Inhoud

Voorwoord

Wat is Agile ontwikkeling precies? Hoe is die ontstaan? Hoe heeft die zich verder ontwikkeld?

In dit boek geeft Uncle Bob weldoordachte antwoorden op al deze vragen. Hij beschrijft tevens op hoeveel manieren Agile verkeerd geïnterpreteerd en verbasterd is. Zijn visie is relevant, want hij is een autoriteit op dit gebied, iemand die aanwezig was bij de geboorte van Agile.

Bob en ik zijn al jaren bevriend. We ontmoetten elkaar voor het eerst toen ik in 1979 bij de telecommunicatiedivisie van Teradyne kwam. Als elektrotechnisch ingenieur hielp ik bij de installatie en ondersteuning van producten; later werd ik hardwareontwerper.

Ongeveer een jaar nadat ik bij het bedrijf gekomen was, begon het op zoek te gaan naar nieuwe productideeën. In 1981 stelden Bob en ik een elektronische telefoniste voor, in wezen een voicemailsysteem met doorschakelfuncties. Het concept viel in goede aarde en we begonnen al snel met de ontwikkeling van 'E.R. – The Electronic Receptionist'. Ons prototype was state-of-the-art. Het draaide het MP/M-besturingssysteem op een Intel 8086-processor. Spraakberichten werden opgeslagen op een Seagate ST-506 harde schijf van 5 MB. Ik ontwierp de hardware voor de spraakpoort terwijl Bob begon met het schrijven van de applicatie. Toen ik klaar was met mijn ontwerp, schreef ik ook applicatiecode en sindsdien ben ik ontwikkelaar.

Rond 1985 of 1986 stopte Teradyne abrupt met de ontwikkeling van E.R. en trok, zonder dat wij dat wisten, de octrooiaanvraag in. Het was een zakelijke

beslissing waar het bedrijf snel spijt van zou krijgen, en een die Bob en mij nog steeds achtervolgt.

Uiteindelijk verlieten we beiden Teradyne voor andere kansen. Bob begon een adviesbureau in de omgeving van Chicago. Ik werd softwareconsultant en instructeur. We bleven contact houden, ook al ben ik naar een andere staat verhuisd.

Tegen het jaar 2000 gaf ik les in Object Oriented Analysis and Design bij Learning Tree International. De cursus omvatte UML en het Unified Software Development Process (USDP). Ik was goed thuis in deze technologieën, maar niet in Scrum, Extreme Programming en vergelijkbare methoden.

In februari 2001 werd het Agile Manifesto gepubliceerd. Zoals met veel ontwikkelaars was mijn eerste reactie: "Het Agile wat?" Het enige manifest dat ik kende was dat van Karl Marx, een fervent communist. Was dit Agile-ding een oproep tot de strijd? Die verdomde softwareradicalen!

Het Manifesto veroorzaakte wel degelijk een soort opstand. Het moest inspireren tot de ontwikkeling van 'lean, clean code' door middel van een collaboratieve, adaptieve, feedback-gestuurde aanpak. Het bood een alternatief voor 'zwaarwichtige' processen als Waterfall en USDP.

Het is nu achttien jaar geleden dat het Agile Manifesto werd gepubliceerd. En daarmee eeuwenoude geschiedenis in de ogen van de meeste ontwikkelaars van vandaag. Om deze reden komt je begrip van Agile mogelijk niet overeen met de bedoeling van de makers.

Dit boek is bedoeld om alle feiten op een rijtje te zetten. Dit boek is een vergrootglas op Agile om die volledig er en nauwkeuriger in beeld te krijgen. Uncle Bob is een van de slimste mensen die ik ken, en hij heeft een grenzeloos enthousiasme voor programmeren. Als iemand Agile uit de doeken kan doen, is hij het wel.

Jerry Fitzpatrick
Software Renovatie Corporation
Maart 2019

1 Inleiding tot Agile

In februari 2001 kwam een groep van 17 software-experts bijeen in Snowbird, Utah, om te praten over de deplorabele staat van softwareontwikkeling. In die tijd werd de meeste software gemaakt met behulp van ineffectieve, loodzware, hoog-rituele processen zoals Waterfall en van overvolle diagrammen van het Rational Unified Process (RUP). Het doel van deze 17 experts was om een manifest op te stellen voor een effectievere, lichtere aanpak.

Dit was een allesbehalve gemakkelijke opgave. De 17 hadden verschillende ervaringen en sterk uiteenlopende meningen. Het was verre van kansrijk dat zo'n groep tot eensgezindheid zou komen. En toch, tegen alle verwachtingen in, werd consensus bereikt, werd het Agile Manifesto geschreven en werd een van de krachtigste en duurzaamste stromingen op softwaregebied geboren.

Stromingen in software volgen een voorspelbaar pad. In het begin is er een minderheid van enthousiaste supporters, een andere minderheid van enthousiaste tegenstanders en een grote meerderheid van hen die het niets kan schelen. Veel stromingen sterven in dat stadium of laten het daarbij. Denk aan aspectgericht programmeren, logisch programmeren of CRC-kaarten. Sommige steken de kloof over en worden buitengewoon populair en controversieel. Sommige slagen er zelfs in om de controverse achter zich te laten en gewoon deel uit te gaan maken van het reguliere gedachtegoed. Objectoriëntatie (OO) is een voorbeeld van dat laatste. En ook Agile.

Zodra een stroming populair wordt, vervaagt de naam van die stroming helaas door onbegrip en usurpatie. Producten en methoden die niets te maken hebben met de feitelijke stroming, lenen de naam om profijt te trekken uit de populariteit en het gewicht van die naam. En zo is het ook gegaan met Agile.

Het doel van dit boek, dat bijna twee decennia na Snowbird is geschreven, is om alles recht te zetten. Dit boek is een poging om zo pragmatisch mogelijk te zijn en Agile te beschrijven zonder nonsens en in niet mis te verstane bewoordingen.

Hier worden de fundamenten van Agile gepresenteerd. Velen hebben deze ideeën verfraaid en uitgebreid – en daar is niets mis mee. Die extensies en verfraaiingen zijn echter niet Agile. Ze zijn Agile plus nog iets anders. Wat je hier gaat lezen, is wat Agile ís, wat Agile was en wat Agile onvermijdelijk altijd zal zijn.

De geschiedenis van Agile

Wanneer is Agile begonnen? Waarschijnlijk meer dan 50.000 jaar geleden toen mensen voor het eerst besloten samen te werken aan een gemeenschappelijk doel. Het idee om kleine tussendoelen te kiezen en daarna de voortgang te meten, is gewoon te intuïtief en te menselijk om te zien als een soort revolutie.

Wanneer begon Agile in de moderne industrie? Dat is moeilijk te zeggen. Ik stel me voor dat de eerste stoommachine, de eerste molen, de eerste verbrandingsmotor en het eerste vliegtuig werden geproduceerd met technieken die we nu Agile zouden noemen. De reden daarvoor is dat het nemen van kleine, afgemeten stappen gewoon natuurlijk en menselijk is en niet op een andere manier had kunnen gebeuren.

Dus wanneer ging Agile met software aan de slag? Ik wou dat ik een vlieg op de muur was geweest toen Alan Turing in 1936 zijn paper[1] schreef. Ik vermoed dat de vele 'programma's' die hij in dat boek neerpende, in kleine stappen zijn ontwikkeld met veel desk checking. Ik stel me ook voor dat de eerste code die hij schreef voor de Automatic Computing Engine, in 1946, in kleine stappen werd geschreven, met veel desk checking en zelfs met echte testen.

De begindagen van software zitten vol met voorbeelden van gedrag dat we nu zouden omschrijven als Agile. De programmeurs die de besturingssoftware voor de Mercury-ruimtecapsule schreven, werkten bijvoorbeeld in stappen van een halve dag die werden onderbroken door unittests.

Over deze periode is elders al veel geschreven. Craig Larman en Vic Basili schreven een geschiedenis die is samengevat op de wiki van Ward Cunningham[2], en ook in Larmans boek, *Agile & Iterative Development: A Manager's Guide*.[3]

Maar Agile was niet het enige wat er speelde. Er bestond al een concurrerende methodologie die aanzienlijk succes had gehad in de productie en de industrie in het algemeen: Scientific Management.

---

[1] Turing, A. M. 1936. On computable numbers, with an application to the Entscheidungsproblem [proof]. *Proceedings of the London Mathematical Society*, 2 (gepubliceerd 1937), 42(1):230–65. Je begrijpt dit artikel het beste door het meesterwerk van Charles Petzold te lezen: Petzold, C. 2008. *The Annotated Turing: A Guided Tour through Alan Turing's Historic Paper on Computability and the Turing Machine*. Indianapolis, IN: Wiley.

[2] Wards wiki, c2.com, is de originele wiki – de eerste die ooit op internet is verschenen. Moge hij lang worden opgediend.

[3] Larman, C. 2004. *Agile & Iterative Development: A Manager's Guide*. Boston, MA: Addison-Wesley.

Scientific Management is een top-down, command-and-control benadering. Managers gebruiken wetenschappelijke technieken om vast te stellen wat de beste procedures zijn om een doel te bereiken en geven vervolgens alle ondergeschikten opdracht om hun plan naar de letter te volgen. Met andere woorden, er is een grote planning vooraf, gevolgd door een zorgvuldig gedetailleerde implementatie.

Scientific Management is waarschijnlijk zo oud als de piramides, Stonehenge of een van de andere grote werken uit de oudheid, omdat je je onmogelijk kunt voorstellen dat dergelijke werken zonder zouden zijn gemaakt. Nogmaals, het idee om een succesvol proces te herhalen is gewoon te intuïtief en menselijk om als een soort revolutie te worden beschouwd.

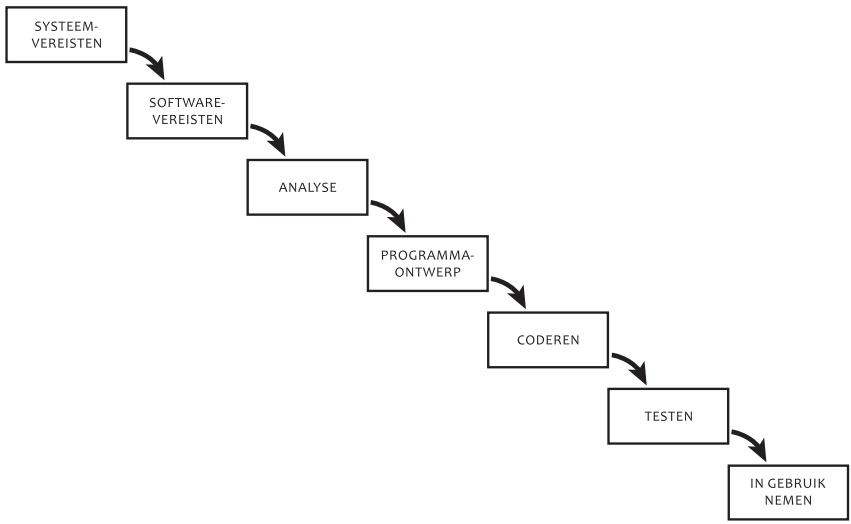
Scientific Management dankt zijn naam aan het werk van Frederick Winslow Taylor in de jaren 1880. Taylor formaliseerde en commercialiseerde de aanpak en verdiende een fortuin als managementconsultant. De techniek was enorm succesvol en leidde in de daaropvolgende decennia tot een enorme boost van efficiëntie en productiviteit.

En zo gebeurde het dat in 1970 de softwarewereld zich op het kruispunt van deze twee tegengestelde technieken bevond. Pre-Agile (Agile voordat het "Agile" heette) nam korte op elkaar reagerende stappen die werden gemeten en verfijnd om in een gerichte random walk naar een goed resultaat te zigzaggen. Scientific Management stelde actie uit tot na een grondige analyse en een daaruit voortvloeiend gedetailleerd plan. Pre-Agile werkte goed voor projecten met lage veranderingskosten en loste gedeeltelijk gedefinieerde problemen op met voorlopig gespecificeerde doelen. Scientific Management werkte het beste voor projecten met hoge veranderingskosten en loste zeer goed gedefinieerde problemen op met uitermate specifieke doelen.

De vraag was: wat voor soort projecten waren softwareprojecten? Hadden ze hoge veranderingskosten en waren ze goed gedefinieerd met specifieke doelen, of hadden ze lage veranderingskosten en waren ze gedeeltelijk gedefinieerd met voorlopige doelen?

Laat die laatste alinea maar voor wat het is. Bij mijn weten heeft niemand die vraag gesteld. Ironisch genoeg blijkt het pad dat we in de jaren zeventig hebben gekozen, meer per ongeluk dan met opzet te zijn ingeslagen.

In 1970 schreef Winston Royce een artikel[4] waarin hij zijn ideeën beschreef voor het managen van grootschalige softwareprojecten. Het artikel bevatte een diagram (afbeelding 1.1) dat zijn plan illustreerde. Royce was niet de maker van dit diagram, en hij pleitte er ook niet voor als plan. Het diagram diende in feite meer als handvat voor een idee dat hij op de volgende pagina's van zijn artikel onderuit zou halen.



Afbeelding 1.1: *Het diagram van Winston Royce dat inspireerde tot de ontwikkeling van Waterfall.*

Niettemin leidde de prominente plaatsing van het diagram – en de neiging van mensen om de inhoud van een artikel af te leiden uit het diagram op de eerste of tweede pagina – tot een dramatische paradigmaverschuiving in de software-industrie.

Het oorspronkelijke diagram van Royce leek zo veel op water dat langs de rotsen naar beneden stroomde, dat de techniek bekend werd als 'Waterfall'.

Waterfall was de logische afstammeling van Scientific Management. Het behelsde een grondige analyse, vervolgens een gedetailleerd plan en ten slotte de volledige uitvoering van dat plan.

[4] Royce, W. W. 1970. Managing the development of large software systems. *Proceedings, IEEE WESCON*, August: 1–9. Op http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf.

Ook al was het niet wat Royce aanraadde, het was wél het concept dat men uit zijn artikel haalde. En het was de daarop volgende drie decennia de dominante strategie.[5]

Hier verschijn ik in het verhaal. In 1970 – ik was achttien jaar oud – werkte ik als programmeur bij een bedrijf genaamd A.S.C. Tabulating in Lake Bluff, Illinois. Het bedrijf had een IBM 360/30 met 16K core, een IBM 360/40 met 64K core en een Varian 620/f minicomputer met 64K core. Ik heb de 360's geprogrammeerd in COBOL, PL/1, Fortran en assembler. Ik schreef alleen assembler voor de 620/f.

We moeten niet vergeten hoe het was om programmeur te zijn in die tijd. We schreven onze code met potlood op codeerformulieren en lieten die door ponskaartoperators in kaarten ponsen. We dienden onze zorgvuldig gecontroleerde kaarten in bij computeroperators die onze compilaties en tests uitvoerden tijdens de derde ploeg, omdat de computers het overdag te druk hadden met echt werk. Er zaten vaak dagen tussen het oorspronkelijk programmeren en de eerste compilatie, elke doorloop daarna was gewoonlijk één dag.

Met de 620/f lag het een beetje anders. Die machine was alleen voor ons team, dus we hadden er continue toegang toe, zeven dagen per week. We konden twee, drie, misschien zelfs vier doorlopen en tests per dag doen. Het team waarin ik zat bestond uit mensen die, in tegenstelling tot de meeste programmeurs van die tijd, ook konden typen. Dus we ponsten onze eigen kaarten in plaats van dat over te laten aan de grillen van de ponskaartoperators.

Welk strategie volgden we in die dagen? In ieder geval geen Waterfall. We hadden geen strategie voor het volgen van een gedetailleerd plan. We hackten erop los van dag tot dag, voerden compilaties uit, testten onze code en repareerden bugs. Het was een eindeloze lus zonder structuur. Het was ook geen Agile, zelfs geen Pre-Agile. Er was geen discipline in de manier waarop we werkten. Er was geen testsuite en er waren geen afgemeten tijdsintervallen. Het was gewoon programmeren en fixen, programmeren en fixen, dag na dag, maand na maand.

Ergens rond 1972 las ik in een vakblad voor het eerst over Waterfall. Het kwam me voor als een geschenk uit de hemel. Zou het echt zo zijn dat we het

[5] Opgemerkt moet worden dat mijn interpretatie van deze tijdlijn is aangevochten in hoofdstuk 7 van Bossavit, L. 2012. *The Leprechauns of Software Engineering: How Folklore Turns into Fact and What to Do About It*. Leanpub.

probleem vooraf konden analyseren, dan een oplossing voor dat probleem ontwerpen en dat ontwerp vervolgens implementeren? Konden we echt een planning ontwikkelen op basis van die drie fasen? Als we klaar waren met de analyse, zouden we dan echt voor een derde klaar zijn met het project? Ik voelde de kracht van het concept. Ik wilde het geloven. Want als het werkte, was een droom waarheid geworden.

Blijkbaar was ik niet de enige, want ook veel andere programmeurs en programmeerstudio's waren door het virus gegrepen. En, zoals ik al eerder zei, Waterfall begon onze manier van denken te domineren.

Het domineerde, maar het werkte niet. De volgende dertig jaar bleven ik, mijn medewerkers en mijn medeprogrammeurs over de hele wereld maar proberen en proberen om die analyse en dat ontwerp goed voor mekaar te krijgen. Maar elke keer dat we dachten dat we het hadden, glipte het ons tijdens de implementatiefase door de vingers. Al onze maanden van zorgvuldige planning verdampten voor de vlammende ogen van managers en klanten in het niets, door de onvermijdelijke wilde race naar vreselijk vertraagde deadlines.

Ondanks de vrijwel onophoudelijke stroom van mislukkingen volhardden we in de Waterfall-obsessie. Hoe kon dit immers fout gaan? Hoe konden grondige analyse van het probleem, zorgvuldig ontwerp van een oplossing en implementatie van dat ontwerp keer op keer zo spectaculair mislukken? Het was ondenkbaar dat het probleem aan die strategie te wijten was. Het probleem moest bij ons liggen. Op de een of andere manier deden we iets verkeerd.

Hoezeer Waterfall ons beheerste, kan worden afgelezen aan de taal van die dagen. Toen Dijkstra in 1968 met Structured Programming kwam, kwamen Structured Analysis[6] en Structured Design[7] niet lang daarna. Toen Object-Oriented Programming (OOP) populair begon te worden in 1988, kwamen Object-Oriented Analysis[8] en Object-Oriented Design[9] (OOD) ook niet lang

[6] DeMarco, T. 1979. *Structured Analysis and System Specification*. Upper Saddle River, NJ: Yourdon Press.

[7] Page-Jones, M. 1980. *The Practical Guide to Structured Systems Design*. Englewood Cliffs, NJ: Yourdon Press.

[8] Coad, P., and E. Yourdon. 1990. *Object-Oriented Analysis*. Englewood Cliffs, NJ: Yourdon Press.

[9] Booch, G. 1991. *Object Oriented Design with Applications*. Redwood City, CA: Benjamin-Cummings Publishing Co.

daarna. Dit drietal, dit driemanschap van fasen, had ons in zijn greep. We konden ons simpelweg geen andere manier van werken voorstellen.

En toen ineens konden we dat wél.

Het begin van de Agile-reformatie begon eind jaren tachtig, begin jaren negentig. De Smalltalk-gemeenschap begon er in de jaren tachtig tekenen van te vertonen. Er waren signalen in Booch's boek uit 1991 over OOD[9]. Meer uitgesproken daagde het op in Cockburns Crystal Methods in 1991. De Design Patterns-gemeenschap begon erover te praten in 1994, aangespoord door een artikel van James Coplien.[10]

In 1995 hadden Beedle,[11] Devos, Sharon, Schwaber en Sutherland hun beroemde artikel over Scrum geschreven.[12] En de sluizen gingen open. Er was een bres geslagen in het bastion van Waterfall, er was geen weg terug.

Hier kom ik weer in beeld. Wat volgt komt uit mijn geheugen en ik heb niet geprobeerd het te verifiëren bij de andere betrokkenen. Je mag daarom aannemen dat in deze herinnering van mij veel weggelaten is en nog meer dat apocrief is, of op zijn minst enorm onnauwkeurig. Maar geen paniek, ik heb in ieder geval geprobeerd het een beetje vermakelijk te houden.

Ik ontmoette Kent Beck voor het eerst op de PLoP-94,[13] waar Copliens artikel werd gepresenteerd. Het was een informele bijeenkomst en er kwam niet veel van terecht. Ik ontmoette hem vervolgens in februari 1999 op de OOP-conferentie in München. Maar tegen die tijd wist ik al veel meer over hem.

In die tijd was ik C++- en OOD-consultant en vloog heen en weer om mensen te helpen bij het ontwerpen en implementeren van applicaties in C++ met behulp van OOD-technieken. Mijn klanten begonnen me te vragen naar de strategie. Ze hadden vernomen dat Waterfall niet samenging met OO, en ze

[10] Coplien, J. O. 1995. A generative development-process pattern language. *Pattern Languages of Program Design*. Reading, MA: Addison-Wesley, p. 183.

[11] Mike Beedle werd op 23 maart 2018 in Chicago vermoord door een geestelijk gestoorde dakloze man die 99 keer eerder was gearresteerd en weer vrijgelaten. Hij had opgenomen moeten worden. Mike Beedle was een vriend van mij.

[12] Beedle, M., M. Devos, Y. Sharon, K. Schwaber, en J. Sutherland. *SCRUM: An extension pattern language for hyperproductive software development*. Op http://jeffsutherland.org/scrum/scrum_plop.pdf.

[13] Pattern Languages of Programming was een conferentie die in de jaren negentig werd gehouden in de buurt van de Universiteit van Illinois.

wilden mijn advies. Ik was het eens[14] over de mix van OO en Waterfall en had zelf veel nagedacht over dit idee.

Ik had er zelfs over gedacht mijn eigen OO-strategie te schrijven. Gelukkig heb ik die poging al snel opgegeven, omdat ik op de stukken van Kent Beck over Extreme Programming (XP) stuitte.

Hoe meer ik las over XP, hoe meer ik gefascineerd raakte. De ideeën waren revolutionair (althans dat dacht ik toen). Ze waren logisch, vooral in een OO-context (alweer, dat dacht ik toen). En dus wilde ik graag meer leren.

Tot mijn verbazing merkte ik op de OOP-conferentie in München dat ik een lezing gaf aan de andere kant van de hal dan Kent Beck. Ik kwam hem tijdens een pauze tegen en zei dat we elkaar bij de lunch moesten spreken over XP. Die lunch vormde de basis voor een belangrijke samenwerking. Mijn gesprekken met hem leidden ertoe dat ik naar zijn huis in Medford, Oregon vloog om met hem samen te werken aan het ontwerp van een cursus over XP. Tijdens dat bezoek maakte ik voor het eerst kennis met Test-Driven Development (TDD) en was ik verkocht.

In die tijd runde ik een bedrijf genaamd Object Mentor. We werkten samen met Kent aan een vijfdaagse bootcampcursus over XP die we XP Immersion noemden. Van eind 1999 tot 11 september 2001[15] waren ze een groot succes! We hebben honderden mensen opgeleid.

In de zomer van 2000 nodigde Kent een uitgelezen groep mensen uit de XP- en Patterns-gemeenschap uit voor een bijeenkomst in de buurt van zijn woonplaats. Hij noemde het de "XP Leadership"-bijeenkomst. We maakten boottochtjes op de rivier de Rogue en wandelden langs de oevers. En we kwamen samen om te beslissen wat we precies met XP wilden.

Een van de ideeën was om een non-profitorganisatie rond XP op te zetten. Ik was hier voorstander van, maar veel anderen niet. Ze hadden blijkbaar een ongunstige ervaring gehad met een soortgelijke groep die was opgericht rond Design Patterns. Ik verliet die bijeenkomst gefrustreerd, maar Martin Fowler volgde me naar buiten en stelde voor dat we elkaar later in Chicago zouden ontmoeten om het uit te praten. Ik ging akkoord.

[14] Dit is een van die vreemde toevalligheden die van tijd tot tijd gebeuren. Er is niets bijzonders aan het feit dat OO minder waarschijnlijk te combineren is met Waterfall, en toch kreeg die meme in die tijd veel aandacht.

[15] De betekenis van die datum mag niet over het hoofd worden gezien.

Dus ontmoetten Martin en ik elkaar in de herfst van 2000 in een café in de buurt van het ThoughtWorks-kantoor waar hij werkte. Ik beschreef hem mijn idee om alle met elkaar strijdende voorstanders van lichtgewicht processen bijeen te brengen om te komen tot een manifest van eenheid. Martin deed verschillende aanbevelingen voor een uitnodigingenlijst en we schreven samen de uitnodiging. De uitnodigingsbrief heb ik later die dag verstuurd. Het onderwerp was *Light Weight Process Summit*.

Een van de genodigden was Alistair Cockburn. Hij belde me om te zeggen dat hij op het punt stond een soortgelijke bijeenkomst te beleggen, maar dat hij onze uitnodigingenlijst beter vond dan die van hem. Hij bood aan zijn lijst met die van ons samen te voegen en het nodige werk te doen om de vergadering op te zetten als we ermee instemden die te houden in het skigebied Snowbird, in de buurt van Salt Lake City.

De bijeenkomst in Snowbird was dus gepland.

Snowbird

Ik was nogal verrast dat zo veel mensen ja zeiden op de uitnodiging. Ik bedoel, wie wil er nou echt een bijeenkomst bijwonen met de titel "The Light Weight Process Summit"? Maar hier zaten we dan allemaal in de Aspen-kamer van de Lodge in Snowbird.

We waren zeventien man sterk. We hebben het sindsdien zwaar te verduren gekregen, omdat we zeventien blanke mannen van middelbare leeftijd waren. Die kritiek is tot op zekere hoogte terecht. Er was echter wel één vrouw, Agneta Jacobson, uitgenodigd, maar die kon niet komen. En tenslotte bestond de overgrote meerderheid van senior programmeurs in de wereld destijds uit blanke mannen van middelbare leeftijd. Waarom dit zo was, is een verhaal voor een andere tijd en een ander boek.

Die zeventien van ons vertegenwoordigden nogal wat verschillende gezichtspunten, waaronder vijf verschillende lichtgewicht processen. Het grootste cohort was het XP-team: Kent Beck, ikzelf, James Grenning, Ward Cunningham en Ron Jeffries. Vervolgens kwam het Scrum-team: Ken Schwaber, Mike Beedle en Jeff Sutherland. Jon Kern vertegenwoordigde Feature-Driven Development en Arie van Bennekum vertegenwoordigde de Dynamic Systems Development Method (DSDM). Ten slotte vertegenwoordigde Alistair Cockburn zijn Crystal-familie van processen.

De rest van de mensen waren relatief ongebonden. Andy Hunt en Dave Thomas waren pragmatische programmeurs. Brian Marick was testconsulent. Jim Highsmith was consulent voor softwarebeheer. Steve Mellor hield ons eerlijk, want hij vertegenwoordigde de Model-Driven-filosofie, die door velen van ons werd gewantrouwd. En ten slotte Martin Fowler, die, hoewel hij nauwe persoonlijke banden had met het XP-team, sceptisch stond tegenover elk proces met een 'handelsmerk' en ze allemaal wel iets vond hebben.

Ik herinner me niet veel van de twee dagen dat we bijeenkwamen. Anderen die erbij waren herinneren het zich anders dan ik.[16] Ik ga je dus gewoon vertellen wat ik me herinner, en raad je aan het te zien als de bijna twintig jaar oude herinnering van een 65-jarige. Ik vergeet misschien een paar details, maar de essentie is waarschijnlijk correct.

Er was op de een of andere manier afgesproken dat ik de bijeenkomst zou openen. Ik bedankte allen voor hun komst en stelde voor dat het onze missie moest zijn om te komen tot een manifest waarin wordt beschreven wat we denken gemeen te hebben met al deze lichtgewicht processen en softwareontwikkeling in het algemeen. Toen ging ik zitten. Ik geloof dat dat mijn enige bijdrage aan de bijeenkomst was.

We deden het standaard soort dingen waarbij we problemen op kaarten schreven en de kaarten vervolgens op de vloer sorteerden naar verwantschap. Ik weet niet echt of dat ergens toe heeft geleid. Ik herinner me alleen dat ik het deed.

Ik weet niet meer of de magie op de eerste of op de tweede dag gebeurde. Ik geloof dat het tegen het einde van de eerste dag was. Het kan de groepering naar verwantschap zijn geweest waardoor de vier waarden boven kwamen drijven, namelijk *Mensen en hun onderlinge interactie*, *Werkende software*, *Samenwerking met de klant* en *Inspelen op verandering*. Iemand schreef deze op het whiteboard en had toen het briljante idee om te zeggen dat deze de voorkeur verdienen, maar geen vervanging zijn voor de complementaire waarden van processen, tools, documentatie, contracten en planning.

Dit is het centrale idee van het Agile Manifesto, en niemand lijkt zich duidelijk te herinneren wie het voor het eerst op het bord heeft gezet. Ik meen me te

[16] Er was een recent gepubliceerd verhaal over het gebeuren in *The Atlantic*: Mimbs Nyce, C. 2017. De wintervakantie die de softwarewereld op zijn kop zette. *The Atlantic*. Dec 8. https://www.theatlantic.com/technology/archive/2017/12/agile-manifesto-a-history/547715/. Op het moment van schrijven heb ik dat artikel niet gelezen, omdat ik de herinnering die ik hier noteer niet wil vervuilen.

herinneren dat het Ward Cunningham was. Maar Ward denkt dat het Martin Fowler was.

Kijk naar de afbeelding op de webpagina agilemanifesto.org. Ward zegt dat hij deze foto nam om dat moment vast te leggen. Het toont duidelijk Martin aan het whiteboard met een aantal van ons daaromheen.[17] Dit ondersteunt Wards idee dat het Martin was die op het idee kwam.

Aan de andere kant is het misschien maar beter dat we het nooit echt zullen weten.

Op dat magische moment verzamelde de hele groep zich rond het whiteboard. Er was wat abracadabra met woorden, en een weinig tweaken en afstemmen. Zoals ik me herinner, was het Ward die de inleiding schreef: "Wij laten zien dat er betere manieren zijn om software te ontwikkelen door in de praktijk aan te tonen dat dit werkt en door anderen ermee te helpen." Anderen maakten daar kleine wijzigingen op en deden wat suggesties, maar het was duidelijk dat we klaar waren. Er was een gevoel van eenstemmigheid in de kamer. Geen meningsverschil. Geen argument. Zelfs geen echte discussie over alternatieven. De vier regels waren:

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

Zei ik dat we klaar waren? Zo voelde het. Maar er waren natuurlijk nog veel details uit te knobbelen. Om te beginnen, hoe zouden we dit ding dat we hadden gevonden gaan noemen?

De naam "Agile" stond nog niet vast. Er waren veel verschillende kanshebbers. Ik vond "Light Weight" wel wat hebben, maar niemand anders. Ze dachten dat het "inconsequent" impliceerde. Anderen vonden het woord "Adaptive" prima. "Agile" werd genoemd en een persoon merkte op dat dat momenteel trendy was in het leger. Hoewel niemand het woord "Agile" echt goed vond, was het uiteindelijk gewoon het beste uit een reeks slechte alternatieven.

[17] Van links naar rechts, in een halve cirkel rond Martin, toont die foto Dave Thomas, Andy Hunt (of misschien Jon Kern), mij (je kunt het zien aan de blauwe spijkerbroek en de Leatherman aan mijn riem), Jim Highsmith, iemand, Ron Jeffries en James Grenning. Er zit iemand achter Ron, en op de grond bij zijn schoen ligt een van de kaarten die we gebruikten bij de affiniteitsgroepering.