Network Function Virtualization

Dé missing link voor agile infrastructuurontwikkeling?

Scrum is niet meer weg te denken uit de softwareontwikkeling. Het is niet de enige, maar wel de bekendste *agile* ontwikkelmethode. Een ander voorbeeld is DevOps. Hoewel de basis voor agile ouder is, wordt de term *Scrum* al ongeveer sinds 1995 gebruikt. Sindsdien zijn veel softwareprojecten overgestapt naar een agile ontwikkelmethode. Voor infrastructuur-projecten blijven echter watervalmethoden als Prince-II dominant.

De basis van agile ontwikkeling is het continu blijven aanpassen van doelen en planning door verandering in vraag en voortschrijdend inzicht. In plaats van een aan het begin van het project vastgesteld plan, wordt er gebruikgemaakt van iteratieve ontwikkeling. Bij een op een *watervalmethode* gebaseerd project staat het **doel** vast en wordt er geschoven met **tijd**, **kosten**, en - helaas maar al te vaak - **kwaliteit** om dat doel te bereiken. Bij een *agile methode* staan **kwaliteit**, **tijd** en **kosten** vast, maar het **doel** is variabel.

Een project zonder vastgesteld doel: dat klinkt als een *recipe for disaster*

Succesvoller met agile

Voor een doorgewinterde Prince-II projectleider klinkt dat als een *recipe for disaster*: hoe kun je een project succesvol afronden als je niet weet wat je doel is? Toch geven veel organisaties die zijn overgestapt op bijvoorbeeld Scrum aan dat hun projecten succesvoller zijn en de kwaliteit van hun producten hoger. Hoe komt dat? Bij een watervalgebaseerd project is het doel in feite óók een *moving target*. De methode is er echter op gebaseerd om dit te negeren. Natuurlijk zijn *change requests* mogelijk, maar dit is vaak voor zowel klant als project niet wenselijk. Aanpassing van de *deliverables* van het project betekenen immers aanpassingen in planning en budget.

Tijdens een project leren wat nodig is

Het is vaak een onmogelijke opgave om aan het begin van een project al precies te weten wat er aan het einde van het project gewenst is. Hoe langer het project duurt, hoe lastiger het wordt en hoe groter de kans dat eisen en wensen – als ze aan het begin al correct waren geïdentificeerd – veranderen. Vaak zelfs onder invloed van het project. Tijdens de ontwikkeling of het testen - of erger: na implementatie - komen er zaken naar boven waar eigenlijk nooit goed over is nagedacht.

In een agile ontwikkeltraject is het dan ook van belang om niet teveel *van te voren* te plannen en alles duidelijk te willen hebben, maar juist *tijdens* het project te leren wat nodig is.

Focus op kwaliteit en niet op tijd

Omdat niet aan het begin van het project geprobeerd wordt álles duidelijk te krijgen is *rework* soms onvermijdelijk. Werk dat al eens gedaan is, wordt opnieuw gedaan omdat het eerste resultaat niet (meer) voldoet. Dat klinkt onwenselijk: waarom zou je opnieuw doen wat je al gedaan hebt? Agile methoden bekijken dit van de andere kant: waarom zou je iets *niet* opnieuw doen, terwijl je weet dat het niet voldoet? Opnieuw dus de focus op kwaliteit en niet op tijd.

Natuurlijk zit hier een risico aan vast: het kan *altijd* beter. Gelukkig is hier in de ontwikkelmethode rekening mee gehouden. In Scrum werken de ontwikkelaars in principe altijd aan datgene wat het meeste oplevert voor de klant. Het voor de derde keer herschrijven van een stuk code om het nog nét iets beter te doen leidt zelden tot veel toegevoegde waarde. Het staat dan ook niet hoog op de prioriteitenlijst (*backlog* in Scrum).

Agile in infrastructuurontwikkeling

In infrastructuurprojecten zien we Scrum vooralsnog minder dan in softwareontwikkelings-trajecten. Een belangrijke mijlpaal in een infrastructuurproject is vaak de hardwareselectie. Daar gaat een traject aan vooraf waarin eisen en wensen worden geïnventariseerd en vastgelegd. Dat moet ook wel, want de hardware is vaak een flink deel van het budget van een infrastructuur-project. Een corerouter of -switch kan met gemak enkele tonnen kosten. Een accesswitch van 500 euro kost ook een half miljoen als je er 1000 van nodig hebt.

Kortom: een verkeerde hardwareselectie kan een gigantische strop voor een project tot gevolg hebben.

En daar wringt ‘m de schoen: in agile ontwikkelingen wil je je niet te vroeg aan een ontwerp committeren. In infrastructuurprojecten is dat vaak gewoon nodig. Het kan niet zo zijn dat je er in sprint 7 achter komt dat de 2 miljoen euro die je in sprint 3 hebt uitgegeven weggegooid geld is, omdat de switch die je hebt aangeschaft een *feature* mist die je écht nodig hebt. Dan zit er dus niks anders op dan toch eerst het design klaar te hebben voordat je tot hardware-aanschaf overgaat. Zelfs als je de hardware alleen maar voor het lab aanschaft kan dit een grote kostenpost zijn. Zoals gezegd, een corerouter of -switch kan zo enkele tonnen kosten.

Verschil met softwareontwikkeling

Hoe anders is dat in software-ontwikkelingstrajecten. Waar gebruik wordt gemaakt van virtuele servers die - mits op de juiste manier geautomatiseerd - in seconden aangemaakt én weer vernietigd kunnen worden. Het hele lab draait vaak op een handvol relatief goedkope x86-servers die ook nog voor vervolgtrajecten gebruikt kunnen worden. Bovendien geeft dit de mogelijkheid om gescheiden ontwikkelstraten voor Ontwikkeling, Test en Acceptatie (OTA) te maken. Iets waar infrastructuurprojecten vaak alleen maar van kunnen dromen.

Network Function Virtualization

Een vrij recente ontwikkeling in infrastructuren is *Network Function Virtualization (NFV).* NFV is nauw verwant aan *Software Defined Networking (SDN).* Bij SDN is het idee om alle intelligentie uit het netwerk te halen en te verplaatsen naar centrale controllers. Bij NFV gaat het meer over *network services*. NFV ‘bemoeit’ zich meestal niet met packetforwarding, maar juist met de toegevoegde functies. Hierbij kun je denken aan Firewalls, WLAN-controllers, Network concentrators, BGP Route-Reflectors, Multicast Rendez-Vous Point en Orchestration tools.

Gevirtualiseerde functies

Ook in telecomnetwerken wordt steeds meer gebruikgemaakt van gevirtualiseerde functies als GGSN, PDN-GW en MME. Zonder dit soort features wordt het fysieke netwerk een relatief eenvoudig IP-netwerk. De keuze van de apparatuur wordt niet meer bepaald door de ondersteunde services, maar door zaken als het aantal poorten, poortsnelheid en oversubscriptie.

De complexere services worden via virtuele routers toegevoegd aan het netwerk. Dit betekent ook dat het netwerk veel meer incrementeel gebouwd kan worden. Zoals bij agile applicatieontwikkeling gebruikelijk is: eerst wordt het *minimum viable product* opgeleverd. Letterlijk het ‘minimaal levensvatbare product’, maar de vertaling ‘minimaal bruikbare product’ past wellicht beter. Daarna kunnen extra features toegevoegd worden.

Infrastructuur als ‘product’

Om dit allemaal te laten werken, moeten we de infrastructuur - en dat hoeft niet beperkt te zijn tot het netwerk - meer als product zien. In de meeste bedrijven is een nieuw netwerk echt een project. Met een begin, een eind en een min of meer vaststaande scope. Eens in de zoveel tijd wordt (een deel van) de infrastructuur vernieuwd en tussen die projecten wordt er voornamelijk onderhoud gedaan.

Zien we de infrastructuur meer als een product waar je continu verbeteringen op aanbrengt? Dan kan de infrastructuur beter afgestemd worden op wat de organisatie nodig heeft én kun je sneller inspringen op businesstrends.

Agile biedt kansen

Er zijn grote kansen om met behulp van meer agile methoden als Scrum of DevOps infrastructuren te ontwikkelen. De kracht van deze ontwikkelmethoden komt pas echt tot zijn recht als we - net als bij softwareontwikkeling - eenvoudig nieuwe services in de lucht kunnen brengen. En die eenvoudig weer op kunnen ruimen als er een beter alternatief is. Bovendien kan NFV eindelijk het testlab van het netwerk op het benodigde niveau brengen.

Jeroen Roos

Jeroen Roos is principal consultant bij NiVo network architects en sinds 1998 werkzaam in de IT. Hij heeft diverse klanten in o.a. overheid en telecom geholpen met hun netwerkvraagstukken. Hij heeft voor deze klanten netwerken ontworpen, gebouwd en beheerd.