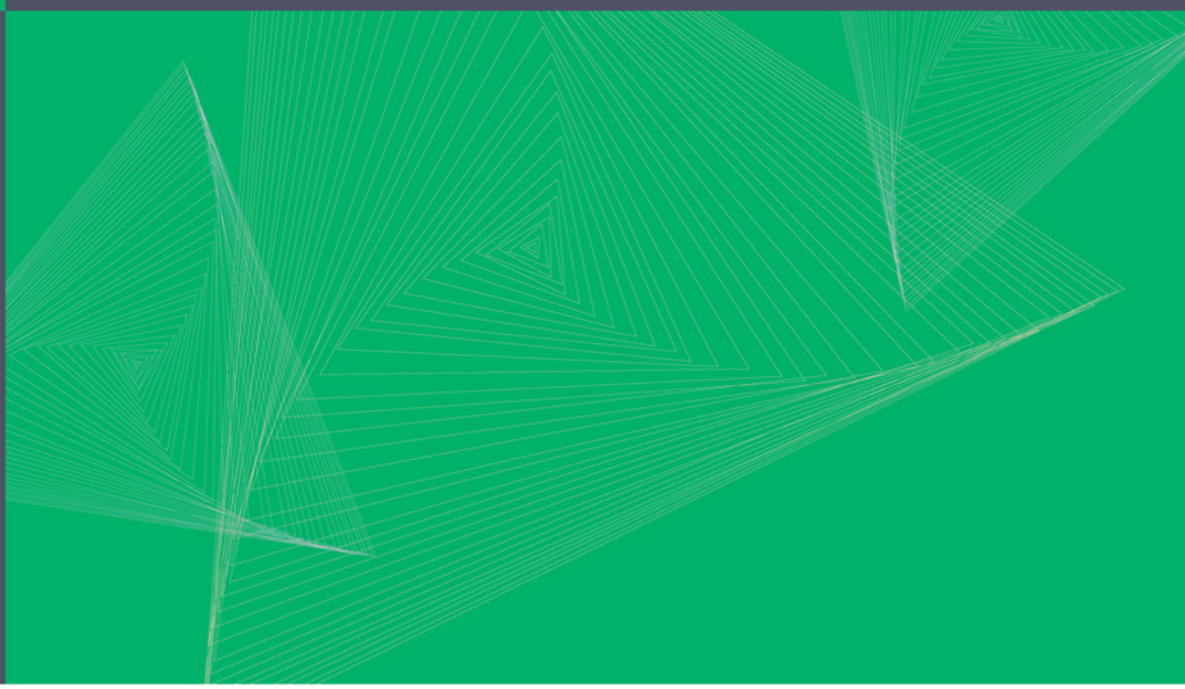
DEVOPS HANDBOEK



Oleg Skrynnik

Van Haren PUBLISHING

Andere uitgaven bij Van Haren Publishing

Van Haren Publishing (VHP) is gespecialiseerd in uitgaven over Best Practices, methodes en standaarden op het gebied van de volgende domeinen:

- IT en IT-management;
- Enterprise-architectuur;
- Projectmanagement, en
- Businessmanagement.

Deze uitgaven zijn beschikbaar in meerdere talen en maken deel uit van toonaangevende series, zoals *Best Practice, The Open Group series, Project management* en *PM series.*

Van Haren Publishing is tevens de uitgever voor toonaangevende instellingen en bedrijven, onder andere: Agile Consortium, ASL BiSL Foundation, CA, Centre Henri Tudor, Gaming Works, IACCM, IAOP, IPMA-NL, ITSqc, NAF, KNVI, PMI-NL, PON, The Open Group, The SOX Institute.

Onderwerpen per domein zijn:

IT and IT Management
ABC of ICT
ASL®
CATS CM®
CMMI®
COBIT®
e-CF
ISM
ISO/IEC 20000
ISO/IEC 27001/27002
ISPL
IT4IT®
IT-CMF™
IT Service CMM
ITIL®
MOF
MSF
SABSA
SAF
SIAM™
TRIM
VersiSM™

Enterprise Architecture
ArchiMate®
BIAN
GEA®
Novius Architectuur Methode
TOGAF®

Business Management
BABOK® Guide
BiSL® and BiSL® Next
BRMBOK™
BTF
EFQM
eSCM
FSM
IACCM
ISA-95
ISO 9000/9001
OPBOK
SixSigma
SOX
SqEME®

Project Management
A4-Projectmanagement
DSDM/Atern
ICB / NCB
ISO 21500
MINCE®
M_o_R®
MSP®
P3O®
PMBOK® Guide
Praxis®
PRINCE2®

Voor een compleet overzicht van alle uitgaven, ga naar onze website: www.vanharen.net

DevOps Handboek

Oleg Skrynnik

Van Haren PUBLISHING

Colofon

Titel:	DevOps Handboek
Auteur:	Oleg Skrynnik
Oorspronkelijke titel:	DevOps – A Business Perspective
Oorspronkelijke uitgever:	Van Haren Publishing, 's-Hertogenbosch, 2019
Vertaling:	Theo Wanders, Maarssen
Tekstredactie:	Janneke Wolters, Amsterdam
Illustraties:	Oleg Skrynnik
Uitgever:	Van Haren Publishing, 's-Hertogenbosch, www.vanharen.net
ISBN Hard copy:	9789401804363
ISBN eBook:	9789401804370
ISBN ePub:	9789401804387
Druk:	Eerste druk, maart 2019
Lay-out en DTP:	Coco Bookmedia, Amersfoort – NL

Voorwoord

Dit boek is geschreven door een IT-manager, voor IT-specialisten, IT-managers en IT-executives. Het toont DevOps niet als een fenomeen dat geassocieerd wordt met nieuwe automatiseringstools, programmeertechnieken of -technologieën, maar het verklaart de managementaspecten van DevOps voor degenen die zich professioneel bezighouden met informatietechnologie en informatiemanagement.

Het verschilt van andere boeken door het gestructureerde verhaal (misschien is het overmatig gestructureerd) en door de poging om het fenomeen DevOps volledig te dekken op zowel een basis- als een fundamenteel niveau. Dit betekent niet dat de inhoud oppervlakkig is, net voldoende om het bewustzijn over het nieuwe onderwerp te vergroten. 'Fundamenteel niveau' betekent het bouwen van de fundering, de basis: ik heb het over de oorsprong van DevOps, de onvermijdelijkheid van zijn opkomst, de belangrijkste voorwaarden ervan en hoe die in de praktijk tot uiting komen, over de praktijk zelf en de principes waarop die is gebaseerd.

Ondanks de overvloed aan literatuur over dit onderwerp, is dit het boek dat ik echt miste toen ik zelf DevOps studeerde. Ik streef ernaar om een duidelijke, gestructureerde en beknopte beschrijving van dit complexe maar zeer interessante onderwerp te geven. Ik durf te hopen dat er geen overtollige termen in dit boek staan, en dat alle noodzakelijke termen juist wel aanwezig zijn.

Ik moet mijn oprechte dank betuigen aan mijn familie en vrienden. Ik kan niet zeggen dat ze me hebben geholpen om dit boek te schrijven: gelukkig hebben ze heel weinig te maken met zaken als DevOps. Ze hebben er echter absoluut last van gehad: heel vaak, van juli tot december 2017, zonderde ik mijzelf af van de buitenwereld en reageerde ik niet op hun signalen; soms eiste ik zelfs 's avonds stilte.

Ik moet ook mijn collega's bij Cleverics bedanken. We hebben dit bedrijf opgezet samen met de slimste mensen die ik ooit heb ontmoet, en dat was toevallig een van de belangrijkste beslissingen van mijn leven. Gemeenschappelijke doelen en principes, vrijheid in besluitvorming, verantwoordelijkheid voor de resultaten, en

partners die klaarstaan om me te ondersteunen wanneer het nodig is – zonder dit zou ik geen tijd vinden om mijn denken over DevOps te structureren en het in dit boek op te nemen.

Tot slot bedank ik onze klanten: ze blijven ons nieuwe en opwindende problemen aanbieden om op te lossen; nieuwe uitdagingen. Ze blijven nieuwe trainingen, workshops en simulaties eisen; ze willen meer en beter. Ze staan letterlijk niet toe dat we stil blijven staan en stuwen ons constant vooruit.

De auteur, Moskou, herfst 2018

Over dit boek

In dit boek staat de stof voor de EXIN DevOps Foundation certificering. Dit examen test het begrip van (1) de basisconcepten van DevOps, (2) hoe deze aan elkaar gerelateerd zijn, en (3) de waarde van DevOps voor de business. EXIN DevOps Foundation is het eerste niveau van het EXIN DevOps certificeringsprogramma. De EXIN DevOps Professional certificering test de kennis van de DevOps praktijk en van hoe teams te integreren. De EXIN DevOps Master certificering gaat over het bevorderen van organisatieverandering en over het toeleiden naar continu leveren en verbeteren.

Uit het veld

'Ik dacht dat ik wel een beetje wist van DevOps, maar ik heb veel geleerd van dit boek. Het is in een verhalende stijl geschreven, wat ik leuk vind. Het is geschreven vanuit het perspectief van *legacy* werkwijzen die zich verplaatsen naar nieuwe manieren van werken en niet zozeer vanuit een ontwikkelingstechnische focus. Het raakt alle punten die ik zou raken als ik er ooit toe zou zijn gekomen om zo'n boek te schrijven. Ik zal het nu niet doen, omdat Oleg het zo goed heeft gedaan. Goed gedaan!'

Rob England

'*DevOps Handboek* is een goed geschreven en zorgvuldig samengestelde samenvatting van de belangrijkste DevOps-onderwerpen die IT-managers moeten kennen. Het combineert onpartijdigheid met scherpzinnige en bruikbare persoonlijke observaties – de auteur kent zijn stof duidelijk. Ik verwacht het van tijd tot tijd te raadplegen en ik aarzel niet om het aan te bevelen.'

Mark Smalley

'Een gemakkelijk te lezen en goed doordacht en geconstrueerd overzicht van de geschiedenis van Devops in termen van werkwijzen en de bijbehorende technologie. Het biedt een aantal goede samenvattingen, toont dilemma's waar organisaties voor staan en geeft enkele goede voorbeelden van het maken van keuzes om vooruit te komen. Het signaleert tegelijkertijd potentiële belemmeringen en valkuilen om te vermijden. Als je een DevOps 101 wilt, is dit het.'

Paul Wilkinson

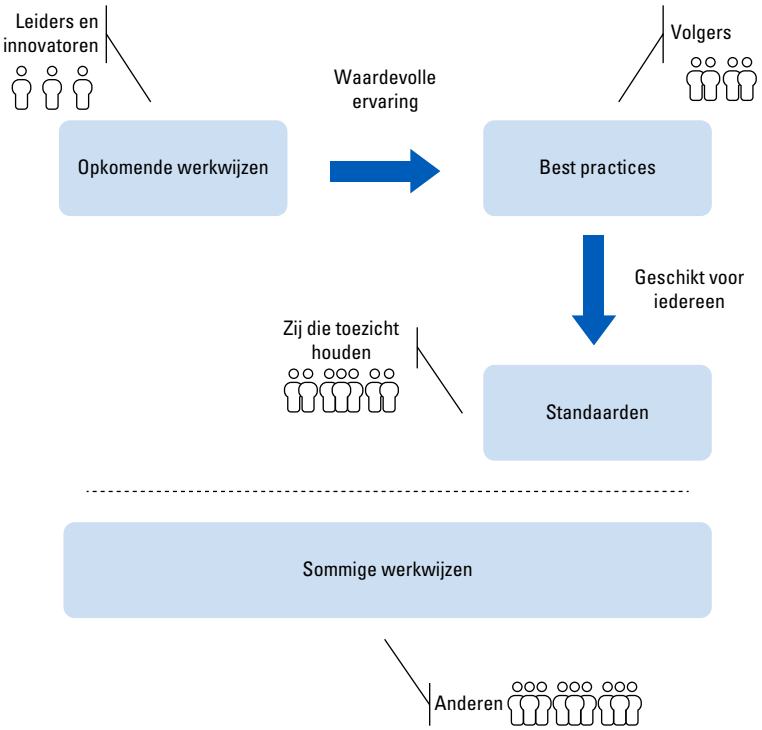
Inhoud

1 WAT IS DEVOPS?

Methoden voor IT-management staan niet stil. Het ontwikkelen en beheren van informatiesystemen pakt men tegenwoordig anders aan dan enkele decennia geleden. Bovendien zal er in de nabije toekomst alweer een volgende generatie opkomen van verfijnde methoden en technieken, die gebaseerd zullen zijn op nieuwe kennis, ervaring en technologie. Meestal evolueren managementmethoden geleidelijk, door middel van het systematiseren en aanscherpen van de modellen die eerder zijn gemaakt, op basis van bepaalde basisprincipes en behoeften. Van tijd tot tijd treden er echter discontinuïteiten op, waardoor individuele leidende organisaties een belangrijke stap voorwaarts kunnen maken met betrekking tot effectief en efficiënt gebruik van informatietechnologie.

Een goed voorbeeld is de overgang van de focus op IT-systemen naar het managen van IT-services door IT-management. Dit wordt IT-servicemanagement (ITSM) genoemd. Deze verandering in de visie van het management van rond het jaar 2000 stelde pioniers in staat belangrijke concurrentievoordelen te behalen. Opkomende *management practices* werden overgenomen door leiders, en veranderden in zogenoemde *best practices*. Enkele van de best practices evolueerden verder naar algemeen aanvaarde *good practices* en droegen zelfs bij tot industriestandaarden. Niet alle organisaties gebruikten best practices of normen in hun werk omdat niet alle sectoren van de economie in die tijd in belangrijke mate afhankelijk waren van informatietechnologie. Onder 'practices' (praktijkhandelingen of werkwijzen) verstaan we *activiteiten uitgevoerd volgens bepaalde principes om een gewenste uitkomst te produceren*.

Laten we eens kijken naar IT-servicemanagement. In de jaren tachtig ontstond het idee om waarde te genereren uit informatietechnologie in de vorm van services (diensten) en om IT-activiteiten in de vorm van processen te organiseren. Bepaalde Europese bedrijven werden pioniers, ontwikkelden nieuwe werkwijzen voor het organiseren van werk en nieuwe benaderingen voor het oplossen van managementproblemen. Enkele van de practices, zoals de introductie van een servicedesk, onderscheid tussen incidenten en problemen, beheerde en gecontroleerde



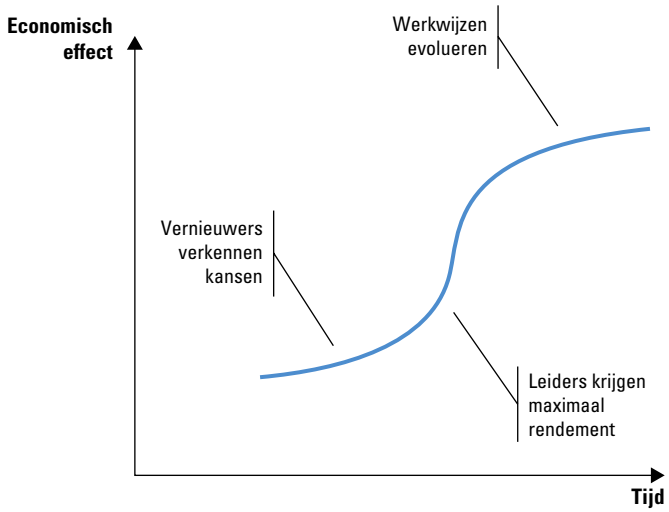
Figuur 1.1 Opkomst en gebruik van nieuwe practices

wijzigingen in de IT-infrastructuur, enzovoort, werden geformuleerd in 2000-2001 in belangrijke publicaties zoals ITIL® (dat staat voor *IT-infrastructure library*)[1]. Leidende organisaties en vele 'volgers' gebruikten deze publicaties, en daardoor werden ze best practices. Uiteindelijk werd in het jaar 2002 de eerste standaard voor IT-servicemanagement gepubliceerd: BS 15000-1: 2002, die een bepaalde norm vastlegde voor degenen die een samenhangend IT-servicemanagementsysteem wilden opbouwen. Inmiddels is BS 15000-1 vervangen door ISO/IEC 20.000-1 en ISO/IEC 20.000-2. Daaruit zal duidelijk zijn dat practices, publicaties en standaarden niet stoppen met hun ontwikkeling, maar zich verder ontwikkelen.

Soortgelijke dynamiek kan nu worden waargenomen in Agile softwareontwikkeling. De revolutie die hier plaatsvindt, beïnvloedt echter een groter gebied dan alleen softwareontwikkeling en de omvang van de gevolgen kan even groot zijn als die van ITSM.

Nieuwe, opkomende practices worden geëtiketteerd als 'DevOps' (Development + Operations), wat net zover verwijderd is van de beoogde betekenis als die van ITIL® ver verwijderd is van het 'bibliotheekconcept'. Evenzo is de huidige COBIT

1 https://www.axelos.com/best-practice-solutions/itil



Figuur 1.2 Ontwikkeling van practices

ver verwijderd van controledoelstellingen (Control Objectives for Information and Related Technology)[2].

> Tijdens het publiceren van COBIT 5 in 2012 wees de auteursrechthouder erop dat, hoewel COBIT oorspronkelijk een afkorting was van Control Objectives for Information and Related Technology, het nu juist een naam is.
>
> AXELOS Limited heeft sinds 2013 soortgelijke opmerkingen gemaakt over ITIL®.
>
> DevOps-experts, die de initiatiefnemers van deze beweging waren, erkennen de beperkte aard van de naam, en hebben nauwkeuriger namen geopperd. De kans om de naam te wijzigen is nu echter onwaarschijnlijk.

Het DevOps-fenomeen is dus het bestuderen waard. Om de essentie van DevOps volledig te begrijpen, is het noodzakelijk om de achtergrond van zowel het idee als de bijbehorende beweging te beschouwen.

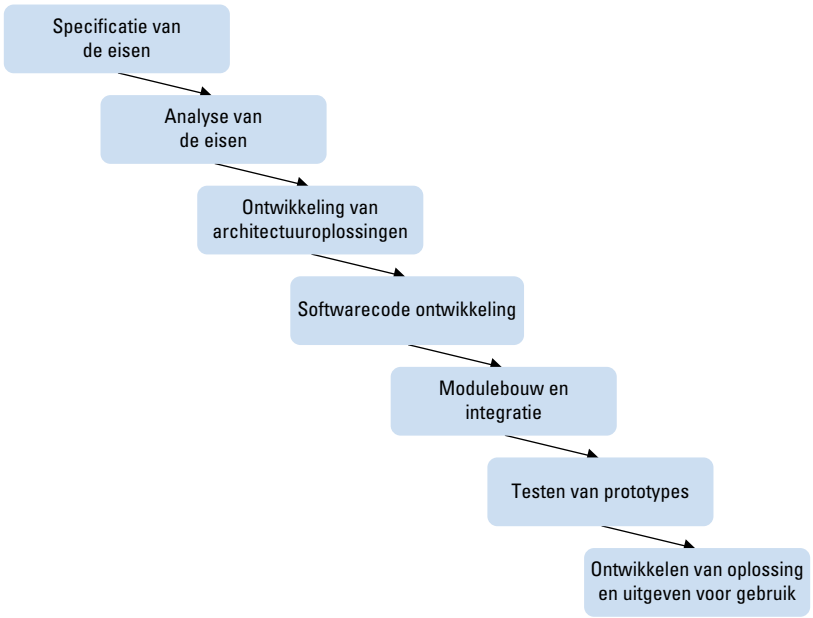
1.1 DE OORSPRONG

Je zou kunnen stellen dat DevOps verscheen vanwege twee factoren: brede acceptatie van Agile softwareontwikkelmethoden en van beheer van IT-infrastructuur als code. Uitleg over IT-infrastructuur als code volgt in paragraaf 1.1.2. Laten we eerst kijken naar Agile softwareontwikkelmethoden.

2 http://www.isaca.org/COBIT/Pages/FAQs.aspx

1.1.1 Agile methoden voor softwareontwikkeling

Aan het einde van de twintigste eeuw was de dominante methode van softwareontwikkeling het zogenoemde 'watervalmodel': sequentiële uitvoering van vooraf bepaalde fasen, die elk een aanzienlijke tijd in beslag nemen en eindigen met het behalen van eerder overeengekomen resultaten; overgang naar de volgende fase gebeurt in veel gevallen pas nadat de vorige fase volledig en formeel is voltooid. Zie figuur 1.3. Een bijkomend onderscheidend kenmerk van dit model is de functionele specialisatie van de betrokken personen in elke fase: analisten, architecten, ontwikkelaars, testers, enzovoort.

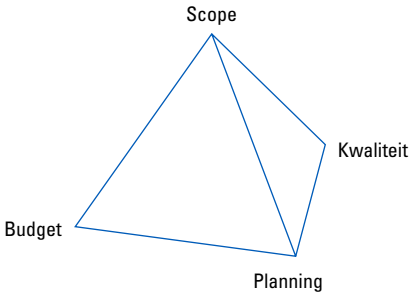


Figuur 1.3 Een voorbeeld van een waterval softwareontwikkelingsmodel

Voor het ontwikkelen van grote informatiesystemen met vooraf gedefinieerde functionaliteit, zonder dat snelle levering vereist werd, in een weinig veranderende omgeving, was het een bruikbaar model dat een effectieve en gedetailleerde kostenbeheersing mogelijk maakte.

Eind jaren negentig, met de snelle groei van internettechnologieën en webprogrammering, begonnen de negatieve kanten van het watervalmodel de interactie en het begrip te beïnvloeden tussen klanten van informatiesystemen (interne of externe bedrijven) en providers (interne of externe softwareontwikkelaars). Marktkansen voor zakelijke klanten vereisten immers snelle lancering (binnen enkele maanden) van nieuwe producten op de markt. Een typische ontwikkelingscyclus van het begin van het project tot het eerste werkende prototype zou echter zes tot achttien maanden in beslag kunnen nemen; en twee tot drie jaar in grotere ondernemingen. Met de opkomst van voorheen onbekende, maar

potentieel veelbelovende marktkansen, zouden de eisen van de klant in de loop van de ontwikkeling kunnen veranderen, wat buitengewoon moeilijk was om mee te nemen zonder de deadlines te verschuiven, het budget te overschrijden of de scope en de kwaliteit van het product te verminderen. Zie figuur 1.4.



Figuur 1.4 Klassieke piramide van de beperkingen van projectmanagement

Zo ontstond er spanning tussen klanten en providers; tussen de corebusiness en softwareontwikkelaars. Innovatieve benaderingen van programmeren waren het antwoord op deze uitdaging. Ken Schwaber publiceerde verschillende boeken over Scrum.[3] Kent Beck publiceerde een boek over eXtreme Programmering, ofwel XP.[4] Het effect van de toepassing van deze nieuwe ideeën was echter bescheiden, vooral omdat ze zich concentreerden op slechts één van de fasen van de softwareontwikkelingscyclus – de eigenlijke programmering – terwijl het probleem groter was. De end-to-end softwareontwikkelingscyclus moest worden vereenvoudigd en versneld.

In 2001 kwamen Schwaber en Beck samen met vijftien andere experts bijeen om de bestaande problemen te bespreken en een oplossing uit te werken. Het resultaat van de bijeenkomst was het zogenoemde Agile Manifesto[5]. Dit is ontworpen om de kloof tussen bedrijfs- en softwareontwikkelaars te overbruggen. Een van de auteurs van het manifest, Robert C. Martin, legt uit:[6]

'Vertrouwen tussen ontwikkelaars en bedrijven kan ontstaan en zich ontwikkelen wanneer de juiste disciplines en het juiste minimale proces worden gebruikt. Bedrijven zullen de ontwikkelaars gaan vertrouwen in plaats van te denken dat ze luie, corrupte, vervelende wezens zijn, en de ontwikkelaars zullen aandacht gaan schenken aan de business en beseffen dat het redelijke en rationele wezens zijn, in plaats van iemand van een andere planeet.'

3 Bijvoorbeeld: Schwaber, K., Agile Software Development with Scrum, 2001, ISBN 978-0130676344
4 Beck, K., Extreme Programming Explained: Embrace Change, 1999, ISBN 978-0201616415; 2nd edition, 2004, ISBN 978-0134052021
5 http://Agilemanifesto.org/iso/en/manifesto.html
6 https://www.youtube.com/watch?v=hG4LH6P8Syk, ook: https://www.aaron-gray.com/a-criticism-of-Scrum

De daaropvolgende ontwikkeling en toepassing van Agile methoden door de gemeenschap van programmeurs en projectmanagers heeft de ontwikkeling van software enorm versneld en geherstructureerd.

Agile Manifesto

Wij laten zien dat er betere manieren zijn om software te ontwikkelen door het te doen en door anderen ermee te helpen. Daarmee komen we tot de volgende waardestatements:

Mensen en hun onderlinge interactie	boven	processen en tools
Werkende software	boven	allesomvattende documentatie
Samenwerking met de klant	boven	contractonderhandelingen
Inspelen op verandering	boven	het volgen van een plan

Dat wil zeggen dat hoewel de items aan de rechterkant waardevol zijn, wij toch aan de items aan de linkerkant meer waarde hechten.

We volgen deze principes:

1. Onze hoogste prioriteit is de klant tevreden te stellen door het vroegtijdig en voortdurend opleveren van waardevolle software.
2. Verwelkom veranderende behoeften, zelfs laat in het ontwikkelproces. Agile processen benutten verandering tot concurrentievoordeel van de klant.
3. Lever frequent werkende software op. Liefst elke paar weken, ten minste elke paar maanden, met een voorkeur voor een korte tijdsperiode.
4. Mensen uit de business en ontwikkelaars moeten dagelijks samenwerken gedurende het project.
5. Bouw projecten rond gemotiveerde individuen. Geef hun de ondersteuning en omgeving die ze nodig hebben, en vertrouw erop dat ze de klus klaren.
6. De efficiëntste en effectiefste manier om informatie te delen in en met een Development Team is een face-to-facegesprek.
7. Werkende software is de primaire maatstaf voor voortgang.
8. Agile processen bevorderen constante ontwikkeling. De opdrachtgevers, ontwikkelaars en gebruikers moeten in staat zijn om een constant tempo te handhaven.
9. Voortdurende aandacht voor een hoge technische kwaliteit en voor een goed ontwerp versterken Agility.
10. Eenvoud – de kunst van het maximaliseren van werk dat niet gedaan hoeft te worden – is essentieel.
11. De beste architecturen, eisen en ontwerpen komen voort uit zelforganiserende teams.
12. Op regelmatige tijdstippen onderzoekt het team hoe het effectiever kan worden en past het vervolgens zijn gedrag daarop aan.

De belangrijkste elementen van Agile ontwikkeling zijn: nauwere interactie tussen de klant en de ontwikkelaar, vermindering van de batchgrootte, producten die met korte tussenpozen worden geleverd (cycli) en beperkte omvang van de teams.

Met behulp van een Agile benadering brengt het softwareontwikkelteam om de twee tot vier weken een nieuw levensvatbaar (deel)product uit. Eindgebruikers zijn nauw betrokken bij de ontwikkeling en zorgen voor snelle feedback, wat op zijn beurt tot snellere veranderingen leidt.

In veel bedrijven was het effect echter kleiner dan verwacht, toen men het watervalmodel losliet ten gunste van Agile ontwikkeling. Het niet profiteren van Agile in veel bedrijven heeft vaak weinig te maken met de voordelen van het watervalmodel of de nadelen van Agile. Het probleem komt voort uit het feit dat de ontwikkeling van de code slechts een van de schakels in een lange waardeketen is.

Sterker nog, voorafgaand aan de ontwikkeling moeten er nog steeds belangrijke stappen gezet worden die gericht zijn op het identificeren van zakelijke behoeften, hun uitwerking, analyse, prioritering, enzovoort. Bovendien moeten applicaties na ontwikkeling snel worden geïmplementeerd in de productieomgeving, zodat de klanten alle voordelen te zien krijgen die hun waren beloofd, en feedback kunnen geven aan de ontwikkelaars.

De IT-infrastructuur van vrijwel elke organisatie die voor 2010 is opgericht, is echter gebaseerd op rigide, dure hardware die lang geleden is aangeschaft; budgetten daarvoor werden met grote moeite verkregen en het budgetteringsproces voor nieuwe aanbestedingen is lang. Ook is de IT-infrastructuur vrij fragiel in een groot aantal organisaties. Een van de factoren die aan deze kwetsbaarheid bijdragen, is dat de gebruikte IT-oplossingen uiterst complex zijn. Er zijn vele duizenden onderling verbonden items in de infrastructuur. Een andere factor is het gebrek aan documentatie over IT-systemen en de snelle veroudering van de documentatie. Dit laatste wordt noodzakelijkerwijs voortdurend versterkt door het verlies van kennis als gevolg van het personeelsverloop.

In veel organisaties is het onveilig om de IT-infrastructuur te veranderen. Wijzigingen zijn het grootste kwaad voor de operationele activiteiten van de IT-afdeling en een constante grote stroom van wijzigingen kan tot catastrofale gevolgen leiden. Geavanceerde methoden voor softwareontwikkeling worden dus opgehouden door obstakels aan de kant van operationele IT-activiteiten, waardoor het mogelijke positieve effect van het toepassen van Agile benaderingen wordt verminderd.

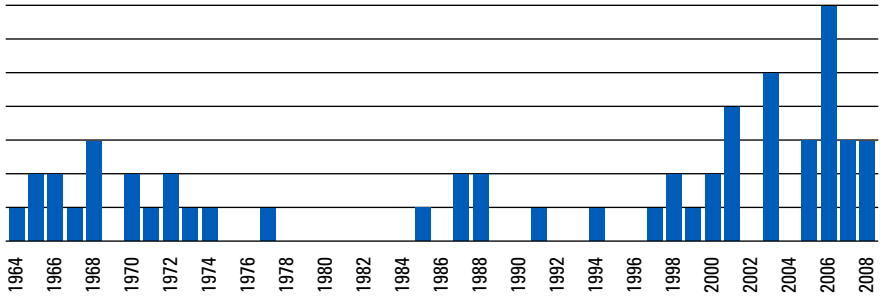
Om met fragiliteit van de IT-infrastructuur om te gaan, gebruiken sommige organisaties een geformaliseerd change management proces dat ontworpen is om de

stroom van wijzigingen te structureren en de risico's verbonden aan hun implementatie te minimaliseren.

1.1.2 Het managen van de IT-infrastructuur als code

De opkomst van het managen van de IT-infrastructuur als code werd voorafgegaan door de ontwikkeling van twee technologieën: virtualisatie en *cloud computing*.

De geschiedenis van virtualisatie van software- en hardwareomgevingen begon redelijk lang geleden, in 1964, met het begin van de ontwikkeling van het IBM CP-40-besturingssysteem[7]. Tijdens de jaren van constante ontwikkeling op dit gebied is aanzienlijke vooruitgang geboekt. De eerste in de handel verkrijgbare systemen voor mainframes verschenen in de jaren zeventig, en die voor later meer gebruikelijke machines op basis van de Intel x86-architectuur verschenen in de jaren tachtig.[8] Figuur 1.5 toont een aantal belangrijke gebeurtenissen gerelateerd aan virtualisatie tussen 1964 en 2008 (de grafiek stopt niet per ongeluk in dat jaar, zoals u later zult zien).



Figuur 1.5 Belangrijke virtualisatiegebeurtenissen verdeeld in de tijd

Virtualisatie maakte het niet alleen mogelijk om dure en krachtige hardware efficienter te gebruiken, maar ook om een extra niveau van abstractie te introduceren tussen de uitvoerbare code (de businessapplicatie) die iets nuttigs biedt voor de klant en de onderliggende systeemsoftware. Er is een belangrijke stap gezet in de richting van het scheiden van de competenties en verantwoordelijkheden van, zo te zeggen, *application engineers* en *system engineers*, in de brede zin van deze concepten.

Cloud computing technologie is nog sneller ontwikkeld. Tot het midden van de jaren negentig boden telecommunicatiebedrijven hun klanten de Wide Area Network service (WAN-service) aan door de relevante eindpunten met elkaar te verbinden: voor elke klant met directe bekabeling, de zogenoemde 'huurlijnen'.

7 https://en.wikipedia.org/wiki/IBM_CP-40
8 Het is interessant om op te merken dat, volgens Jez Humble, in die jaren gedurende een bepaalde periode IBM het aanbevelen van virtualisatieproducten aan zijn klanten vermeed, aangezien dit de verkoop van de hardware beïnvloedde.

Met de opkomst van private virtuele netwerktechnologie (VPN, Virtual Private Network) werd het echter mogelijk om datapakketten van verschillende klanten via dezelfde datatransmissiekanalen te verzenden, waarbij ook het vereiste niveau van beveiliging, privacy en servicekwaliteit werd geboden. In die tijd begonnen aanbieders het cloudsymbool te gebruiken om de grens te laten zien tussen het privé-netwerk van de klant en het gedeelde netwerk, en de respectieve scheiding van verantwoordelijkheid.

Met de nieuwe mogelijkheid om gegevens over lange afstanden over te zetten, begonnen klanten deze technologieën niet alleen te gebruiken voor de informatie-uitwisseling tussen hun externe systemen, maar ook voor het distribueren van de computerberekeningen tussen verschillende knooppunten van hun netwerken. De opkomst van een technologie om deze interactie te vereenvoudigen en in te regelen, werd ingegeven. Kleine providers namen de eerste stappen, maar echt belangrijke veranderingen vonden plaats in 2006, toen Amazon EC2 (Elastic Compute Cloud) presenteerde. Al snel, in 2008, lanceerde Microsoft zijn service, Azure, en introduceerde Google de Google App Engine, die vervolgens is geëvolueerd naar het Google Cloud Platform. Dit zijn natuurlijk niet de enige voorbeelden van het verhuren van computercapaciteit, maar het zijn de grootste.

Virtualisatie en cloudtechnologieën hebben het computerlandschap aanzienlijk veranderd. Middelen aangeboden door commerciële aanbieders zijn betaalbaar en betrouwbaar geworden; ze hebben ook gezorgd voor het nodige beveiligingsniveau. De houding van de klant ten opzichte van de cloud en het gebruik ervan is veranderd van: 'Iemand anders beheert mijn hardware ergens' naar: 'Ik heb een infrastructuur die ik op afstand kan beheren.'

Het Amerikaanse National Institute of Standards and Technology (NIST) heeft vijf essentiële kenmerken van cloud computing geïdentificeerd:[9]

1. On demand zelfbediening. Een consument kan eenzijdig en automatisch computercapaciteiten, zoals servertijd en netwerkopslag, toewijzen, zonder dat er menselijke interactie met een serviceprovider nodig is.
2. Brede netwerktoegang. Computerresources zijn beschikbaar via het netwerk en toegankelijk via standaardmechanismen die het gebruik door heterogene dunne of dikke klantenplatforms bevorderen.
3. Gedeeld gebruik. De computerresources van de provider worden samengevoegd om meerdere consumenten te bedienen met behulp van een *multi-tenant* model, waarbij verschillende fysieke en virtuele resources dynamisch worden toegewezen en opnieuw toegewezen op basis van de consumentenvraag.
4. Snelle elasticiteit. Mogelijkheden kunnen elastisch worden ingesteld en in sommige gevallen automatisch worden vrijgegeven, om snel naar buiten

9 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

en naar binnen toe af te stemmen op de vraag. Voor de consument lijken de beschikbare mogelijkheden voor IT-capaciteit vaak onbeperkt en kunnen op elk gewenst moment in elke hoeveelheid worden toegewezen.
5. Meetbare service. Cloudsystemen besturen automatisch en optimaliseren het gebruik van resources door gebruik te maken van een monitoring- en meetsysteem op een abstractieniveau dat geschikt is voor het soort dienst.

Wat betekent het op afstand beheren van de infrastructuur? Denk eens terug aan een van de belangrijkste paradigma's van UNIX-systemen: alle noodzakelijke acties met het systeem moeten toegankelijk zijn vanaf de *command line* (dus met behulp van een script). Grafische gebruikersinterfaces zijn mooi, maar optioneel.

Laten we nu de virtuele cloud technologieën en de command line interface combineren voor alle taken. Daardoor konden IT-professionals de onderdelen van de IT-infrastructuur creëren die ze nodig hadden, inclusief servers, opslagsystemen en netwerkcomponenten, en alle interfaces daartussen; alle instellingen en configuraties door middel van tekstcommando's.

De mate van de automatisering is aanzienlijk toegenomen, net als de snelheid van de implementatie van de wijziging. Voorheen moest men om een IT-infrastructuur te implementeren op basis van interne hardware:
- een begroting rechtvaardigen en overeenkomen (weken en maanden);
- wachten op de volgende aankoopcyclus (maanden);
- apparatuur bij de leverancier bestellen en ervoor betalen (dagen);
- wachten op aflevering (weken en maanden);
- ontvangen, installeren, configureren en klaarmaken voor gebruik (dagen en weken).

Tegenwoordig is het mogelijk om een vergelijkbare IT-infrastructuur te creëren door:
- een script te draaien, te wachten op de voltooiing van de uitvoering (minuten, zelden uren);
- de factuur van de cloudprovider te betalen aan het einde van de maand.

Dat wil zeggen dat de vereiste infrastructuur wordt gecreëerd met behulp van code. Ze is niet alleen gecreëerd, maar kan ook worden beheerd als een programmacode: met versiebeheer, het bijhouden van wijzigingen, debugging, het hergebruiken van vorige versies, enzovoort. Deze aspecten zullen in meer detail worden besproken in hoofdstuk 3.

Om het beeld af te sluiten, willen we ook aandacht geven aan het tweede leven dat sommige relatief oude technologieën hebben verkregen. Virtualisatie op besturingssysteemniveau was bijvoorbeeld beschikbaar in veel UNIX-systemen in de jaren tachtig. Het serieuze commerciële succes van deze technologie, die

vaak 'containerisatie' wordt genoemd, kwam echter pas in de tweede helft van de periode 2000-2010, samenvallend met de hiervoor beschreven gebeurtenissen. Hoewel het originele chrootmechanisme[10] van Unix nogal beperkt was wat betreft functionaliteit en mogelijkheden, is het nu mogelijk om een bestandssysteem voor containers te isoleren, schijfquota toe te wijzen, het beschikbare RAM-geheugen te beperken, of processortijd, I/O-bandbreedte enzovoort.

1.1.3 Het was onvermijdelijk

> 'Iemand zegt dat dit onvermijdelijk is – en wanneer je iemand dat hoort zeggen, is het zeer waarschijnlijk dat een reeks bedrijven campagne voert om het waar te maken.'
>
> Richard Stallman, oprichter van Free Software Foundation en maker van het GNU-besturingssysteem voor cloud computing, 2008.

Nu we de oorsprong van DevOps hebben onderzocht, kunnen we de volgende conclusies trekken. Ten eerste is er, door de opkomst van nieuwe manieren om interactief te zijn met zakelijke klanten en de adequate toepassing van Agile ontwikkelingstechnieken, behoefte ontstaan aan nieuwe manieren van IT-management. Ten tweede werd het binnen de IT met de opkomst van nieuwe infrastructuur-managementtechnologieën mogelijk om het IT-werk anders te organiseren.

Als we de hiervoor geciteerde woorden van Richard Stallman[11], bekijken (het lijkt erop dat hij zich vergiste in cloud computing), kan aangenomen worden dat het verschijnen van iets wat lijkt op DevOps slechts een kwestie van tijd was.

1.2 DE DEFINITIE

Alleen zeer zelfverzekerde of oneindig incompetente mensen, evenals algemeen erkende goeroes, kunnen een fenomeen serieus bespreken zonder het een definitie te geven, of zonder te vertrouwen op een algemeen geaccepteerde definitie. Helaas is de situatie verre van eenvoudig met DevOps.

Sommige experts proberen iets zelf te verzinnen, dicht bij hun begrip. Anderen beweren dat het op dit moment onmogelijk is om DevOps te bepalen, omdat het eerder een fenomeen, een beweging, een idee, maar geen discipline of een

10 Een chroot op Unix-systemen is het veranderen van de rootdirectory van het nu draaiende proces en de subprocessen naar een andere map. Een programma dat in zo'n aangepaste omgeving draait, kan bestanden buiten deze mappenstructuur niet bereiken.
11 https://www.theguardian.com/technology/2008/sep/29/cloud.computing.richard.stallman

methodologie is. Weer anderen zeggen dat iedereen zijn eigen DevOps heeft en bieden een bekende metafoor aan voor de blinde mannen die een olifant aanraken: de ene zegt dat het hoogstwaarschijnlijk een boom is, de andere dat het voelt als een kleed, de derde zegt dat het slangachtig is, enzovoort.

Toen ik dit onderwerp bestudeerde, heb ik een groot aantal boeken en online publicaties gelezen, gecommuniceerd met verschillende mensen die betrokken zijn bij de DevOps-beweging, zowel in Rusland als in Europa, en heb ik gespecialiseerde trainingen bijgewoond om verschillende internationale examens af te kunnen leggen. Naar mijn mening is het enigszins overdreven om te stellen dat het onmogelijk is om DevOps te definiëren. Natuurlijk, zoveel mensen, zoveel meningen, en in het geval van consultants loopt het nog meer uit de hand: twee consultants betekent minstens drie meningen. Echter, met een systemische mindset, een graad in IT en consultancyervaring op het gebied van IT-management, vond ik het nodig om het probleem op een duidelijke en gestructureerde manier te benaderen. Zonder te beweren dat het universeel is of de ultieme waarheid, stelde ik de volgende definitie samen:

DevOps is een evolutie van ideeën van Agile softwareontwikkeling en Lean manufacturing, toegepast op de end-to-end-waardeketen in IT, waarmee bedrijven met moderne informatietechnologieën meer kunnen bereiken als gevolg van culturele, organisatorische en technische veranderingen.

Er zijn vier belangrijke punten die in deze definitie moeten worden benadrukt.

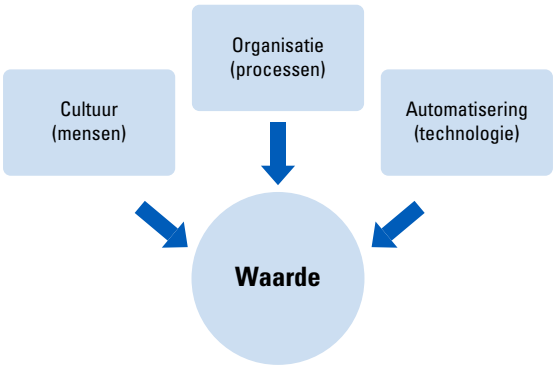
Ten eerste is het belangrijk om erop te wijzen dat DevOps Agile en Lean niet vervangt, maar ze een beetje absorbeert. Mijn communicatie met collega's, klanten en trainingsdeelnemers laat zien dat degenen die niet bekend zijn met Agile ontwikkeling, veel nieuwe en interessante dingen in DevOps ontdekken. Degenen die relevante training en ervaring hebben, zijn verrast door het aantal overlappingen tussen DevOps en andere praktijken, zoals Lean, Scrum en Kanban. Naar mijn mening is het niet helemaal correct om dit verschijnsel een 'overlap' te noemen. Het gaat veeleer om het lenen en uitbreiden van de ideeën van Agile ontwikkeling en Lean manufacturing. Deze kwestie zal in meer detail worden besproken in hoofdstuk 2.

Ten tweede, de essentie van DevOps ligt in het feit dat de IT-afdeling samen met de business niet alleen denkt aan softwareontwikkeling, maar ook aan de hele waardeketen. Deze keten begint met het genereren van nieuwe ideeën samen met belanghebbenden uit de business, en vindt plaats via ontwikkeling, testen en implementatie, tot aan het gebruik. Deze aanpak bevordert de analyse, identificatie en eliminatie van knelpunten in de end-to-end-waardeketen. Het legt feedbacklussen vast, niet alleen vanaf het einde van de keten tot het begin, maar ook

tussen de stappen en binnen elke stap. DevOps besteedt de meeste aandacht aan deze elementen: systeembenadering, werken met beperkingen en openstaan voor feedback. Dit wordt hierna in detail beschreven.

Ten derde is het belangrijk om de verwachte waarde van het gebruik van DevOps te benadrukken, die ligt in een hoger rendement op informatietechnologie. Volgens de klassieke opvatting stelt het gebruik van informatietechnologie organisaties in staat meer voordelen te behalen (door nieuwe kansen te creëren of bestaande beperkingen weg te nemen), risico's te verminderen en middelen te optimaliseren. Bij correct gebruikt adresseert DevOps alle drie de aspecten. Het zou onjuist zijn om te zeggen dat organisaties op traditionele manieren, zonder DevOps, niet kunnen profiteren van informatietechnologie. DevOps biedt echter meer waarde, wat zich uit in het versnellen van de levering van de nieuwe en gewijzigde producten aan de markt, snellere reactie op klantbehoeften, verbeterde beschikbaarheid en duurzaamheid van IT-systemen, en efficiënter gebruik van beperkte middelen. Dit onderwerp zal in meer detail worden gepresenteerd in paragraaf 1.3.

Ten vierde zijn er in het laatste deel van de definitie expliciete aanwijzingen voor drie essentiële elementen: culturele, organisatorische en technische middelen. In feite is dit de oude mantra van processen, mensen en technologie. De ervaring van de DevOps-pioniers, evenals hun volgers, toont aan dat het belang ervan nog steeds groot is.



Figuur 1.6 Drie essentiële elementen

Ik geef nu de DevOps-definitie met een uitsplitsing naar de componenten, waarmee de belangrijkste punten kunnen worden benadrukt.

DevOps is:
a. een evolutie van de ideeën van Agile softwareontwikkeling en Lean manufacturing;
b. toegepast op de end-to-end waardeketen in IT;

c. waarmee bedrijven meer kunnen bereiken met moderne informatietechnologieën;
d. als gevolg van culturele, organisatorische en technische veranderingen.

1.3 WAAROM DEVOPS?

> 'Luister, als sterren verlicht zijn, betekent dit dat er iemand is die het nodig heeft.'
>
> Vladimir Mayakovsky[12], 1914

Sommige managementframeworks komen over als een product van de verbeelding van de auteur (misschien een expert of zelfs een goeroe): een soort theoretisch onderzoek. Hun toepasbaarheid of niet-toepasbaarheid wordt bewezen door aanhangers en volgers die nieuwe technieken proberen te gebruiken in hun werk en in het managen van anderen.

Andere benaderingen worden geboren als antwoord op tamelijk dringende behoeften. Ze zijn niet gemaakt door de Britse overheid en niet door groepen speciaal gerekruteerde consultants. Deze benaderingen zijn ontwikkeld door beoefenaars die manieren zoeken om bepaalde moeilijkheden of beperkingen op te heffen, of om het gebruik van beschikbare beperkte middelen efficiënter te maken, of om nieuwe business, nieuwe niches en nieuwe tools te creëren om specifieke (en echte) problemen op te lossen.

Het lijkt erop dat DevOps dichter bij de tweede groep staat dan bij de eerste. Details over de oorsprong ervan zijn besproken in paragraaf 1.1. Laten we ons nu concentreren op de belangrijkste problemen die verschillende organisaties proberen op te lossen met behulp van DevOps.

1.3.1 Reduceer time-to-market

Bedrijven die DevOps gebruiken, melden meestal de noodzaak om de *time-to-market* aanzienlijk te verkorten. Verschillende mensen bedoelen verschillende dingen met deze term. In het algemeen wordt eronder verstaan: *de tijd vanaf het begin van een bedrijfsidee tot de realisatie ervan en daaruit volgend de mogelijkheid voor een klant om een nieuw product te kopen of om een nieuwe service te krijgen*. Dus, een berekening (of liever een beoordeling) van de time-to-market

12 Vladimir Mayakovsky, Listen! Early Poems, vertaald door Maria Enzenberger, 1991, ISBN 978-0872862555

gaat over een vrij groot tijdsbestek. In het geval dat de IT-afdeling hierbij betrokken is, bestaat dit tijdsbestek uit de volgende stappen:

- structureren en initieel en formeel opstellen van een bedrijfsidee, of liever verschillende bedrijfsideeën, en hun rechtvaardiging;
- evaluatie en selectie van een bedrijfsidee voor implementatie;
- planning van de acties die nodig zijn voor implementatie; verkrijgen van budget;
- voorbereiding van personeel en bedrijfsprocessen;
- tegelijkertijd: formalisering van de eisen, ontwikkeling van prototypen, initiële test, ontwikkeling van een volledig uitgerust IT-systeem, grondig testen, release en *deployment*;
- op hetzelfde moment: marketingactiviteiten, voorbereiding van de markt, voorbereiding van de verkoopkanalen en hulpmiddelen;
- een nieuwe lancering van een product of dienst.

Het beschreven proces introduceert bepaalde uitdagingen. Ten eerste kan het jaren duren, terwijl het bedrijf het graag tot maanden wil reduceren. De zakelijke rechtvaardiging voor urgentie is hier duidelijk: tijdens de ontwikkeling van een nieuw product kan de markt zo sterk veranderen dat het product zelf niet langer relevant zal zijn, of kunnen concurrenten eerder een vergelijkbaar product uitbrengen. Vroegtijdige toetreding tot de markt met een aantrekkelijk concurrerend aanbod helpt bij het verwerven van een dominante positie in nieuwe niches, wat op zijn beurt de marktleider de gelegenheid biedt om de markt verder te veranderen en voor zichzelf aan te passen. Dit is een essentieel voordeel dat maar weinigen hebben, maar iedereen streeft ernaar. Bovendien mogen we de steeds toenemende snelheid van verandering niet vergeten. Een van de beste illustraties van deze stelling is de Wet van de versnellende opbrengst, geformuleerd in 1999 door Ray Kurzweil.[13] Volgens die stelling heeft de snelheid van verandering in een breed scala aan evoluerende systemen, inclusief (maar niet beperkt tot) nieuwe technologieën, de neiging om exponentieel te groeien. In de praktijk betekent dit dat technologische doorbraken, inclusief informatie, vaker voorkomen. Bedrijven die het tempo van verandering verhogen, worden leiders en alleen degenen die hun snelle tempo kunnen behouden, krijgen de kans om niet aan de zijlijn te blijven. Wat kan er gezegd worden over diegenen die niet snel kunnen veranderen?

> 'Het is moeilijk om een schrijver van een filmscript te zijn. Men kan de omstandigheden die razendsnel veranderen niet bijhouden. … Eerst schrijf je het script, dan wordt het gerepeteerd, dan wordt het overhandigd aan de distributiekantoren en pas dan bereikt het het scherm. Op dat moment komt er iets nieuws en relevants voor het moment uit …

13 http://www.kurzweilai.net/the-law-of-accelerating-returns

De auteurs … van verschillende en actuele spelen bevinden zich precies in dezelfde positie. Binnenkort zullen ze moeten schrijven terwijl ze 's ochtends koffiedrinken en kranten lezen. Dus tegen het middaguur is het toneelstuk klaar voor de generale repetitie en 's avonds wordt het aan het publiek getoond. Alleen onder deze onmisbare voorwaarde kunnen ze het moment grijpen.'

Van: *The Theatre*, Moscow daily theatre newspaper[14], augustus 1917

De tweede moeilijkheid van het hiervoor beschreven proces is de behoefte aan duidelijke coördinatie en harmonisatie van onderling afhankelijke stappen, met name van de stappen die gelijktijdig worden uitgevoerd. Op dit moment vallen veel bedrijven in de klassieke valstrik: als er geen gereed eindproduct is, is er niets om te adverteren en te verkopen, maar als het er is, leiden marketingactiviteiten slechts met vertraging tot verkopen (vandaar het financiële rendement). Deze valstrik verhoogt de daadwerkelijke time-to-market verder en vereist dat ieders werk nog zorgvuldiger wordt gecoördineerd.

Merk op dat de rol van de traditionele IT-afdeling bij het vergroten van de time-to-market nauwelijks kan worden overschat. In sommige organisaties is IT verantwoordelijk voor meer dan 50% tot 70% van de totale time to market duur van anderhalf tot twee jaar.

Een ander begrip van de term time-to-market is minder globaal, maar niet minder belangrijk. Dynamische bedrijven die digitale producten maken, zijn gewend om snel te handelen. Ze geven de voorkeur aan *safe-to-fail* experimenten boven een nauwgezette en gedetailleerde planning, en het woord 'idee' wordt vervangen door 'hypothese'. In dit geval ziet het proces er als volgt uit:

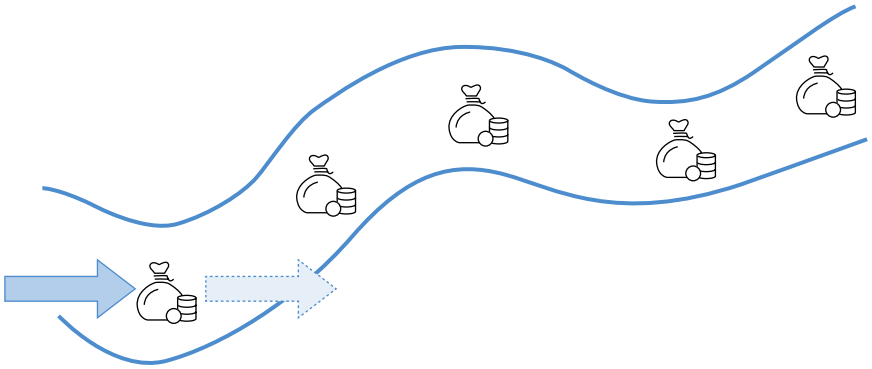
- creatie van een hypothese, ontwikkeling van de validatiemethoden;
- praktische implementatie van de hypothese;
- resultaatevaluatie, A/B-tests[15], vergelijking met doelen;
- aanpassing op basis van de analyse, terugkeren naar de eerste of tweede stap.

Het is gemakkelijk om hier een cyclus te zien; de verwachte duur is weken. Dit snelle tempo is nodig omdat de essentie van deze beweging voortdurend onderzoek is. Bij de start is de eindtoestand compleet onbekend, en dat geldt ook voor de weg

14 https://project1917.ru/groups/teatr

15 Een A/B-test is een vorm van gerandomiseerd onderzoek met testgroepen, waarbij twee (of meerdere) varianten van bijvoorbeeld een website met elkaar kunnen worden vergeleken. Gebruikers worden gerandomiseerd ingedeeld in verschillende groepen, waarbij elke groep een andere versie van de website krijgt voorgeschoteld. De controlegroep, die een ongewijzigde versie van de website te zien krijgt wordt traditioneel aangeduid met de letter 'A', en de eerste gewijzigde variant met de letter 'B'. Door het gedrag van gebruikers in de verschillende groepen te meten en met elkaar te vergelijken kan worden bepaald welke opzet, A of B, het gewenste doel – vaak de hoogste omzet – het best benadert. Bron: Wikipedia.

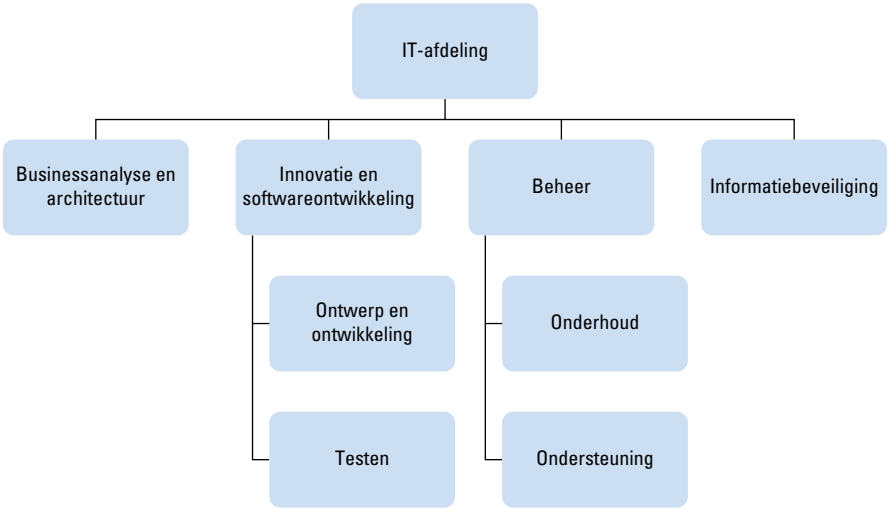
ernaartoe en voor het eindproduct. Langetermijnplanning is niet zinvol, het bedrijf ziet alleen de volgende, de naaste stap; of, om precies te zijn, het probeert die te raden. Een bekende metafoor die dit illustreert, vergelijkt de overleving en ontwikkeling van een bedrijf met onderzoek naar een geldrivier. Na één keer deze rivier te hebben bevaren en een nieuwe niche en nieuwe kansen te hebben gevonden, zal het bedrijf de veranderende rivierbedding moeten blijven verkennen. Traditionele processen, voorschriften en reeds bestaande producten zullen waarschijnlijk de traagheid van het bedrijf vergroten, en als ze onbeheerd achterblijven, zullen ze zonder geld naar de kust stromen.



Figuur 1.7 De geldrivier

Het is niet moeilijk te raden dat de bijdrage van de IT-afdeling aan het vertragen van deze cyclus groot is. De rol van IT is inderdaad van cruciaal belang bij het creëren van digitale producten, dus de vertragingen in de implementatiefase van de hypothese zijn waarschijnlijk te wijten aan het feit dat de 'langzame' IT-afdeling maanden in plaats van de verwachte weken neemt.

Om de time-to-market te reduceren, biedt DevOps een verscheidenheid aan technieken, bijvoorbeeld: het verminderen van de batchgrootte, het verminderen van het aantal overdrachten, het continu identificeren en elimineren van verliezen, en andere. Ze zullen in meer detail worden behandeld in hoofdstuk 4. Het is belangrijk om het volgende op te merken: het is naïef om te denken dat het gebruik van DevOps-technieken (om het werk van de IT-afdeling te versnellen) tegelijkertijd tot een vermindering van IT-kosten zal leiden. Integendeel, de kosten van informatietechnologie zullen toenemen, voornamelijk als gevolg van de toename van het aantal IT-medewerkers. Inderdaad, de traditionele organisatie van de IT-afdeling veronderstelt afzonderlijke functionele eenheden, die elk alle taken binnen hun vakgebied behandelen (bedrijfsanalyse, ontwikkeling en testen, beheer, ondersteuning, enzovoort). Tegelijkertijd is binnen elke functionele eenheid de noodzakelijke uitwisselbaarheid van specialisten verzekerd en het aanzienlijke aantal specialisten met dezelfde kwalificaties en competenties maakt het mogelijk de werklast gelijkmatig te verdelen.



Figuur 1.8 Functionele structuur van een traditionele IT-afdeling

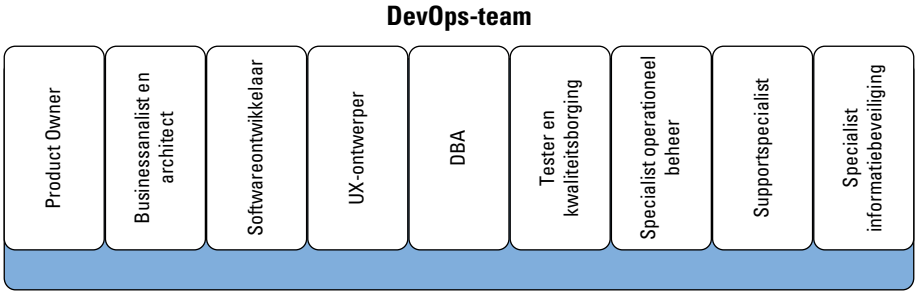
In tegenstelling tot de structuur uit figuur 1.8 groepeert DevOps specialisten in toegewijde productteams. Elk zelfstandig team bestaat uit een product owner (producteigenaar), architecten, ontwikkelaars, testers en specialisten die verantwoordelijk zijn voor de werking en voor informatiebeveiliging. Met een groot aantal teams, uitsluitend gericht op hun product, is het moeilijker om te verzekeren dat de werkdruk gelijk is; het kan leiden tot onderbenutting en dus tot hogere personeelskosten (dit onderwerp zal worden behandeld in paragraaf 4.2).

Er kan dus worden gesteld dat de traditionele IT-eenheid het 'optimaliseren voor kosten'-model volgt, terwijl de DevOps-organisatie is gericht op het 'optimaliseren voor snelheid'-model en deze doelen zijn over het algemeen tegenstrijdig. Merk ook op dat DevOps hulpmiddelen en technieken biedt om de kostenstijging te beperken, zoals de automatisering van alle routinematige handelingen of de uitwisselbaarheid van specialisten binnen de teams. Bovendien wijzen DevOps-aanhangers er terecht op dat snelheidsoptimalisatie er in veel gevallen op gericht is om het bedrijf in staat te stellen meer te verdienen, wat de stijgende kosten van IT compenseert. In dit geval wordt de IT-afdeling behandeld als een echte zakenpartner, niet als kostenplaats.

1.3.2 Reduceren van technical debt

Het concept van *technical debt* (technische schuld) werd voorgesteld door Ward Cunningham in 1992.[16] Het treedt op wanneer een programmeur een niet-optimale manier kiest om een probleem op te lossen om daarmee de ontwikkelingstijd te verkorten.

16 http://wiki.c2.com/?WardExplainsDebtMetaphor



Figuur 1.9 Voorbeeld van een DevOps-team

Technical debt is eigenlijk *uncompleted work*. Het weerspiegelt het extra ontwikkelingswerk dat nodig is wanneer programmacode die makkelijk en snel te implementeren is wordt gebruikt en er niet voor de meest optimale oplossing wordt gekozen. In een ontwikkelingsproces is er dan na verloop van tijd herstructurering van bestaande code (*refactoring*) nodig, om de programmacode beter en effectiever te maken. Als dit niet gerepareerd wordt blijft het terugkomen, bijvoorbeeld door slechte performance of instabiliteit. Het is eigenlijk een kwaliteitsgebrek in de software die op een bepaald moment in de gebruiksfase tevoorschijn zal komen en dan 'betaalt men de rekening'. De optelsom van verborgen *undone work* zal steeds groter worden als er geen reparatie plaatsvindt.

Cunningham merkte op dat dit een natuurlijk proces is, en het probleem is dat geaccumuleerde niet-optimale oplossingen leiden tot een geleidelijke verslechtering van de ontwikkelingsoutput en, als een gevolg daarvan, tot verslechtering van de productkwaliteit. In de loop van de tijd zal het ontwikkelteam meer middelen moeten inzetten om de gevolgen van eerdere beslissingen te herstellen, dat wil zeggen de bestaande code opnieuw te formuleren in plaats van nieuwe functionaliteit te ontwikkelen. De analogie met de financiële 'schuld' is in dit geval heel duidelijk: om de productie te versnellen, kan het bedrijf 'schulden' krijgen, maar het moet niet toelaten dat alle inkomsten aan schuldaflossing (*debt servicing*) worden besteed.

Martin Fowler ontwikkelde het idee van technical debt verder door een classificatie voor te stellen van de oorzaken ervan (zie figuur 1.10).[17]

	Roekeloos	Voorzichtig
Beraadslagen	We hebben geen tijd voor ontwerpen	We moeten nu opleveren en omgaan met de gevolgen
Onachtzaam	Wat is gelaagdheid?	Nu weten we hoe we het hadden moeten doen

Figuur 1.10 Technical debt kwadrant van Martin Fowler

17 https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

In zijn standpunt komt grofweg de gedachte van Ward Cunningham terug: in een goed georganiseerd ontwikkelingsteam kan het verhogen van technical debt een bewuste stap zijn om voordelen op korte termijn te behalen; het is wel belangrijk daarbij aandacht te schenken aan het 'betalen van de schuld'.

Op dit moment wordt het concept van technical debt meestal in ruimere zin toegepast. Met de uitbreiding naar operationele IT-problemen, wordt er een hele reeks traditionele problemen met de IT-afdeling aan toegevoegd: het verhelpen van storingen door apparaten opnieuw op te starten, een softwarepatch installeren die niet goed is getest, infrastructurele wijzigingen implementeren zonder zorgvuldige planning, handmatige patching of serverconfiguratie zonder de wijzigingen te documenteren – dit zijn slechts enkele voorbeelden van accumulerende technical debts, die niemand in een reguliere IT-afdeling ooit zal 'betalen'. Sommige IT-organisaties plannen dergelijke werkzaamheden of projecten helemaal niet in, anderen denken dat het in orde brengen van de zaak wel kan zodra er een minuutje vrij is. De praktijk is dat er geen minuutjes vrij bestaan op een moderne IT-afdeling.

Bovendien kan worden gesteld dat sommige best practices, bijvoorbeeld die door ITIL® worden aangeboden, ook kunnen leiden tot een toename van technical debt als ze op een geïsoleerd niveau worden toegepast. Het incidentmanagementproces volgens ITIL® heeft bijvoorbeeld niet tot doel om de oorzaken van verstoringen te vinden en te elimineren. Het doel is om het IT-systeem (of de IT-service, het verschil is hier niet relevant) zo snel mogelijk te herstellen, vaak door middel van *workarounds* en tijdelijke oplossingen. Het gebruik van dergelijke oplossingen garandeert bijna dat fouten zich opnieuw zullen manifesteren; vandaar de nieuwe IT-kosten voor hun re-eliminatie. ITIL®-auteurs gingen ervan uit dat *problem management* zou samenwerken met incidentmanagement, met als doel het identificeren en elimineren van de grondoorzaken van incidenten: in feite het verminderen van technical debt in de breedste zin van het woord. Samenwerken voorkomt isolatie en dus technical debt. We merken echter op dat, hoewel in de meeste moderne IT-afdelingen op zijn minst enige praktijk van incidentbeheer bestaat, het uiterst lastig is om een effectief problem management proces 'in het wild' te traceren.

DevOps besteedt veel aandacht aan het verminderen van technical debt, of beter het managen ervan. Twee van de meest gebruikte methoden zijn de volgende. Ten eerste maakt constante refactoring van de programmacode het mogelijk om rekening te houden met de ervaring die is opgedaan in het werkveld (het gebruik), en het werken aan het elimineren van eerder gecreëerde (bewuste of onbedoelde) knelpunten wordt gepland op dezelfde manier als het creëren van nieuwe functionaliteit. Ten tweede beveelt DevOps sterk aan om de praktijk van 'zo veel mogelijk issues onder ogen zien' te gebruiken om 'stagnatie' te voorkomen van problemen die iedereen kent, maar waarvoor niemand de handen uit de mouwen wil steken.