

## Arkitektur for Agile projekter

Agile udviklingsmetoder har været i brug i mere end et halvt århundrede. Men i 1990'erne accelererede anerkendelsen af den iterative softwareudvikling, og en række forskellige agile "opskrifter" blev udviklet, fx Extreme Programming, SCRUM, DSDM og mange andre. Det er et fællestræk for disse metoder, at software løsningens egenskaber specificeres og implementeres løbende i tæt samarbejde mellem forretning og udviklere, og at udviklingsforløbet er opdelt i korte trin, som hver resulterer i levering af konkret funktionalitet.

### The Agile Manifesto

I 2001 mødtes skaberne af mange forskellige agile udviklingsmetoder, og definerede et fælles grundlag for agil udvikling i "[Manifesto for Agile Software Development](#)". Her fremhæves værdien af den personlige interaktion i teamet, demonstration af konkrete funktioner, samarbejde med kunden, og evnen til at håndtere forandring i udviklingsforløbet. Manifestet suppleres af en række principper for agil softwareudvikling, som primært adresserer værdier, processer og roller i samarbejdet.

### Arkitekturen hjælper Agile

Det kan det være hensigtsmæssigt at kombinere en agil udviklingsmetode med en række principper, som definerer rammerne for udviklingsprojektet. Her tænker vi først og fremmest på de overordnede arkitekturprincipper, som beskriver de forretningsmæssige og tekniske rammer, man har valgt at arbejde inden for.

Allan Bo Rasmussen  
Direktør



De agile udviklingsmetoder blev opfundet i erkendelse af, at den traditionelle vandfaldsmodel let kommer til kort, når opgavestilleren ikke er i stand til at specificere løsningen ved projektets start, eller når forudsætningerne ændrer sig, så de stillede krav – og løsningen – må justeres. Vi bliver jo alle klogere i projektets forløb, og de agile metoder tager udgangspunkt i dette ved at sætte tilpasningsevnen i højsædet, en lille smule ligesom Darwin gjorde det.

Når et projekt følger en agil udviklingsmodel, er den væsentligste fordel, at man kommer hurtigere i gang med at programmere, og dermed på et tidligt tidspunkt får indhøstet nogle af de erfaringer, der manglede ved projektets start. Dermed tackler man risikoen for at spilde ressourcer på at lave noget der ikke er holdbart – eller som der slet ikke er brug for.

Men til gengæld har den agile udvikling andre indbyggede risici: Når man fx vælger standarder og teknologier, inden den samlede løsning er beskrevet, kan man nemt komme ud for, at det bliver svært at opfylde de stillede krav på den valgte platform. Og når løsningen kun i meget begrænset omfang dokumenteres, kan det blive besværligt at drifte og vedligeholde softwaren, efter at udviklingsteamet er draget videre til nye, spændende projekter.

Derfor kan det være hensigtsmæssigt at kombinere en agil udviklingsmetode med en række principper, som definerer rammerne for udviklingsprojektet. Her tænker vi først og fremmest på de overordnede arkitekturprincipper, som beskriver de forretningsmæssige og tekniske rammer, man har valgt at arbejde inden for. Det kunne fx handle om:

1. Forretningsmæssige forudsætninger og mål, herunder målbare succeskriterier
2. Hvordan løsningen indgår i forretningsprocesserne, og hvilke fordele den skal give
3. Hvilke data der skal behandles i løsningen og hvordan de skal være organiseret
4. Løsningens funktionelle afgrænsning i forhold til andre systemer i virksomheden
5. Hvilke teknologiske platforme skal løsningen køre på og hvorledes skal den indpasses i infrastrukturen

Ved at fastlægge rammerne i principform, giver man frihed til den kreative iteration, som bærer udviklingsprocessen. Men samtidig sparer man også en masse tid til fx afprøvning af alternative teknologier eller til at diskutere scope og integration i forhold til omgivelserne. Arkitekturen er en fælles referenceramme, der understøtter hurtig udvikling, netop fordi den definerer frihedsgraderne i hittepåsomheden, og hjælper til at opfylde nye krav med allerede eksisterende praksis og komponenter.

Foruden arkitekturprincipperne kan det være nyttigt at fastlægge andre rammebetingelser for udviklingsprocessen, fx budgetmæssige krav, og krav til den dokumentation der skal følge med løsningen ved aflevering og idriftsættelse. Der kan også være behov for at koordinere projektets forløb i forhold til dets omgivelser – her kan man med fordel se i retning af PRINCE2 for at hente inspiration.

Men til syvende og sidst er valget af projektmodel et spørgsmål om at afveje virksomhedens behov for agilitet mod behovet for styring og kontrol. Og at indregne både virksomhedens og projektets forudsætninger i overvejelserne.

### Hvad taler for valget af agile metoder/elementer

- Projektet er ikke kritisk for virksomheden
- Udviklere med høj erfaring
- Kravene til løsningen skifter ofte
- Antallet af udviklere er lavt
- Virksomhedskulturen trives med kaos

### Hvad taler for valget af planlagt/styret metode

- Projektet er kritisk for virksomheden
- Udviklerne er nye eller uerfarne
- Kravene er velkendte og faste
- Projektet involverer mange udviklere
- Virksomhedens kultur er baseret på orden

For mange virksomheder vil det være relevant at inddrage elementer fra både de agile modeller og fra de traditionelle, kontrollerede forløb, som scorer højt på Carnegie Mellon Universitetets CMMI model. For en virksomhed med flere (mange) projekter er det oplagt at følge en projektmodel, som er fleksibel. Det kan fx realiseres med tilvalg/fravalg af de enkelte elementer, baseret på de konkrete behov i det enkelte projekt.

Læs mere her:

Manifesto for Agile Software Development.

<http://agilemanifesto.org/>

Kelly Waters blog: All About Agile.

<http://www.allaboutagile.com/>

Software Engineering Institute: CMMI or Agile: Why Not Embrace Both.

<http://www.sei.cmu.edu/pub/documents/08.reports/08tn003.pdf>

Iterative and Incremental Development: A brief History.

<http://www2.umassd.edu/SWPI/xp/articles/r6047.pdf>