waarbij de belanghebbende steeds minder de nadruk op papier legt en meer op het daadwerkelijk begrijpen hoe de aandachtspunten worden ingevuld.

Welk communicatiemiddel gebruikt wordt is volledig situatieafhankelijk. De belanghebbenden weten vaak niks van UML of Archimate af, maar zij begrijpen de samenhang soms wel beter wanneer blokjes en pijltjes gebruikt worden. Dit verschilt heel sterk per situatie en het type belanghebbende. Dezelfde stelling wordt ook gemaakt door [Greefhorst, 2009].

De dienende architect ziet ontwikkelaars en andere direct-betrokkenen als bedenkers en bewakers van de architectuur

Het is noodzakelijk dat architectuur het resultaat is van alle direct betrokkenen. [Coplien, 2010] noemt dit "All hands on deck, early on..." en legt uit hoe dit te bewerkstelligen is. Hoewel het een uitdaging is om dit creatieve proces te faciliteren - wat overigens de taak van een architect is - is de waarde immens. Er ontstaat een veel betere architectuur in kortere tijd. Het draagvlak wordt dan een vanzelfsprekendheid. Men heeft immers het gevoel zelf die architectuur te hebben bedacht. Architectuurbewaking is veel eenvoudiger en vraagt slechts ondersteuning door vragen te stellen zoals: "Hebben we nog steeds een goed gevoel over de keuzes, richting, ontwerp,...architectuur?"

Het nadeel van de betrokkenheid van zoveel mensen is dat een chaos kan ontstaan. Daarnaast resulteert het beeld van een stereotype architect die alles bedenkt in uitspraken zoals: "Waarom moet ik meedoen in dit proces? Je bent toch diegene die het moet bedenken. Lever maar het document op en dan zijn we klaar!".

Een ander nadeel is dat de omgeving soms weinig kennis en ervaring heeft met de probleemgebieden. Het is dan een kwestie van zoeken naar de grenzen; wat werkt de architect zelf uit en wat bedenkt hij/zij in gezamenlijkheid met anderen.

De dienende architect heeft een duidelijke visie en weet deze over te brengen

Een dienstverlenende opstelling waarin anderen bediend worden impliceert niet dat de architect architectuurvisie aan zijn omgeving overlaat. De omgeving heeft juist iemand met een sterke persoonlijkheid nodig die kristalhelder weet welke kant iedereen met z'n allen opgaat en dat weet over te brengen. Daarnaast is het herhalen van deze visie een bijna dagelijkse bezigheid. Deze architectuurvisie moet ook vooral eenvoudig zijn. Twee voorbeelden van hoe een visie eenvoudig kan zijn: "Wat we ook bedenken, het moet leiden tot reductie van een hoeveelheid software." en "Het systeem moet drie keer sneller, drie keer kleiner en drie keer minder fouten bevatten dan het vorige". Een minder technisch gerichte visie zou kunnen zijn: "Ontsluiting van een enkele informatiebron ten behoeve van business moet binnen een maand gerealiseerd kunnen worden." Een architectuurvisie kan worden onderbouwd door architectuurprincipes wanneer omgeving meer sturing behoeft.

Het omarmen van nieuwe inzichten gedurende een project is een groot goed. Door de dienende opstelling wordt dat nog beter gefaciliteerd. Dat geldt echter niet voor de architectuurvisie waarin alle significante besluiten¹ tot uitdrukking komen. Het veranderen van architectuurvisie veroorzaakt veel verwarring. Wanneer deze eenvoudig blijft is de kans op verandering gedurende een project zeer klein.

Een eenvoudige uitspraak als "Yes, we can!" had ogenschijnlijk door iedereen bedacht kunnen worden. De eenvoud als een antwoord op complexiteit is echter niet makkelijk. Het vraagt veel inzicht om die te bedenken en de leiderschap om die over te dragen.

Er zijn meerdere uitdagingen in deze opstelling van een architect. De architect moet over leiderschapskwaliteiten beschikken en deze moeten door zijn/ haar omgeving erkent zijn; wanneer de visie niet bij de omgeving aansluit is het effect averechts. Er kunnen misverstanden ontstaan en het gevolg is chaos. Iedereen creëert immers een eigen beeld van de visie.

De dienende architect vervult een faciliterende en coördinerende rol

Hoewel we hier over twee rollen spreken is de opstelling gelijk. In beide rollen draagt de architect zorg voor het creatieve proces, maakt mogelijk dat gebruikers en ontwikkelaars met elkaar dagelijks communiceren, beantwoordt vragen en neemt belemmeringen weg in het ontwikkelproces.

De coördinerende rol is vooral aanwezig bij grotere projecten met meerdere teams. Het bekend probleem is gebrek aan afstemming tussen de teams. Een manier is participatie van architect in alle teams waarin afstemming over de koppelvlakken door de architect wordt verzorgd. Een andere manier is het opstellen van een team waarin ieder lid het team representeert. De architect is de voorzitter.

Effect van de dienende architect op projectrisico's en beheersing

Een architectuurbeschrijving, die perfect uitgewerkt is op papier of in tekeningen, is iets anders dan de "werkelijke architectuur" inherent aan alle systemen. De meeste problemen zijn ook veroorzaakt in de implementatie van de werkelijke architectuur. Met andere woorden, een architectuur op papier pakt in praktijk altijd anders uit. Dat gezegd te hebben is het tamelijk zinloos om de bedachte architectuur te gaan testen. Het heeft veel meer zin om de werkelijke architectuur te testen. Het betekent dus ook dat engineers de aangewezen personen zijn om de architectuur tijdens de normale bouw te toetsen. Natuurlijk, gebeurt het toetsen zo vroeg mogelijk. In plaats van verdediging van de hypothetische platen, zorgt de architect ervoor dat de ervaringen uit praktijk gebruikt worden. Hiermee worden werkelijke problemen opgelost waarmee projectrisico's worden gereduceerd.

¹ Significante besluiten – Grady Booch: "Architecture represents the significant decisions that shape a system, where significant is measured by cost of change"

Het meedoen in deze feedback loop met de team is een van de belangrijkste taken van de architect. De werkelijke architectuur is vooral het resultaat van de terugkoppeling en dus een belangrijke middel bij het oplossen van technische risico's.

Een "gehoorzame" ontwikkelaar met de verwachting dat de omgeving hem/haar vertelt wat hij wel en niet mag doen past niet in dit beeld. De engineers moeten zich verantwoordelijk voelen, zelf nadenken en "klagen" wanneer iets niet duidelijk is of niet klopt.

Effect van de dienende architect op aansluiting met business en andere belanghebbenden

Het blikveld van een architect zouden we kunnen verdelen in een aantal velden: gebruikers, klanten, management, beheerders, domeindeskundigen en engineers. Deze velden zijn niet voor niets specifieke groepen mensen, omdat het mensen zijn waar een architect zich op richt. Systemen zijn belangrijk, maar mensen zijn nog belangrijker.

Het probleem dat de omgeving klaagt over IT (bijvoorbeeld ze worden niet begrepen) wordt sterk gereduceerd doordat de architect luistert, samenvat, terugkoppelt en vertaalt naar oplossingen die duidelijk in lijn staan met business behoefte. Men probeert te vaak Business & IT alignment op te lossen door gebruik te maken van tools, procedures, processen, frameworks en templates. "Begrijpen we elkaar niet goed? Maak een template zodat men niet al te creatief zelf bedenkt wat er wel en niet gecommuniceerd wordt".

Aansluiting verbetert juist door minder tools, procedures en templates te gebruiken en daarmee ontstaat meer de behoefte om echt te communiceren. Een aantal typische vormen is: interviews waarbij de architect de interviewer is, brown paper sessies, presentaties met terugkoppeling over resultaten en aansluiting met business initiatieven en strategie.

Met andere woorden, er is geen kookboek hoe de veel besproken Business/IT alignment verbeterd kan worden, behalve dan door face-to-face te communiceren.

Effect van dienende architect op doorlooptijden

Een project duurt nooit te lang doordat architect, analisten of bouwers traag zijn, maar doordat men met de verkeerde dingen bezig is. Het zijn producten met weinig tot geen toegevoegde waarde voor het totale projectresultaat ofwel werkende en in beheer genomen product.

Wanneer een architect het begrip waste begrijpt en toepast wordt de effectiviteit dramatisch verbeterd. De twee belangrijkste fronten zijn tijdsbesteding aan documenten met weinig waarde en architectuur dat door betrokkenheid van anderen beter van kwaliteit is en minder verdediging behoeft. Met name het laatste kost bijzonder veel tijd.

Waarom zijn architecten niet bij voorbaat dienend?

De eerste systemen waren technisch complex, maar organisatorisch eenvoudig. Door de verschuiving naar een hoger abstractieniveau is steeds meer met technologie mogelijk, maar is de organisatorische complexiteit ook sterk gegroeid.

Er ontstond snel de vraag naar de architectuur en architecten die de samenhang brengen en de complexiteit reduceren om de veranderingen aan te kunnen. De softwarewereld keek dan ook naar de bouwwereld waar ze meer ervaring met architectuur hadden. Er zou dan een architect zijn die een allesbepalende schets maakt hoe het gebouw eruit ziet. De schets geef je aan een bouwmeester en een aantal jaren later staat het gebouw zoals op papier getekend.

Behalve dat de bouwwereld niet met de software wereld vergeleken kan worden, blijkt dat de meest complexe, indrukwekkende, grootste gebouwen praktisch nooit door één persoon of team aan architecten worden bedacht en niet in een keer worden afgebouwd. De software wereld is wezenlijk anders dan de bouwwereld. In de software wereld is het bouw materiaal van vandaag morgen inmiddels verouderd. Men wil graag steeds betere, het meer complexe gebouw en niet nog een dezelfde ernaast. De impact bij het falen is kleiner en het

belangrijkste verschil: de kosten van verandering zijn in de software wereld substantieel kleiner. [Guest, 2007] legt de verschillen verder uit.

Een andere reden voor het ontstaan van het niet dienende verschijnsel is de behoefte van het management om één iemand ter verantwoording te kunnen roepen wanneer er fouten in de uitvoering ontstaan. De omgeving drukt vaak een architect in de positie waarin hij/ zij zelf een command & control opstelling moet hebben. Een machtige positie die in de buurt van projectleider en soms zelfs daarboven staat.

Hier komt nog eens de architect zelf als een persoon bij, wiens ego gestreeld wordt wanneer zoveel macht bij hem/haar komt te liggen. Het gevolg is ook dat de architect zijn eigen kindje gaat beschermen. Een architect is dan ook niet snel geneigd om fouten toe te geven of verbeteringen in architectuur aan te horen.

Waste

Een architect spendeert ook veel tijd aan het maken van platen, raamwerken en architectuurdOCUMENTEN. Het is immers complex genoeg om volgens TOGAF 9 raamwerk te werken en ook nog eens alles perfect in UML 2.0 en/of Archimate 1.0 te tekenen naast het maken van PSA's volgens het DYA proces. Hier hebben we het nog niet eens over kennis van best practices, design patterns, tools en helemaal natuurlijk de toegepaste cocktail aan technologieën. Als de architect ook nog de omgeving moet managen, dan is hij/ zij een superman of een superwoman.

Het probleem is dat een architect door gebrek aan inzicht zich vastgrijpt aan deze modellen en modellen leidend laat zijn in het proces; of doet slechts dat deel, dat hij/zij het beste kent. Het gezonde verstand is dan ondergeschikt aan complexe modellen en de juiste invulling ervan zou tot succes moeten leiden. Het middel is een doel geworden!

Hiermee komen we op het begrip waste zoals uitgebreid door [Liker, 2004] beschreven. Een architect vraagt zich niet af hoe iedere invulling van TOGAF bijdraagt aan business doelen en of er een makkelijkere, efficiëntere manier is. Wanneer ik niet UML, maar blokjes en pijltjes gebruik, ga ik dan mijn business doel missen? De redenen voor het nadenken over waste in het architectuurproces zijn de vertragingen in het ontwikkelproces door het maken van documenten, documentatie die niet gelezen of gevolgd wordt en perfect toegepaste modellen geen enkele garantie voor succes zijn.

Agile antwoord op de architectrol is onduidelijk

Een aantal grote namen uit de software wereld is bij elkaar gekomen en heeft het antwoord gegeven op de vraag hoe software ontwikkeling verbeterd kan worden. Dit antwoord is verwoord in het Agile Manifest [Beck, 200]. Het manifest heeft een grote invloed op de wijze waarop architectuur bedreven wordt. Een command & control architect past niet in de Agile manier van denken. Maar over de exacte omgang met architectuur verschillen de meningen. Een groep vindt dat je geen architect als rol of persoon hebt, maar dat architectuur één van de verantwoordelijkheden van het multidisciplinaire team is. [Coplien, 2010] ziet domain experts als mensen die daar in de buurt komen. Het zijn, gekserend, grijze mannen of vrouwen die "veel weten"; ofwel veel specifieke situatiekennis hebben.

De praktijk wijst uit dat deze manier van kijken naar architectuur kan werken. Een cruciale constatering is dat alle teamleden substantiële architectuurervaring hebben en gedisciplineerd regelmatig hier aandacht aan schenken.

Een website project anno 1995:

De business wenst meer bekendheid te krijgen via internet. Er wordt vastgesteld welke informatie op internet moet komen te staan en in welke structuur. De aandacht gaat vrijwel direct naar de technische oplossingen en de uitwerking ervan: Apache, HTML, HTTP, infrastructuur, internet verbinding, enzovoort.

Een website project anno 2010:

De business wenst meer bekendheid te krijgen via internet. Er komen dan eerst complexe vragen naar boven die niets met techniek te maken hebben: welke doelgroepen spreken we aan, welke samenwerking zoek je op met bestaande sites, misschien social commerce, marketing, hoeveel interactie willen we met gebruiker, beveiligingsrisico's, hoe dynamisch is de informatie, hoe staat het met beheer van content, enzovoort. De technische oplossing: "Google Sites" is misschien voldoende.

In de meeste Agile projecten gaat deze manier van omgaan met architectuur niet goed. Er is gebrek aan kennis en ervaring om samenhang te brengen en waarborgen. Aandacht van de product owner² gaat naar het bedienen van gebruikers ten koste van het totale overzicht, lange termijn en afstemming met andere belanghebbenden. En het belangrijkste is de afwezigheid van communicatieve vaardigheden; wat de belangrijkste factor is bij afstemming met omgeving over complexe architectuuraspecten.

Uiteindelijk is er ook een groep in de Agile wereld die vindt dat de architectrol wel degelijk van belang is; helemaal wanneer we het over enterprise aspecten hebben. De opstelling van een architect moet radicaal anders worden. Zijn/haar werk moet Agile principes volgen om het passend te maken. Het gaat bijvoorbeeld om werkende product, het reageren op veranderingen en meer onderlinge interactie. De toepassing van deze principes in architectuur wordt uitgebreid door [Johnston, 2009] beschreven. Ook de "agile architecting" wordt vaak als de term genoemd [Greefhorst, 2009].

Conclusie

Een architect met een dienstverlenende en faciliterende opstelling levert meer waarde, creëert grotere draagvlak voor zijn eigen ideeën en ervaart minder weerstand. Dezelfde architect vormt ook geen belemmering voor een software bouwer die een directe waarde levert. Dit gedrag past goed bij alle soorten projecten. Zijn / haar blikveld is dan gericht op de behoeften en belangen en minder op architectuurdocumenten en architectuurtools. In een dienstverlenende opstelling beantwoordt de architect de vragen, brengt de gevraagde samenhang en faciliteert het creatieve proces.

Dit artikel is tot stand gekomen na een aantal uitgebreide reviews van Xebia collega's en Rini van Solingen. Mijn dank daarvoor.

Viktor Grgić

LeanArch B.V.

viktor@leanarch.eu

Referenties

- [Greefhorst, 2009] Greefhorst, D., Rodenhuis, S., Schijvenaars, T., Oord, E., Veen van, J.: *Een pragmatische aanpak voor Enterprise Architectuur*, Via Nova Architectura, 11 mei 2009.
- [Johnston, 2009] Johnston, A.: *The Agile Architect*, 31 augustus 2009. "<http://www.agilearchitect.org>"
- [Beck, 2001] Beck, K., et al.: *Manifesto for Agile Software Development*, 2001. "<http://www.agilemanifesto.org/>"
- [Gartner, 2009] Gartner: *Gartner Identifies New Approach for Enterprise Architecture*, 11 augustus 2009. "<http://www.gartner.com/it/page.jsp?id=1124112>"
- [Guest, 2007] Guest, T: *Why Software Development isn't Like Construction*. Geraadpleegd op 14 februari 2010 van "<http://wordaligned.org/articles/why-software-development-isnt-like-construction>"
- [Coplien, 2010] Coplien, J. & Bjornvig, G.: *Lean Architecture: for Agile Software Development*. New York: Wiley, juni 2010
- [Liker, 2004] Liker, J.K. (2004). *The Toyota Way*. New York: McGraw-Hill

² Product Owner: Een persoon met bevoegdheid om backlog te prioriteren en antwoorden te geven op requirements vragen. Deze baseert hij/zij op de klantbelangen.