Knas med udviklingsprojekterne?

Iterativ udvikling kan være løsningen!

Af Cap Gemini:
Direktør Carsten Sennov, carsten.sennov@capgemini.dk
Chefkonsulent Lennart Klamer, lennart.klamer@capgemini.dk
Chefkonsulent Ole Jepsen, ole.jepsen@capgemini.dk

”Hellere mange små succeser end én stor fiasko”

I en verden hvor forretningsudviklingen og den teknologiske udvikling accellererer er der ikke tid til at vente på langsommelige vandfaldsprojekter. Systemudviklingsprojekterne skal accellereres, ellers kan de ikke være til gavn. Løsningen er letvægtsorienterede, samarbejdsorienterede og frem for alt *iterative* udviklingsprojekter.

1. Indledning

Mange oplever i disse år, at udvikling af IT systemer sker under et stadigt større pres fra omgivelserne. Der er adskillige faktorer, som gør, at projekterne bliver presset.
En af de væsentligste faktorer er virksomhedernes skærpede konkurrencesituation. Når virksomhederne presses, så er der ofte brug for hurtigere og billigere udvikling af nye IT-systemer, til at understøtte udviklingen af nye produkter og til at effektivisere de interne arbejdsgange.
En anden væsentlig faktor er den stadigt hastigere teknologiske udvikling, som konstant udfordrer IT leverandørernes og projekternes beslutninger om de tekniske arkitekturer, som udgør fundamentet for de systemer, der udvikles.
Ovenstående giver ofte store udfordringer i udviklingsprojekter – og dermed store problemer for de virksomheder og forretningsområder, hvis dagligdag er mere og mere afhængig af at have nogle velfungerende og tidssvarende IT systemer.
Udfordringerne er store – uanset hvordan vi vælger at køre udviklingsprojekterne. De fleste virksomheder har tidligere kørt næsten alle større projekter som vandfaldsprojekter, men flere og flere synes at klare udfordringerne bedre ved at køre projekterne som iterative projekter. Hvordan og hvorfor? Det er netop hvad du får ud af at læse denne artikel.

2. Sædvanlige problemer i systemudviklingsprojekter

Måske kan du nikke genkendende til nogle af disse situationer:

- En af dine projektledere beroligede dig for nogle få dage siden ved at fortælle dig, at alt gik planmæssigt. Nu kommer den samme projektleder til dig, fordi han pludselig har opdaget, at projektet er bagud… med flere uger!!!

- En projektgruppe er ved at være færdig med et system. Planen er ganske vist blevet justeret nogle gange p.g.a. forsinkelser. Forsinkelserne bundede i, at en del af funktionaliteten i systemet måtte kodes om, fordi der var en hel række misforståelser omkring kravspecifikationen. Nu hvor systemet – endelig – er tæt på at kunne implementeres fuldt ud får du besked om, at kunden har brug for lidt tid til at checke, at det nu også er de rigtige krav, der har ligget til grund for udviklingen af systemet. Du bliver bedt om at udskyde implementeringen indtil dette arbejde med at checke og konsolidere kravene er færdiggjort.

- Et større system er netop gået ind i den endelige testfase. De første par dage gik det meget godt. Der blev ganske vist fundet nogle fejl, men de er nu ved at blive rettet. De sidste par dage er det imidlertid begyndt at vælte ind med fejlrapporter, som ikke er fejl, men som efter udviklernes bedste overbevisning er ændringer i forhold til kravspecifikationen. Det er brugerne absolut ikke enige i. De mener, at udviklerne har misforstået store dele af kravspecifikationen.

Faktaboks: Iterative projekter

- Projekter opdeles i "mini-projekter", som gennemføres i korte timeboxes eller *iterationer* på 1-3 måneder.
- Overordende krav til funktionalitet defineres fra starten af projektet – og fordeles ud på iterationerne.
- I hver iteration arbejdes der udelukkende videre med krav og udvikling af den funktionalitet, som er placeret i iterationen.
- Systemet implementeres helt ud til brugerne – så tit som muligt – og allerhelst efter hver iteration.
- Datoerne for timeboxes fastsættes up front – og de ændres *under ingen omstændigheder* undervejs. Hvis der opstår problemer med tiden, så bortskæres/udskydes funktionalitet, i stedet for at udvide iterationen.

Hvis du ikke genkender noget af ovenstående fra din egen virksomhed, så har du sikkert hørt om andre… Men hvad kan det skyldes, at der er så store problemer med flere og flere udviklingsprojekter?

3. Behov kontra krav

En del af forklaringen er nok, at de fleste systemudviklingsprojekter starter med, at en virksomhed har et ønske, et behov eller en "drøm", som de forsøger at få realiseret gennem udvikling af et nyt IT-system.

I flere og flere projekter viser det sig, at det egentlige behov, som sponsoren gerne vil betale nogle penge for at få opfyldt – er meget overordnet. Behovet kan tolkes på mange forskellige måder. Behovet kan opfyldes på flere forskellige måder – og stadig give sponsoren "value for money".

3.1 Vi beskriver kravene for tidligt

Dette står i skarp kontrast til den måde vi har arbejdet med krav og kravstyring i traditionelle vandfaldsorienterede udviklingsprojekter. Vores rygmarvsreaktion har været at få omskrevet alle de overordnede behovene til en detaljeret og fastlåst kravspecifikation for hele projektet. Det har vi gjort "for at kunne estimere og planlægge projektet, for at kunne lave en aftale/kontrakt – og for at vide, hvad det er for et system vi skal bygge". Kravspecifikationen har således været hele udgangspunktet – ja hele fundamentet – for projektet.

Logisk og rationelt betragtet giver den traditionelle vandfaldsorienterede angrebsvinkel god mening. "Vi kan jo ikke bygge noget, før vi ved, hvad det er vi skal bygge" – har rationalet været.

4. Problemer med vandfaldsprojekter

Erfaringerne har imidlertid vist, at der er masser af problemer indbygget i den traditionelle vandfaldsorienterede angrebsvinkel, hvor vi starter ethvert projekt med at udarbejde en detaljeret kravspecifikation. Nogle af problemerne er:

4.1 Uoverskueligt

Vi har svært ved at overskue alle krav på én gang. Derfor glemmer vi ofte noget. Det er irriterende, når vi netop bygger hele vores udviklingsproces på, at vi har tænkt på det hele fra starten af.

4.2 Omgivelserne ændrer sig

Kravspecifikationen skrives med udgangspunkt i tingenes tilstand på det tidspunkt den skrives. I løbet af projektets levetid ændrer virksomhedens og dermed projektets forudsætninger sig, hvilket ofte betyder at nogle af kravene ender med at være ligegyldige, forkerte og mangelfulde. I

Faktaboks - vandfaldsprojekter

- Som det første beskrives alle krav til systemet i en kravspecifikation.
- Kravspecifikationen fastfryses. Man forsøger at undgå ændringer.
- Kravspecifikationen danner grundlag for tidsplan, prissætning, analyse, design, etc.
- Når analysen er gennemført fastfryses den, og når designet er færdigt fastfryses det – o.s.v.
- Hver fase i projektet danner således et fast fundament for næste fase. Et fundament, som man med fastfrysningen forsøger at holde så stabilt som muligt.
- Hele systemet udvikles og implementeres på en gang, når alle krav er indarbejdet i systemet.

en verden, hvor vi forsøger at få samtlige krav beskrevet fuldstændigt fra starten – for derefter at fastlåse dem – er det frygteligt svært, at ændre projektets kurs, hvis det alligevel viser sig at blive nødvendigt undervejs i projektet.

4.3 Behovene udvikler sig

Omskrivningen af de overordnede behov (”drømmen”) til en detaljeret kravspecifikation er for en stor del en kreativ proces. Når kravspecifikationen er færdig, så ønsker vi ikke, at den udvikler sig mere, for så ændrer vi jo samtidig på projektets forudsætninger. Desværre er det bare sådan, at den kreative proces *ikke* bare lader sig stoppe fra den ene dag til den anden. Erfaringerne viser, at kravene *vil* udvikle sig løbende – også efter kravspecifikationen er færdig.

4.4 Misforståelser

Kravspecifikationen er et forsøg på at beskrive og fastholde aftaler, der er indgået tidligt i projektet om hvad systemet skal kunne. I starten af projektet læses og *fortolkes* kravspecifikationen af projektets interessenter, som – baseret på deres fortolkning – accepterer den som udgangspunkt for projektet. Senere læses og *fortolkes* kravspecifikationen igen af projektets arkitekter og udviklere, som – baseret på deres fortolkning – udvikler systemet.
Erfaringerne viser, at de to fortolkninger giver anledning til misforståelser. I nogle projekter til mindre misforståelser, som kan udredes i mindelighed. I andre projekter til større misforståelser, som betyder at projektet får store problemer – og i værste fald må lukkes uden at have leveret noget.

Sammenfattende kan vi konstatere, at der kun er to ting, vi kan være sikre på omkring krav til IT systemer:

- Kravene vil ændre sig
- Kravene vil blive misforstået

Dette er stærkt problematisk – specielt hvis vi lader kravspecifikationen være udgangspunkt for alle aftaler, kontrakter, planer og beslutninger i projektet.

Men der er flere problemer:

4.5 Svært at motivere

Store vandfaldsorienterede projekter strækker sig ofter over meget lang tid. Projekter med en varighed på op til 12 eller 18 måneder er ikke usædvanligt. Selvom der er faser og deadlines undervejs, så kan det være svært at fastholde fokus og engagement over så lang en periode.

4.6 Teknologien ændrer sig

Ofte begynder man ret tidligt i et projekt at tænke over hvilken teknologisk platform man skal basere sit system på. Skal det være J2EE eller .NET. Eller hvad med noget Open Source?
For nogle år tilbage var det OK at træffe beslutning om teknologi og arkitektur relativt tidligt i projektet, for dengang ændrede teknologien sig ikke så hurtigt.
I de senere år har der været en tendens til, at teknologien udvikler sig hurtigere og hurtigere. Derfor giver tidlige og fastlåste beslutninger om teknologi problemer – for beslutningerne bliver udfordret af nye teknologier, hurtigt efter de er truffet. De beslutninger man har taget om teknologi vil blive forsøgt fastholdt – også selvom der kommer nye og bedre teknologier.

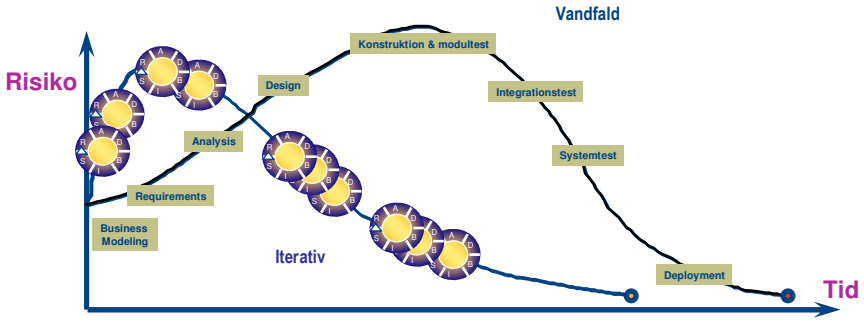
4.7 Arkitekturen bygges på antagelser

I et vandfaldsprojekt beskrives den teknologiske platform, arkitekturen og designet for hele systemet på en gang. Ideen er, at lave et design, som tager højde for alle kravene, så det ikke skal ændres efterfølgende. Først når arbejdet med at beskrive arkitekturen og designet er færdiggjort, så går IT arkitekterne og udviklerne i gang med at konstruere arkitekturen og systemet. Ikke helt sjældent får man nogle grimme overraskelser under konstruktionen og implementeringen af systemet, når man arbejder på denne måde. Det skyldes, at arkitektur og design – hvor gennemtænkt det end måtte være – trods alt bygger på nogle forudsætninger og nogle antagelser, som man har gjort sig under beskrivelsen af arkitektur og design. Nogle antagelser, som først bliver checket op mod den barske teknologiske virkelighed meget sent i projektet.

4.8 Stor risiko

I et vandfaldsprojekt har man ikke sikkerhed for, at få noget ud af sin investering før til allersidst, når systemet er helt færdigt og leveret. Hvis noget går galt undervejs, så kan man risikere at miste alt det, man har investeret i projektet. Dette er en reel risiko, som ikke helt sjældent har resulteret i, at man har investeret en masse penge uden af få noget ud af det. Eneste udbytte har været en kravspecifikation, et design, eller et system som, kun er delvist færdigt.

Risiko i iterative projekter og vandfaldsprojekter



- Høj-risiko problemer angribes tidligt i iterative projekter hvorved risici elimineres tidligt
- I vandfaldsprojekter forsvinder risikoen først til allersidst, når systemet ér implementeret og kører i drift

Accelerated Delivery Center

CAP GEMINI

5. Løsningen: Letvægtsorienteret og iterativ udvikling

Flere og flere har løst ovenstående problemer ved at ændre deres traditionelle vandfaldsorienterede angrebsvinkel til en mere *letvægtsorienteret* og *iterativ* udviklingsproces.

Det *letvægtsorienterede* består i, at man bruger mindre tid på "udenomsaktiviteter" som f.eks. mødereferater, lange og komplette kravspecifikationer, renskrivning af modeller, etc. – og mere tid på egentlige "systemudviklingsaktiviteter" som f.eks. design og programmering af systemet. I et letvægtsorienteret projekt med færre dokumenter er der brug for en tættere kontakt mellem brugere og udviklere, når detaljer skal undersøges og aftales. Den daglige tætte kontakt er yderst gavnende for kommunikationen og samarbejdsklimaet i gruppen – og erfaringerne viser, at vi får bugt med misforståelserne.

Det *iterative* består i, at man – efter en kort og overordnet analyse – opdeler hele systemet i nogle mindre bidder – og at man så udvikler én bid ad gangen i "time boxes" af mellem 1 og 3 måneders varighed. Under opdelingen af systemet i bidder skal man sikre sig, at hver enkelt bid kan implementeres – og give forretningsmæssig værdi – hver for sig. Ofte vælger man at opdele systemet efter "use cases", fordi en use case netop er karakteriseret ved, at den giver en del af systemets brugere en forretningsmæssig værdi.

Det er uhyre vigtigt, at få startet rigtigt. Det viser sig, at følgende angrebsvinkel i starten af et projekt danner det bedst mulige udgangspunkt for resten af projektet:

- Brug så meget tid sammen med kunden som muligt
- Få en overordnet forståelse for kundens behov
- Dan udgangspunkt for en aftale om tid og økonomi
- Tegn og skriv sammen med kunden – undlad at aftale ting på møder, som efterfølgende skal dokumenteres af en af parterne. Sid sammen og dokumentér.
- Prioriter hvad der er vigtigst – hvad der skal laves først
- Opdel udviklingen af systemet i nogle mindre bidder, der kan udvikles, implementeres og give forretningsmæssig værdi, hver for sig.

6. Fordele ved iterativ udvikling

6.1 Motivation

Det er meget lettere at skabe fokus og engagement i et iterativ projekt. Det er meget lettere ved at forholde sig til et mål som ligger 1 til 3 måneder ud i fremtiden, end det er at forholde sig til et mål, som skal være opfyldt "engang til næste år". I vandfaldsprojekter kan man ganske vist indføre nogle milepæle for hver fase, men erfaringerne viser, at det er langt mere motiverende at skulle levere en fuldt færdig del af systemet, end at skulle have en eller anden analyse- eller designrapport færdig.

6.2 Overskueligt

Problemet med, at det er svært at overskue hele systemet på én gang løses ved, at vi ikke forsøger at overskue hele systemet – med alle dets detaljer – på én gang. Først når vi skal i gang med at udvikle den bid af systemet, som vi nu engang er nået til, begynder vi at sætte os ind i detaljerne – og kun detaljerne for denne ene bid. På denne måde udarbejdes detailspecifikationer ganske kort tid før de skal bruges. Det betyder at behovet for afklaringer føles således meget nærværende – og det betyder at forretningsregler, diskussioner og beslutninger er i frisk erindring blandt udviklerne, når de skal implementeres i programkode.

6.3 Omgivelserne og behovene udvikler sig – projektet følger med

Problemet med at omgivelserne kan ændre sig – og problemet med at brugernes kreativitet genererer flere gode ideer, løser vi ved at checke om projektet er på ret kurs, hver gang vi er færdige med en iteration. Hvis det viser sig at være ønskeligt fra kunden side, så ændrer vi på projektets prioriteringer – og på indhold og plan for de næste iterationer. På dette tidspunkt i projektet – lige når vi har færdiggjort og leveret en del af systemet – føles det helt naturligt, at tage sådanne spørgsmål op til overvejelse. Det giver ikke de store problemer i iterative projekter, for der har vi ikke bundet en masse tid og energi i at undersøge alting fra starten af: Når vi har brugt mindre tid på at forstå og beskrive detaljerne, som de så ud i starten af projektet, så betyder det mindre – eller måske ligefrem ingenting – at detaljerne ændrer sig.

6.4 Oplevelsen af systemet giver klarsyn

Det er en helt almindelig oplevelse, at brugerne først for alvor kan forholde sig til systemet, når de ser det færdige resultat. I et iterativt projekt giver vi brugerne mulighed for at forholde sig til systemet adskillige gange i løbet af projektet. Det giver ret tidligt i projektet et klarere syn på systemet og hvad der reelt er behov for. Dette er selvsagt af allerstørste vigtighed for ethvert udviklingsprojekt.

6.5 Misforståelse erstattes af forståelse

Problemet med misforståelser løser vi i iterative projekter ved at bruge mere tid og energi på at forstå brugerne, udvikle systemet og vise det til brugerne – end vi bruger på at skrive om systemet. Det er jo kravspecifikationen med alle dens beskrivelser, der er udgangspunktet for fortolkninger og misforståelser. I iterativ udvikling sætter vi dialog og forståelse højere end en lang og detaljeret kravspecifikation. Det giver selv sagt færre misforståelser. Når vi oven i købet har mulighed for at implementere den første del af systemet tidligt i projektet – og "udsætte" det for en rigtig forretningsmæssig og driftsmæssig virkelighed, så får vi også checket, at brugernes opfattelse af den virkelighed, som systemet skal køre i, er korrekt. Eventuelle misforståelser her opdages tidligt i de første iterationer, og kan udredes tidligt inden de får større betydning for projektet.

6.6 Teknologien ændrer sig – projektet følger med

Når teknologien ændrer sig så meget, at vi bliver nødt til at reagere på det i vores projekter, så giver det altid nogle udfordringer – også i iterative projekter. For en udskiftning af teknologisk platform var næppe med i de oprindelige estimater og priser. I et vandfaldsprojekt befinder man sig lang tid i en fase, hvor man har et stort system, der er under opbygning. I løbet af den fase er det yderst problematisk at skifte teknologi, for så bortfalder hele grundlaget for vandfaldsprojektet – nemlig at man fase for fase fastlægger nogle ting, som man så bygger videre på i næste fase. I et iterativt projekt derimod har vi indbygget nogle naturlige tidspunkter, hvor vi checker, om vi er på rette vej. Hvis der bliver truffet beslutning om at skifte teknologi, så har vi et langt bedre udgangspunkt end i et vandfaldsprojekt, for vi har en del af systemet, der er 100% færdigt. Vi kan beslutte, at den færdige del af systemet kører videre på den gamle arkitektur mens vi udvikler resten af systemet på en ny arkitektur. Eller vi kan beslutte os for at konvertere den færdige del af systemet – og så udskyde udviklingen af resten af systemet. Uanset hvad vi beslutter os for, så har vi den klare fordel i et iterativt projekt, at vi har en del af systemet, der er færdigt og implementeret, og som dagligt giver en forretningsmæssig værdi til kunden.

6.7 Arkitekturen bygger på viden og erfaringer

I et iterativt projekt vil vi – ligesom i vandfaldsprojekter – træffe nogle beslutninger om teknologi og arkitektur meget tidligt i projektet. Men i et iterativt projekt vil vi efterprøve vores beslutninger inden for ganske kort tid – typisk nogle få uger. Længere kan vi ikke vente, for vi skal jo implementere den første del af systemet efter højst 3 måneder. Det giver den klare fordel, at eventuelle forkerte beslutninger bliver identificeret og korrigeret tidligt i projektet, så vi ikke bygger videre på forkerte antagelser.

6.8 Lav risiko

Ligesom i et vandfaldsprojekt investeres det betydelige ressourcer i iterative projekter. På investeringssiden er der således ikke den store forskel. Forskellen ligger i sikkerheden for, om man får udbytte af sin investering. I et iterativ projekt udvikles og implementeres systemet stepvist. Ret hurtigt har man således et kørende system, som giver en forretningsmæssig værdi. Efterhånden som iterationerne skrider frem implementeres mere og mere af systemet, som på den måde giver en større og større værdi. Risikoen for at miste hele sin investering er – i en iterativ verden – ikke eksisterende. Selv hvis kunden og leverandøren skulle beslutte sig for at afslutte samarbejdet midt i projektet, så har kunden fået den del af systemet, som han har betalt for – og kan vælge at bede en anden leverandør om at gøre projektet færdigt.
Selv i situationer, hvor et projekt fejler og må stoppes er risikoen i et iterativ projekt mindre. Det skyldes at man som hovedregel laver det sværeste – det mest risikable først. Herefter prøver man at implementere det – og får skabt vished for om det lykkes. Hvis det viser sig, at det ikke kan lykkes, så får vi identificeret det tidligt i projektet, og kan lukke projektet i en fart, før vi får investeret en masse penge i noget, der alligevel aldrig ville komme til at virke efter hensigten.

7. Hvad skal der til, for at indføre iterativ udvikling

Indførsel af iterativ udvikling i en organisation er ikke nogen helt lille opgave. Specielt de første projekter er svære at få startet. Her er nogle stikord til hvad der skal til for at komme godt i gang:

- Start med ét projekt – gerne strategisk
- Håndpluk projektlederen – skal være en delegerende og stærk leder
- Håndpluk projektdeltagerne
- Forklar kunde/sponsor/styregruppe fordelene ved iterativ udvikling – og lad dem vide, at det er en ledelsesmæssig beslutning at køre projektet iterativt
- Giv projektlederen opbakning og coach-støtte i det omfang, han ønsker det – og lad medarbejderne forstå, at også de forventes at tage imod coaching…
- Vurdér fremdrift ved at se systemet vokse bid for bid. Undlad at bruge en omfattende formél rapportering. Det hindrer projektlederen i at skabe fokus og fremdrift.

Når beslutningen om at køre det første iterative projekt er truffet, så skal man være klar over, at succes'en er stærkt afhængig af en stor indsats fra mange lag i organisationen. Her er en liste over de vigtigste indsatser, der kræves af virksomhedens ledelse, projektlederne og projektgruppens medlemmer:

7.1 Hvad kræves der af ledelsen

- Stille op med engagerede ledere/sponsorer/brugere, som forlanger at se resultater (færdigt software) efter hver iteration
- Allokere medarbejdere med den rette viden og beslutningskraft til aktivt at deltage i projektet
- Opfat de første par iterationer som en investering
- Bakke op omkring iterativ udvikling, så medarbejderne ved, at det ikke bare en projektlederens "egen fikse ide"
- Acceptér (og forvent) at grovplanerne ændres løbende
- Tilrettelæg projekterne, så de tilknyttede medarbejdere ikke har andre opgaver samtidig ("man kan kun brænde for én ting ad gangen")
- Tilvejebring lokaler, hvor hele projektgruppen og projektleder kan sidde sammen.

7.2 Hvad kræves der er projektlederne

- Planlæg udvikling i iterationer (1-3 måneder) og implementér efter hver iteration
- Arranger høj synlighed (=succes-oplevelse) ved færdiggørelse af iterationer
- Indfør løbende prioritering med bruger – for hver iteration
- Invitér gruppen til – sammen med brugeren – at udarbejde grovplan, og hav den på en whiteboard
- Invitér gruppen til at komme med input til den løbende revidering af grovplanen

- Tilbyd projektgruppen, at de selv kan tilrettelægge hvordan de vil arbejde – men fasthold kravet om iterationer

7.3 Projektdeltagerne skal committe sig…

- til at deltage aktivt i planlægningsprocessen
- til at tage ansvar for færdiggørelse af iterationerne – til tiden – (evt. på bekostning af funktionalitet) og vente med funktionalitet, der ligger i en senere iteration.
- til at acceptere og være imødekommende, når brugerne bliver klogere
- til at medvirke til synlighed – ved at levere "noget synligt" system
- til at tage et medansvar for tilrettelæggelse af projektet

8. Faldgruber i iterativ udvikling

Selvom den iterative udviklingsprojekter byder på langt færre problemer end vandfaldsprojekter, så er der nu alligevel en række udfordringer, som man skal forholde sig til. Men udfordringerne er ligesom faldgruber: Hvis man ved hvor faldgruberne er, så er de ikke så svære at undgå:

8.1 Alle krav specificeres fra starten

Faldgrube: Selv om man egentlig har besluttet, at køre iterativt, så får vanens magt alligevel både kunder og udviklere til at prøve at specificere alle krav up front. Dette kan forsinke/umuliggøre, at man får startet på den første egentlige iteration.
Løsning: Gå ind i projektet – og hjælp med at beslutte, at den indledende og overordnede krav-analyse er færdig – og understøt de beslutninger, der skal til for at få fordelt/udskudt resten af specifikationsarbejdet til de enkelte iterationer.

8.2 Detailspecifikationer opfattes som ændringer

Faldgrube Når man graver sig ned i detaljerne i den enkelte iteration, så opdager man ofte, at noget var mere kompliceret end oprindeligt antaget. Dette opfatter mange udviklingsteams som "ændringer" til den oprindelige specifikation.
Løsning: Se under "Ændringer afvises"

8.3 Ændringer afvises

Faldgrube: Selvom iterativ udvikling netop skal understøtte, at det er tilladt for både kunde og udviklere at være kreative og blive klogere undervejs, så får vanens magt alligevel mange udviklingsteams til at reagere negativt på de ændringer, som dette medfører.
Løsning: Indskærp, at det er kunden, der betaler for udviklingen af systemet – og at det derfor også er kundens beslutning, hvad der skal udvikles – og i hvilken rækkefølge.

8.4 ”Et par uger mere”

Faldgrube: Tæt på en iterations afslutning beder udviklerne om at få mere tid til iterationen – fordi de ikke kan nå at blive helt færdige. Man giver dem lidt mere tid – og få dage inden de skulle være færdige gentager historien sig… Jo mere ekstra tid man giver – jo oftere oplever man, at diskussioner, der var afsluttet , bliver bragt til live igen. Eller at der opstår usikkerhed omkring arkitekturen. Eller at brugergrænsefladen måske alligevel ikke er i orden. Eller…
Løsning: Bed om en liste over de ting, der ikke kan nås indenfor den fastsatte tid, og hold fast i, at der skal leveres noget færdigt – men gerne mindre omfangsrigt – software, til den aftalte tid.

8.5 Alt i første iteration

Faldgrube: Arkitekter/udviklerne forsøger – p.g.a. vanens magt – at lave fremtidssikrede arkitektur/løsninger på alting fra starten af. Dette er en nærmest umulig opgave, da alle kravene jo ikke kendes fra starten. En udgang på dette kan være, at arkitekturen baseres på formodninger og gætterier om hvad fremtiden vil bringe. En anden udgang kan være, at man – alligevel – forsøger, at få defineret krav fra starten af.
Løsning: Kommunikér igen og igen, at målet er at implementere tingene gradvist – og at vi først løser eventuelle problemer, når vi står direkte overfor dem i de enkelte iterationer – og at det er OK og mere effektivt, at forbedre arkitektur og design gradvist.

8.6 Systemet sander til

Faldgrube: Der bliver ikke foretaget de nødvendige restruktureringer af arkitektur og design, efterhånden som der kræves mere og mere – i takt med at der implementeres mere og mere funktionalitet. I et iterativt projekt gennemføres der flere ændringer i de enkelte programmer end ellers. Derfor kan kvaliteten af koden og arkitekturen i systemet hurtigt sande til. Det medfører, at nye ting bliver sværere og sværere at implementere.
Løsning: Indfør nødvendige omstruktureringer af koden (”refactoring”), som en naturlig forberedelse til at kunne videreudvikle og udvide med mere funktionalitet. Indfør evt. også automatisk test og automatisk regressionstest, for at forhindre at omstruktureringer af koden ikke får utilsigtede bi-virkninger.

9. Afslutning

Måske sidder du og tænker:

- ”Er det virkelig nødvendigt at indføre iterativ udvikling i min organisation…?”
- ”Er det ikke for besværligt at ændre på den nuværende forholdsvis velfungerende udviklingsproces…?”
- ”Hvad hvis jeg ofrer en masse energi og ressourcer – og det så bare viser sig at iterativ udvikling bare er en døgnflue blandt så mange andre døgnfluer inden for systemudvikling…?”
- ”Er der andre, der har indset fordelene ved iterativ udvikling – og som har indført det i deres virksomheder…?”

For at underbygge, at der er noget om snakken – at iterativ udvikling giver færre problemer end vandfaldsprojekter – vil jeg her præsentere et lille udpluk af nogle meget betydningsfulde ting der er sket omkring systemudvikling i de senere år:

- Rational Software har etableret Rational Unified Process (RUP), som er imponerende samling af best practices inden for systemudvikling. RUP foreskriver, at man organiserer sit projekt efter faserne Inception, Elaboration, Construction og Transition. Det kan lyde vandfaldsorienteret, men er det ikke, for RUP foreskriver også, at man arbejder iterativt – primært for at mindske projektets risiko.

- Flere og flere virksomheder opgiver at videreudvikle på deres egne udviklingsmodeller, og tager RUP – og således også iterativ udvikling – i anvendelse.

- IBM har gennem sit opkøb af Rational Software vist, at de tror på RUP og dermed også på iterativ udvikling.

- Kent Becks tanker om eXtreme Programming (XP) er blevet meget vel modtaget i systemudviklerkredse. XP bygger på værdierne Kommunikation, Enkelthed, Feedback og Mod. Derudover bygger XP på 12 praktikker som f.eks. On-site customer, Pair programming og Small releases. Den sidste er et synonym for iterativ udvikling, som altså også hyldes af XP.

- En række førende specialister inden for systemudvikling etablerede for et par år siden Agile Alliance, som får stadigt flere medlemmer, og som har interessegrupper i flere og flere lande. Fundamentet for Agile bevægelsen er manifestet, som lyder således:

  We are uncovering better ways of developing
  software by doing it and helping others do it.
  Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
That is, while there is value in the items on
the right, we value the items on the left more.

- En stor konsulentvirksomhed som Cap Gemini har investeret betydelige ressourcer i opbygningen af en række centre, som er specielt designet til at gennemføre iterativ udvikling i et tæt samarbejde med kunder. Centrene, der er navngivet Accellerated Delivery Center (ADC), er foreløbig etableret i 30 lande – og flere kommer til.