INTERVJU: JEFFREY SNOVER

Powershellplattformens far om programutvikling, prosjekter og samarbeid

Ingress:
Det å utvikle et program eller et helt system handler ikke bare om bits og bytes. Det sentriske menneskesynet er også svært viktig for sammensetningen av utviklingsteamet, og utgjøre en suksess.

Alexander Solaat Rødland
alexander@itpro.no

Intervju: Jeffrey Snover

Det å utvikle et program eller et helt system handler ikke bare om bits og bytes. Det sentriske menneskesynet er også svært viktig for sammensetningen av utviklingsteamet.

Det handler om hverdagen og virkeligheten for millioner av utviklere over hele verden. Ikke bare utviklere av IKT-systemer, men også de som jobber med design og utvikling av nye biler, fly eller sko for den saks skyld. Samarbeid og koordinering av informasjonsstrømmen er uansett det største stikkordet for et utviklingsteam.

I dette intervjuet vil oppfinneren av det vi i dag kjenner som «Powershell», fortelle om hva som kan gå galt i utviklingsprosesser, i tillegg til å gi deg og ditt team en idé om hvordan man kan løse et påbegynnende problem.

Om Jeffrey Snover

Jeffrey Snover har tittelen som "Distinguished Engineer" hos Microsoft, og jobber til daglig med Windows Server og System Center Datacenter, hvor han har overordnet ansvar for arkitekturen for begge produktene.

Jeffrey er oppfinneren av Windows Powershell, som er et objektorientert Microsoft-rammeverk for å automatisere administrative oppgaver for Windows-plattformen. Powershell er en del av .NET-rammeverket, og ble lansert i 2006.

Jeffrey er en hyppig foredragsholder på industri- og forskningskonferanser på en rekke administrative og utviklingsspråklige emner, blant annet som hovedtaler på Nordic Infrastructure Conference (NIC) i Oslo i 2013.

Du fortalte i keynote-talen på NIC i Oslo at i forbindelse med utviklingen av Windows PowerShell, oppstod det intern motstand – og det gikk så langt at du måtte gå ned i gradene for å fullføre prosjektet

Jeg var på den tiden sjefsarkitekt for alle våre Management-teknologier og produkter i Microsoft, og et utviklingslag skulle lage et erstatningsskall for UNIX. Utviklingsteamet så for seg å forbinde «k-shell» eller «bash» til systemet. Selv holdt jeg på med å eksperimentere med en type teknologi, og fortalte gruppen; «*nei, nei, nei – dere gjør det feil, og det vil aldri fungere på denne måten*». Normalt sett er jeg veldig god på å forklare ting, men selv etter å ha forklart om og om igjen, på alle mulige måter, forstod ikke gruppen hvordan de skulle gjøre det.

Min tanke var da at «*okey, jeg gjør noe galt her*» - og sa at jeg skulle skrive en demo.

Jeg satte i gang og skrev på kort tid en demo på rundt ti tusen linjer, som i utgangspunktet inneholdt alle de sentrale begrepene i PowerShell. Det var en herlig følelse å vise utviklingsfolkene resultatet, og jeg ble helt hektet. Det åpenbarte seg at denne kodesnutten var usedvanlig kraftig, og helt ærlig – en av de beste idéene jeg noen gang har hatt.

Når jeg så gikk til min sjef og forklarte at dette var noe vi virkelig burde satse på, kom motstanden fort. «Det gir ingen mening for vår organisasjon», svarte sjefen, med henvisning til at dette var et kommandolinjeverktøy og vi jobbet med «Windows». Samtidig mente hun at denne typen arbeid heller ikke passet til min lederstilling. Jeg var naturlignok helt uenig.

Etter mye diskusjon frem og tilbake, ble enden på visen at jeg lot meg degraderes for å kunne fortsette å utvikle denne plattformen.

Første hinder overstått – hvor lang tid tok det fra planlegging, til å utvikle, starte og tilslutt ende opp med en fungerende plattform?

Ja, det tok faktisk ganske lang tid. Det oppstod mange hindringer på veien som måtte overkommes, men ikke mange av dem var tekniske hindringer. Det var selvfølgelig noen tekniske hindringer også, men de største hindringene var innad i organisasjonen, finansiering og deretter kulturelt.

Jeg skrev «Monad Manifesto» tilbake i 2002 – og første versjon av PowerShell ble lansert i 2006, så det var ganske mange år. Dels fordi vi forsøkte å gjøre utviklingene i India, noe som ikke fungerte i det hele tatt, og på grunn av dette var om lag atten måneder av prosjektet bortkastet. Så vi måtte starte fra bunnen av igjen – ikke noe av koden fra India var brukbart – ah, for et rot!

Deretter var det også mye rot rundt «Longhorn», som også forårsaket mye forsinkelse. Generelt sett, tok utviklingen alt for lang tid – men det var ingen spesielt tekniske grunner til forsinkelsene.

Hva var grunnen til at utviklingen i India kollapset på denne tiden?

Det som skjedde var at vi prøvde å finansiere og utvikle vårt indiske utviklingssenter. Noe av det mest fornuftige vi gjør i Microsoft er at vi prøver å finne de beste talentene i verden. Her ønsket vi å skape et miljø der vi kunne få benytte dyktige programmerere fra India – uten at de trengte å flytte til Redmond for å arbeide for Microsoft.

Vi var dermed på utkikk etter prosjekter som kunne finansieres og utvikles på denne måten. Noen ville ha et erstatningsskall – men det var ikke en «big deal» som tidligere nevnt - og utviklingssenteret i India ble alternativet. Det ble opparbeidet et gruppe på sannsynligvis 12-15 utviklere i India, samt fem personer som sørget for styring og definering av prosjektet i Redmond.

Vel – det fungerte ikke. Tidssonene, spørsmål som gikk frem og tilbake skapte mye frustrasjon. Jeg stilte et spørsmål som de mottok tolv timer senere. De svarte, og det tok ytterligere tolv timer før jeg leste svaret. De forstod ikke spørsmålet, og jeg måtte konkretisere spørsmålet ytterligere … du skjønner tegningen. Samarbeidet og koordineringen fungerte ikke i det hele tatt, og vi måtte avslutte prosjektet der.

Så, etter å ha startet på bar bakke igjen, var det om lag ti-femten utviklere fra Redmond som jobbet med prosjektet.

Det har altså vært mange utviklere involvert i prosjektet. Vet du hvor mange som jobber med Powershell i dag?

Powershellplattformen har absolutt vært en stor investering for selskapet, og mange involverte utviklere og ingeniører. Du skjønner det, med PowerShell graver vi oss ut av et tretti- år dypt hull. Med det mener jeg at i tretti år var alt som ble gjort hos Microsoft GUI- basert, ingenting ble gjort med kommandolinjer. Det tar litt tid å omgjøre en slik prosess, men vi er godt på vei.

Til syvende og sist handler alt om innflytelse. Vi satte noen retningslinjer og krav, og utviklergruppene implementerer alle kommandoene, mens andre avdelinger legger til funksjoner osv.

For å komme til bunns i spørsmålet ditt, er det vanskelig å tegne en ring rundt utviklerteamet i PowerShell – det er så spredt utover i avdelinger og divisjoner, men jeg har fremdeles hovedutviklergruppen i PowerShell. Det er vanskelig å sette et konkret tall.

Figur 1 - Bilde fra TechEd-konferansen i Nord-Amerika i mai 2014. «Windows PowerShell Unplugged» med Don Jones (t.v.) og Jeffrey Snover (t.h.).

Møtte du på noen spesielle utfordringer rundt koding, maskinvare eller programvare underveis?

Vel, egentlig ikke. PowerShell er svært avansert teknologi, bruk av objektadapter «the type system» etc., men selve teknologien er veldig grei.

En ting jeg har opplevd som svært viktig, både i forbindelse med PowerShell-utviklingen og andre prosjekter, er å finne én person man stoler på. Forklar personen nøyaktig hvor du vil med prosjektet – og be alle som har idéer om å gå via den personen for å få et konsekvent synspunkt. I tilfellet med PowerShell, var det faktisk to personer jeg var avhengig av, og det var Jim Truth og Bruce Payette. De jobbet som én, og gjorde en fantastisk jobb. De ga meg noe som var minst ti ganger bedre enn hva jeg hadde håpet på.

Jeg hadde ganske beskjedne forventninger i starten. Det jeg var ute etter, var en automatiseringsmotor og kommandolinjer. Det Bruce og Jim leverte var et fantastisk miljø som spenner seg over både enkel, interaktiv bruk og hele veien opp til avansert systemprogrammering.

Tilbake til utviklingsprosessen av PowerShell. Var det noen spesielle utviklingsmetoder som ble brukt, eksempelvis XP (extreme programmering), SCRUM, eller valgte utviklerne selv hva som fungerte best selv?

Vi har vært igjennom en rekke utviklingsfaser. Først var det den tradisjonelle fossefallsmetoden – en metode jeg hatet! Hos Microsoft har vi noen veldig, veldig tunge prosesser som gjennomføres på flere plan og det er ikke alle som er like

velegnet til å finne ut av ting underveis. For eksempel prototyping og slikt, noe som gjør det veldig vanskelig å finne de beste løsningene.

Så på et tidspunkt byttet vi over til smidige utviklingsmetoder hvor blant annet SCRUM ble utforsket. SCRUM virket til sitt over en periode, men når ting skal synkroniseres over en så stor organisasjon og når det skal tilpasses Windows miljøet er det mye som må tas hensyn til, sant? Windows er det største showet på planeten. Enn hva du gjør, ikke kluss til Windows, for alt i verden! Om du gjør det klusser du til en multi-miliarder-industri, og dette er igjen en av grunnene til at vi følger prosessene slik vi gjør.

Ting som er verd å nevne er tidsfrister, altså, tidsfrister er virkelig viktig og timeplaner må holdes og hver enkelt må kunne være i stand til å si «*Jeg skal levere denne funksjonen på denne datoen, slik at du kan dra nytte av den. Slik kan du fortelle noen andre når du skal levere dine funksjoner til dem*». Slike ting er superviktig i en stor organisasjon og under utviklingen av Windows.

Det vi har akkurat nå er en form for hybrid-modell hvor vi noen ganger benytter SCRUM, mens vi i andre situasjoner bruker fossefallsmetodikken.

Går vi litt tilbake til når jeg startet, var det et tydelig skille mellom utviklerne, testerne og ledere. Hva som skjedde var at utviklerne skrev koden, så testet selvsagt testerne dette. Hva jeg mener er altså, de utviklet den ferdige funksjonelle koden. For meg gav dette ingen mening overhodet og så at dette var noe som måtte endres på og fortalte gruppen: «*Fra nå av skal utviklerne også skrive enhetstester til programkoden sin*».

Det tok ikke lange tiden før jeg måtte utdype denne meldingen, «*Ja, forresten, enhetstestene må bli laget samtidig som koden blir utviklet*». Utviklerne trodde altså at de først kunne utvikle programkoden før de på slutten av utviklingen kunne lage enhetstestene. Jeg var nødt for å gjøre det veldig tydelig for dem at enhetstestene ble levert inn samtidig som kildekoden. Ingenting skulle leveres med mindre det hadde enhetstestere klargjort.

Det stoppet selvsagt ikke her, for ikke lenge etterpå måtte jeg be dem om å sørge for at enhetstestene kjørte før de leverte inn koden. …for så å måtte utdype nok en gang, «*Nei, nei, nei – enhetstesten må **virke** før du sender inn koden*». Sett under ett var det en veldig merkelig utviklingskultur vi hadde tilbake til 2002-2003. Den kulturelle forandringen vi har gjort og streber etter enda er å sørge for at utviklerne er ansvarlig for kvaliteten på funksjonene, mens testerne retter seg mer mot de forskjellige scenarioene som kan oppstå, ytelse og slike ting.

Hele selskapet beveger seg nå mot denne arbeidsmodellen.

Hvor lenge varer en typisk iterasjon?

Nå er Windows PowerShell en Windows-komponent, så vi følger ruten for Windows som helhet. Slik som for 2012-lanseringen var det en treårig iterasjon.

Det jeg kan fortelle deg at det var tøft og vi produserte et fantastisk arbeid på de tre årene, før de rett etter forandret og sa «den neste iterasjonen blir et år!». Så med en

gang snudde vi oss rundt og leverte et forbedret produkt på et år. Som du sikkert skjønner er det først ved hver utgivelse at vi kan se fremover og se hvordan den neste timeplanen vil bli.

Det er en aldri så liten utfordring

Hvordan er din erfaring med å utvikle programvare fra bunnen av? Er det noen spesielle problemstillinger du må ta hensyn til? Noen tips?

Faktisk så har jeg jobbet på alle slags prosjekter. PowerShell var naturligvis fra bunnen av og mange av de tingene jeg har gjort opp gjennom årene har blitt utviklet fra bunnen av. Samtidig har jeg også tatt over en del prosjekter som sett under ett har vært i en ganske fæl tilstand for så måtte snu dem på rett spor, så jeg har arbeidet med begge typer og det er egentlig ganske forskjellige utfordringer på disse to problemstillingene.

Å starte helt fra begynnelsen tror jeg de fleste kan forestille seg følgende; Se for deg et helt blankt ark og vite hvor og hvordan du skal begynne.

Dette er noe mange undervurderer, for det er virkelig en tøff oppgave. Hos Microsoft har vi noen visjonærer som sier hvordan fremtiden vil bli og at vi om så-og-så mange år vil ha flyvende biler, vi vil ha datamaskiner som vet hva du egentlig ønsker å gjøre og hva du ikke vil.

Det krever en helt annen ferdighet og en må ha helt andre evner for å si «dette» er det første steget, og «dette» er det neste – og så videre. Altså det å begynne fra begynnelsen uten å ha noe som helst for så å se for seg et velfungerende system, program eller funksjon. Det er en aldri så liten utfordring.

Jeg kan nevne for deg, en av spøkene vi har her er at det er en hårfin linje mellom smidig og tilfeldig. Hvor noen personer gjør smidige utviklingsmetoder er det nesten som en tilfeldig spasertur langs programkoden og sluttresultatet.

La oss gå litt tilbake. Du nevnte noen utfordringer med å ha utviklingsteamet i India. Jeg antar at tidssonene, tilstedeværelsen og nærheten er viktig. Hvor viktig er

samhandlingsverktøyene for utviklingen. Helt generelt – hvilke kriterier er de viktigste for et velfungerende utviklingsteam?

Aller først vil jeg si at vi har et stort antall prosjekter som kjører vellykket I India, og det som fort skjer er, og du satte fingeren på det, er at de må bli gitt tilstrekkelig frihet til å kunne arbeide selvstendig uten noen form for oppdatering mot motparten som sitter langt unna.

Hva som ikke fungerer er når du har et prosjekt eller en utviklingsgruppe krever mye kommunikasjon og informasjonsutveksling på tvers av tidssoner og avstand. Hva jeg erfarte, og grunnen til at vi ikke lyktes med prosjektet var:

1. Utviklingsteamet vi brukte var et «junior-team», mens jeg var en «seniorutviklen» og leder.
2. De forstod ikke administrasjon og arbeidsstrukturen, så vi kunne ikke ha en effektiv samtale. Jeg spurte de om noe, hvor de ikke hadde den fjerneste anelse om hvordan administrasjonen skulle bli utført.
3. Det var en helt ny teknologi (.NET)
4. Fremgangsmetoden og arkitekturen vi skulle utvikle var en så radikal endring fra hva vi noen gang hadde gjort. Dette førte til at vi ikke kunne ha noen korte samtaler og enkle meldingsutvekslinger. Alt måtte være en lang avhandling med påfølgende lange samtaler.

Det var de kritiske tingene, men samtidig var det mange artige utfordringer knyttet til kultur. På den tiden – og det er gjerne slik enda. Det indiske samfunnet er et svært status- orientert samfunn. Hvor din sosiale tilhørighet virkelig betyr noe for deg som person og din plassering i systemet.

Hva som skjedde var at jeg var en partner-arkitekt og hva jeg sa, altså de ordene som kom ut ifra min munn. De ordene var som Guds ord, og det er ikke slik jeg arbeider. Ikke i det heletatt.

Min jobb er ikke å komme på de beste ideene, men å finne de beste ideene og formidle disse videre for så å realisere dem. Så hvis du har en bedre ide enn hva jeg har så må du fortelle meg. Det er den beste løsningen vi går videre med.

Til sist, hva mener du er viktigst for å fremme et vellykket utviklingsteam? Hvilke premisser og goder er viktig å dra frem?

Da jeg arbeidet for Tivoli hadde vi en spøk som sa at en velsmurt ingeniøravdeling høy på koffein er noe som var til bedriftens beste. Men helt ærlig så har jeg arbeidet for mange forskjellige selskaper.

Det var selskaper hvor det var fri flyt av alkohol på kontoret, og jeg har arbeidet hos selskaper hvor en ansatt ble oppsagt når sjefen fant en flaske med brennevin i baksetet på bilen til vedkommende. Sjefen gikk forbi bilen på parkeringsplassen og sjekket opp i registreringsnummeret. Før vi visste ord av det var personen oppsagt.

Det er egentlig noe om kan bli sagt om alt og alle former, fordeler og ulemper, men personlig tror jeg det er enklere enn som så. Ingeniører er spesielle mennesker. Vi er individer som elsker å skape og være til nytte.

Jeg kan fortelle deg en historie. Når Microsoft ringte meg (de rekrutterte Jeffrey, red. anm.), ville jeg ikke arbeide for dem i det hele tatt. Jeg var overhodet ikke interessert. Jeg likte ikke programvaren deres, jeg likte ikke fremgangsmåten og så videre.

Det som skjedde var at en leder som jeg kjente og hadde en god tone med ringte meg. Jeg svarte og han overtalte meg til å komme opp på et besøk for å møte ham. Ikke Microsoft, men ham. Jeg tenkte hvorfor ikke, han er en grei person så jeg kan høre hva han har å si. Når jeg først kom til ham og ble vist rundt og bedre kjent med de som arbeidet her oppdaget jeg at de som var her var fantastiske personer. Ikke bare var de fantastisk, de var morsomme og spennende individer. Så det å arbeide med spennende og positive mennesker forbedrer alles hverdag og arbeidssituasjon.

Det som virkelig overbeviste meg, var muligheten for å kunne utgjøre en forskjell. Når jeg arbeidet for andre selskaper tidligere har jeg gjerne utviklet programvare som påvirket tusener, ti-tusener eller i beste fall noen hundrede tusen mennesker. Var du heldig var en million mennesker avhengig av dine løsninger og programvare.

Hos Microsoft er det annerledes. Her har vi muligheten for å kunne ta våre ideer videre, utvikle dem og levere de som løsninger som blir brukt av flere milliarder mennesker. Jeg kan fortelle deg at det er ingenting som tvinger deg ut av sengen av fri vilje, som muligheten til å kunne utgjøre en så stor forskjell i verden. Om du en dag våkner og har en god ide, og er fast bestemt på å følge denne ideen kan den påvirke så mange menneskeliv. Jeg tror dette er noe av det viktigste å trekke frem når det gjelder å motivere utviklere.

Du skjønner det, gratis Cola er flott det, jeg liker slike ting, men i det store og hele er ikke dette hva som er viktig - og helt ærlig, hadde jeg hatt noen bedritne kontorer, forferdelige parkeringsforhold og en tragisk kantine – men likevel hatt muligheten til å gjøre et meningsfullt arbeid som påvirker så mange menneskers liv. Jeg hadde så definitivt valgt en slik jobb!

På vegne av redaksjonen ønsker vi å takke Jeffrey og hans vilje til å dele sin kunnskap og erfaring med oss. Jeg ønsker også å gi en spesiell takk til Arlen Espeland, som har hjulpet til med intervjuet og transkriberingen.