Smidig utvikling med PS2000

Veileder

med fokus på hjelp til begge parter for å gjennomføre utviklingsprosjekter basert på PS2000 og smidig systemutviklingsmetodikk

DEN NORSKE DATAFORENING

Ver.	:	1.0
Dato oppdatert	:	10.11.2008

INNHOLDSFORTEGNELSE

FIGURFORTEGNELSE

1 Innledning

1.1 Bakgrunn

Smidige metoder benyttes i stadig større utstrekning for store systemutviklingsprosjekter, både i privat og offentlig sektor. Samtidig opplever de fleste kunder og leverandører store utfordringer med å benytte standardiserte leveransekontrakter i prosjekter som skal følge smidig metodikk. De fleste kontraktsstandarder forutsetter en detaljert kravspesifikasjon, og legger opp til en risikofylt fossefallsorientert prosjektmetodikk, fremfor en mer endringsorientert smidig metodikk.

Konsekvensen er i praksis at majoriteten av smidige utviklingsprosjekter gjennomføres på bakgrunn av timebaserte bistandsavtaler der kunden selv tar resultatansvaret. Vi mener at slike avtaler kan være det riktige utgangspunktet for en rekke prosjekter som ønskes gjennomført i henhold til smidig metodikk. Men det finnes situasjoner der IT-prosjektene likevel ønsker eller må ta i bruk leveransekontrakt:

- Kunden ønsker å dele resultatansvar og økonomisk risiko med leverandøren
- Formelle krav til kontraktsregulering og anbudsprosesser

PS2000 er en kontraktsstandard som er basert på iterativ prosjektgjennomføring, og anses derfor av mange som den kontraktsstandarden som best lar seg kombinere med smidig metodikk. Med bakgrunn i den økende utbredelsen av PS2000 som foretrukket kontraktsstandard innenfor både offentlig og privat sektor, har Dataforeningens faggruppe for IT-kontrakter nedsatt en arbeidsgruppe som skulle undersøke om PS2000 kontraktsstandarden kan brukes til å gjennomføre prosjektene etter smidig metodikk. Samtidig ønsket man å publisere erfaringer og beste praksis for hvordan PS2000 kan benyttes som en formell leveransekontrakt, samtidig som man får ta del i de fordeler som ligger i smidig gjennomføring. Resultatet av dette arbeidet er denne veilederen.

1.2 Arbeidsgruppens deltakere

Arbeidsgruppen har i henhold til Dataforeningens prinsipper om balansert utarbeidelse av standardkontrakter vært sammensatt av både representanter både fra kunder, leverandører og tredjeparts rådgivere:

- Erik Bollestad og Jan Raae fra FLO/IKT, Forsvaret
- Jan Erik Ressem fra Statens lånekasse for utdanning
- Inger Beate Botheim fra NAV
- Frithjof Frederiksen fra Bekk Consulting
- Lars Frode Haugen fra Computas
- Bodil Rabben fra Capgemini
- Simen Fure Jørgensen fra Iterate
- Peter Hidas fra Gartner
- Kjetil Moløkken-Østvold fra Conceptos Consulting
- Arbeidsgruppen har vært ledet av PROMIS ved Kjetil Strand.

Veilederen har blitt sendt til et utvalg av enkeltpersoner og firmaer for høring. Følgende personer har gitt gode innspill underveis:

- Anne-Kathrine Pihlfeldt fra Avenir
- Geir Amsjø fra Spitia
- Marit Søholt Stokes fra Accent AS
- Johannes Brodwall og Knut Magnar Nilsen fra Steria
- Lars Ivar Næss fra Know IT ObjectNet
- Kristin Halvorsen fra ObjectWare
- Niklas Bjørnerstedt fra LeanWay
- Sigrun Kongsrud fra Computas
- Jørgen Petersen, Odd Gunnar Alterhaug, Kjetil Karlsen og Trond Åsheim fra Promis

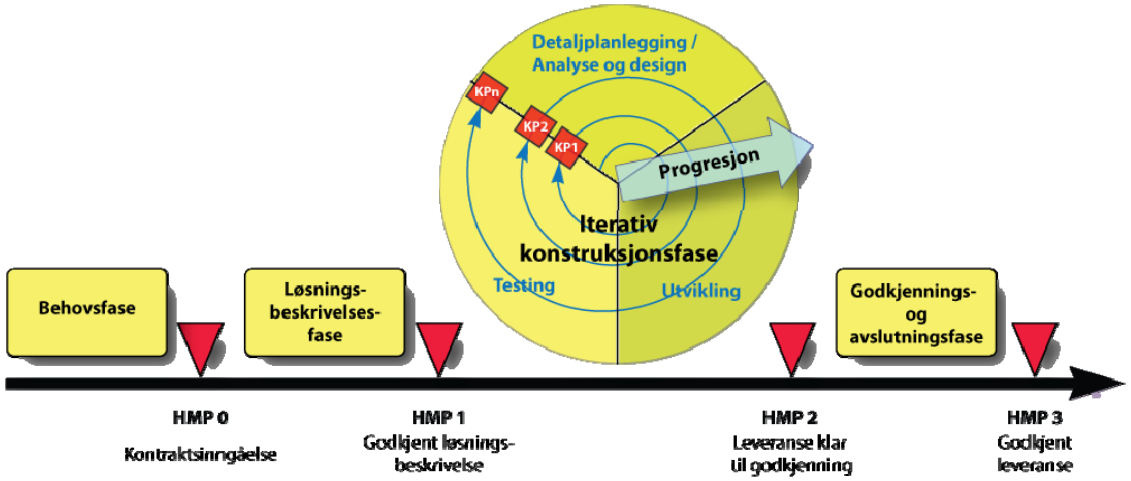
1.3 Konsekvenser for kontraktsteksten

Mandatet til arbeidsgruppen var å undersøke om PS2000 kontraktsstandarden kan brukes til å gjennomføre prosjektene etter smidig metodikk. Arbeidsgruppen mener at PS2000 kan legges til grunn uten at det skal være nødvendig å endre i selve kontraktsteksten. Vi peker på to mulige endringer, som kan innarbeides i kontraktens Del I. Det ene går på gjennomføring av kontrollpunktet og godkjenning av neste iterasjon (se kapittel 5.6), mens det andre går på håndtering av endringer som verken har kostnadsmessig eller tidsmessig konsekvens (se kapittel 5.8). Etter vår oppfatning skal det imidlertid være fullt mulig å kjøre smidig også i henhold til den opprinnelige kontraktsteksten.

1.4 Leserveiledning

Kapittel 2 gir en kort motivasjon for å benytte smidig metodikk, noen begrepsavklaringer samt en gjennomgang av de viktigste rollene i smidige prosjekter.

Kapitlene 3 til 7 gir en systematisk gjennomgang av hovedfasene i et PS2000-prosjekt, slik de er gjengitt i figur 1-1. Det gis fortløpende anbefalinger for hvordan PS2000-fasene kan gjennomføres i henhold til smidig metodikk, men samtidig uten å forrykke balansen i PS2000-kontrakten.



Figur 1-1: Ulike faser i et PS2000 basert prosjekt

Veilederen baserer seg gjennomgående på Scrum som den underliggende smidige metodikk. Scrum har etablert seg som den mest utbredte varianten av smidige metoder innenfor større utviklingsprosjekter i Norge og internasjonalt. Scrum er blant de best dokumenterte og helhetlige metodekonseptene på området. Som metoderammeverk har Scrum et dokumentasjonsnivå og et omfang som egner seg godt sammen med et kontraktsrammeverk som PS2000.

Vi har fritt latt oss påvirke av andre tilgrensende metodekonsepter, som for eksempel Lean, eXtreme Programming (XP), DSDM, EVO og Test-Driven Development (TDD) (se vedlegg, kapittel 4).

Veilederens vedlegg gir utfyllende bakgrunnsinformasjon om de mest kjente smidige metodene, samt utfyllende detaljer knyttet til scrum.

Denne veilederen er ikke å anse som en innføringsbok i smidig metodikk. Leseren henvises til aktuell litteratur på området, se blant annet litteraturanbefalinger gitt i vedleggene, kapittel 8.

Veilederen er et tillegg til den generelle veilederen til PS2000 kontrakten

2 Smidige metoder

Agile Alliance ble etablert i februar 2001, da sytten metodeeksperter kom sammen i Utah, USA og formulerte ’The Agile Manifesto’. ’Agile’ har blitt oversatt til smidig på norsk. Dette er et godt begrep, men man skal være klar over at Agile også har et bredere meningsinnhold på engelsk. I Agile ligger - i tillegg til smidig - det å være på hugget, hurtighet.

2.1 Hvorfor benytte smidige metoder

I følge norske og internasjonale studier sliter programvareutvikling fortsatt med budsjettoverskridelser, forsinkelser og dårlig kvalitet på samme nivå som for 30-40 år siden [1, 2]. Imidlertid er det funn som tyder på at smidige utviklingsmetodikker kan bidra til å redusere overskridelser, blant annet fordi man gjennom slike metoder fasiliteter samarbeid mellom kunde og leverandør [3].

Kjernen innenfor smidig utvikling er det smidige manifestet[1]. Uavhengig av hva slags type smidig utviklingsmetodikk som omtales, så er de grunnleggende prinsippene:

1) **Individer og samspill** framfor prosesser og verktøy
2) **Fungerende system** framfor utførlig dokumentasjon
3) **Samarbeid med kunden** framfor kontraktsforhandlinger
4) **Å reagere på endringer** framfor å følge en plan

Selv om det som står på høyre side oppfattes som verdifullt, verdsettes det på venstre side høyere.

Det finnes etter hvert en rekke erfaringsrapporter som viser det store potensialet ved smidige metoder. Påstanden er at man med smidige metoder kan oppnå:

- Større grad av tilpasning til forretningsmessige behov
- Større treffsikkerhet i forhold til sluttbrukernes reelle behov
- Mer effektiv produktutvikling
- Bedre kontroll med prosjektparametre som tid, kostnader, fremdrift og risiko
- Høyere kvalitet
- Kraftige besparelser i de tidlige fasene av prosjektet

En studie [18] viser hvordan Systematic, et CMMI level 5 selskap i USA, har klare interne målinger på store forbedringer, selv om de allerede var svært gode. Systematic og Trifork rapporterer om hvordan de nå har vunnet smidige kontrakter basert på følgende hovedregler:

- Endringer er gratis på ikke påbegynt funksjonalitet
- Endringer i prioritet er gratis - hvis ikke totalen endrer seg
- Nye funksjoner kan legges inn underveis, så lenge funksjoner av tilsvarende størrelse tas ut
- Kunden kan avbryte kontrakten når som helst. I så tilfelle skal 20 % av gjenstående kontraktssum tilfalle leverandøren[2]
- Det kreves av kunden at alle behov/krav må prioriteres
- Kunden sørger for at representative brukere følger prosjektet aktivt underveis og at disse er utstyrt med tilstrekkelig ansvar og myndighet til å kunne gi nyttig feedback og forøvrig bidra uten å sinke prosjektet

2.2 Begrepsdefinisjoner

Med utgangspunkt i Scrum og tilgrensende konsepter har vi lagt vekt på å bruke norske uttrykk for roller, aktiviteter og artefakter, der dette er mulig.

Et av de mest sentrale begrepene i Scrum er sprint. En sprint er navnet på en iterasjon i utviklingsløpet. Vi anbefaler å sette likhetstegn mellom sprint og begrepet ’trinn’ som benyttes i PS2000. Begrepet sprint brukes derfor gjennomgående i denne veilederen.Tabellen

1 http://www.agilemanifesto.org/

2 I veiledningen til PS2000 er anbefalt et avbestillingsgebyr på 4-6%

under viser de oversettelsene som er brukt i denne veilederen og de tilhørende engelske begrep som eksisterer i ulik litteratur på området.

Uttrykk i denne veilederen	Uttrykk i engelsk litteratur om Scrum
Brenndiagram	Burndown Chart
Daglig scrum	Daily Scrum
Interessenter	Stakeholders
Produkteier	Product Owner
Produktkø	Product Backlog
Scrumleder	Scrum Master
Sprintdemo	Sprint demo
Sprintkø	Sprint Backlog
Sprintplanlegging	Sprint planning
Sprintteam	Team
Sprinttilbakeblikk	Sprint Retrospective
Fart	Velocity
Brukerhistorie	User story
Brukstilfelle	Use case

2.3 Roller

Rollene som beskrives i dette avsnittet bør innarbeides av partene i bilag B i PS2000.

Scrum har tre klart definerte roller: produkteier, scrumleder og sprintteam. Veilederen baserer seg på disse tre rollene, i tillegg til at man introduserer et produktteam.

En av de viktigste egenskapene med en smidig tilnærming er praksisen med å gjøre brukere, forretningssiden og andre interessenter delaktige i prosessen. Det er kritisk at de er engasjerte og gir tilbakemelding på det sprintteamet produserer, blant annet ved å delta på sprintdemo og ved å teste nye funksjoner i produktet etter hvert som de ferdigstilles. Dette kundeansvaret må gå tydelig frem av bilag B. Roller, ansvar og myndighet må være klart beskrevet og det bør klart fremgå på hvilken måte tilbakemelding skal gis.

I det videre er de viktigste rollene beskrevet. Ytterligere detaljer og informasjon om roller og interessenter er presentert i kapittel 5 i vedleggene.

2.3.1 Produkteier

Produkteier er den viktigste rollen i smidig gjennomføring. Dette er en rolle på kundesiden, som først og fremst har ansvaret for å oppdatere produktkøen underveis og før hver ny sprint.

I mindre PS2000-prosjekter kan rollen produkteier sammenfalle med rollen som kundens prosjektleder. I de fleste prosjektene vi kjenner til, er imidlertid disse rollene fordelt på to

ressurser, slik at produkteier kan konsentrere seg fullt og helt om produktkøen, mens kundens prosjektleder følger opp kundens forpliktelser i henhold til kontrakten, inkludert rapportering til koordineringsgruppen.

2.3.2 Produktteamet

Vi har erfart at produkteier trenger ressurser rundt seg for å kunne forberede produktkøen i forkant av sprintene, og i bearbeiding og prioritering av produktelementer. For dette formålet vil vi anbefale at det etableres et produktteam med medlemmer fra både kundenkunden og leverandøren.

2.3.3 Sprintteamet

Sprintteamet består av 7, pluss/minus 2 medlemmer og er tverrfaglig sammensatt: utvikling, arkitekturkompetanse, testing og dokumentasjon. I større prosjekter må det etableres flere sprintteam.

2.3.4 Scrumleder

En scrumleder er en leder og fasilitator for sprintteamet og ansvarlig for å få til et velfungerende team som har høyest mulig produktivitet.

I små prosjekter kan rollen som scrumleder og leverandørens prosjektleder sammenfalle. Men der det er flere enn ett sprintteam, vil disse rollene i praksis være delt på flere ressurser. Vi har gode erfaringer med at leverandørens prosjektleder i slike prosjekter er ansvarlig for å kjøre 'Scrum of Scrums', der scrumleder for hvert sprintteam deltar i en daglig scrum. Dette møtet holdes gjerne litt i etterkant av sprintteamenes daglige scrum, slik at scrumlederne har fått justert brenndiagram og håndtert eventuelle hindringer.

3 Behovsfasen

Før kunden går i gang med å utarbeide et konkurransegrunnlag basert på en behovsanalyse bør det utarbeides en strategi, som blant annet tar stilling til valg av kontraktsform og gjennomføringsmetode. Et bevisst forhold til dette kan gi gevinster i form av en mer effektiv prosess. Dersom det overlates for mange valg til leverandørene som en del av konkurransen, må kunden være klar over at tilbudene kan bli vanskelig å sammenligne og at man ikke setter seg selv i stand til å velge den leverandøren som passer kunden og behovet best.

Smidig utviklingsmetodikk er generelt godt egnet i de fleste tilfellene der PS2000 er et naturlig valg som kontraktsform, for eksempel når det ikke er hensiktsmessig eller mulig å på forhånd fastsette nøyaktige eller detaljerte spesifikasjoner. Smidig metodikk er ofte velegnet for offentlig sektor når man ikke kan eller ikke ønsker å lage en detaljert kravspesifikasjon i konkurransesituasjonen. Det er også velegnet med smidig metodikk i privat sektor når man finner det hensiktsmessig, for eksempel når kundens kravforståelse er liten eller der forretningsprosessene er i kraftig endring og det er gjennom prosjektet at kravene blir avklart, prioritert og realisert.

Smidige metoder stiller imidlertid store krav til partene, og ikke minst kunden. Disse kravene kan mange ha vanskelig for å akseptere eller følge på en effektiv måte. Den smidige modellen lar seg ikke kombinere med det tradisjonelle scenariet der kunden starter med å fortelle leverandøren hva som skal leveres og deretter overlater til leverandøren å utføre arbeidet frem til at resultatet kan testes. Den smidige modellen krever tett, aktivt, dag-til-dag samarbeid mellom de som skal gjøre utviklingen (leverandøren) og de som skal bringe domenekunnskapene og prioriteringen til prosjektet (kunden). Kunden må avsette noen av sine beste, mest erfarne folk til utviklingsprosjektet, og gi dem tilstrekkelig myndighet slik at arbeidet kan skride frem uten forsinkelser. Det er viktig at kunden er villig til dette for å lykkes med smidig gjennomføring.

Et tredje kriterium for å lykkes med smidig metodikk, er at partene er innforstått med den prosessen som skal legges til grunn, rollene som inngår og ansvarsfordelingen dem imellom. Det er derfor viktig å stille krav til leverandørene at de kan dokumentere erfaring fra prosjekter der tilsvarende metodikk er lagt til grunn. Kunden kan bruke referanseinnhenting som ett av kriteriene for tildeling. Man kan også legge inn caseoppgaver i forhandlingsfasen for å teste leverandørenes evne til smidig prosessledelse. PS2000 gir rom for avbestilling etter løsningsbeskrivelsesfasen og dette kan fungere som en endelig prøve av leverandøren.

3.1 Hvorfor behovsfasen?

I behovsfasen skal kunden analysere og spesifisere formålene med og kravene til leveransen. I denne fasen legges grunnlaget for prosjektets smidighet. Enkelte krav bør klargjøres tidlig, mens andre krav har best av å vente. Man mister mye av det rommet man trenger for å kunne gjennomføre prosjektet smidig, dersom kunden går for langt i å detaljspesifisere krav til leveransen og tekniske spesifikasjoner. I tillegg blir mye arbeid gjort uten at det på dette tidspunkt er sikkerhet for om det gir ønsket nytteeffekt.

Behovsfasen består av kundeinterne diskusjoner, diskusjoner og presentasjoner med leverandørene og av at kunden overleverer et formelt konkurransegrunnlag som leverandørene skal gi tilbud på. Som vanlig skal tilbudene evalueres og kontrakt tegnes med den foretrukne leverandør.

Det kan være vanskelig for kunden å være presis på kravene i en så tidlig fase. Likevel må krav/behov spesifiseres, men ofte trenger de ikke spesifiseres på et detaljert nivå. Uten en

tilstrekkelig spesifikasjon kan ikke leverandørene fremstille sine løsningsforslag som er detaljert nok. Da blir løsningsforslagene vanskelig å prise, tilbud vanskelig å sammenligne og planer vil ha liten troverdighet. Det er denne balansen med 'akkurat nok krav' som kunden bør søke å finne.

Et behov kan komme til overflaten på mange måter og kan være alt fra en idé om et konkret produkt til å understøtte nye bestemmelser i lovverket. Som regel er det underliggende årsaker til behovet som blir uttrykt, og disse årsakene må kunden søke å identifisere og beskrive. Ved å uttrykke det underliggende behovet uten å diktere en gitt løsning, blir rommet for mulige løsninger utvidet. Dette gir rom for smidighet, både med tanke på leverandørens grove løsningsforslag og prosjektgjennomføringen.

En analogi kan være arkitektkonkurranser: da er konkurransegrunnlaget en detaljert beskrivelse av de funksjoner og den kapasitet et nytt bygg skal inneholde, men det er helt opp til arkitektene å bestemme utforming, plassering på tomten, materialvalg med mer.

3.2 Produkter fra behovsfasen

Det er flere viktige produkter fra behovsfasen:

- Behovsanalysen som kunden utarbeider og som representerer kravgrunnlaget på dette stadiet. Den kan konkretiseres med en kost-nytteanalyse som avklarer om prosjektet har livets rett. I begynnelsen vil fokuset være på forretningsmessig nytte, mens analysen vil kompletteres med sikrere tall for kostnader når tilbudene fra leverandørene foreligger
- De løsningsforslag og tilbud som leverandørene utarbeider (som blir forbedret gjennom diskusjoner med kunden)
- Resultater fra en overordnet risikoanalyse som beskriver de antatt største utfordringene i forbindelse med gjennomføringen av prosjektet (risiko, sannsynlighet, konsekvenser og tiltak)
- De øvrige kontraktsdokumentene i avtalens juridiske rammeverk, bilag A med flere, som blir signert i HMP0
- Evalueringsdokumenter som synliggjør kundens kriterier for valg av løsning, og dermed leverandør

3.3 Avgrensning og oppdeling

I mange tilfeller makter ikke kunden å beskrive alle sine krav i forkant av prosjektet. Årsaken er at prosjektet er et engangstilfelle og en læringsprosess for både kunde og leverandør, det vil igjen si at en del av kravene først blir klare etter at arbeidet har pågått en stund. Videre kan det være sentrale krav som er ustabile, for eksempel på grunn av store organisatoriske og reformmessige endringer. Derfor er det ofte bortkastet å strebe etter et komplett kravgrunnlag som det vil ta mange måneder å utarbeide, uten at det vil bli komplett. Det er de viktige og rimelig stabile krav det gjelder å spesifisere.

Krav er både funksjonelle (hva systemet skal gjøre?) og ikke-funksjonelle (hvordan skal det gjøres?). Videre kan det finnes kontraktuelle eller juridiske krav.

Avgrensning er ofte vanskelig – hva skal leveres av *dette* prosjektet og hva skal overlates til andre prosjekter. Hva er innenfor og utenfor dette prosjektets scope? Hvordan skal det avgjøres om en leveranse oppfyller kravene og vil bli akseptert? Det er ikke uvanlig at kunden legger alle sine behov og tanker inn i ett prosjekt som dermed eser ut og blir meget stort. Dette er ikke hensiktsmessig, men gjøres ofte fordi det kan være vanskelig å få bevilget investeringsmidler i flere omganger.

Prioriteringer og funksjonalitetsgrupperinger kan gi grunnlag for opsjoner i kontrakten. Et viktig aspekt ved smidig og iterativ utvikling er at man kan produksjonssette deler før hele det påtenkte systemet er ferdig utviklet. Ved å dele omfanget i flere hovedleveranser, der man utlyser den første hovedleveransen med opsjoner på de øvrige hovedleveransene, kan man ved ferdigstillelse og aksept av den enkelte hovedleveransen vurdere situasjonen og neste leveranse. På denne måten vil det gjennomføres flere løsningsbeskrivelsesfaser i løpet av hele prosjektperioden, som tar hensyn til læring. En slik tilnærming gir smidighet for kunden.

En annen måte å redusere omfanget på i første omgang, er å bevisst utsette utviklingsoppgaver til forvaltningsfasen. Ved å tegne rammeavtale på videreutvikling, kan kunden etter utviklingsprosjektet bestille utvidelser i form av oppdragsavtaler. Disse kan i sin tur organiseres som smidige prosjekter etter beskrivelsen i kapittel 5.

PS2000 åpner i tillegg for å produksjonssette deler av løsningen underveis i arbeidet med en og samme hovedleveranse. Dette prinsippet er kalt ’delvis overtakelse’ i kontrakten. Dersom det ligger til rette for det, er det ingen ting i veien for at kunden produksjonssetter leveransene etter hvert kontrollpunkt, eller nærmere planlagte milepæler underveis. Hyppige produksjonssettinger er et viktig smidig prinsipp og det er erfaringsmessig et av de viktigste risikoreduserende tiltak sammenliknet med tradisjonell systemutvikling. På denne måten vil all driftsproblematikk, opplæring av sluttbrukere med videre bli distribuert utover prosjektperioden, og tilbakemeldinger fra produksjon vil tidlig kunne hensyntas i det videre prosjektforløpet. Delvis overtakelse stiller noen nye krav til partene, blant annet knyttet til feilhåndtering på det som er produksjonssatt, men som samtidig ikke har gjennomgått godkjennningsprøven.

3.4 Karakteren av kravgrunnlaget

Den tradisjonelle kravspesifikasjonen med detaljerte føringer på design og løsning forøvrig, anser vi som oftest å være et dårlig utgangspunkt for smidig gjennomføring. Vekten bør heller ligge på visjon, målsetting, effektmål, dokumentasjon av forretningsprosessene (som de er, og som de ønskes i fremtiden) og behovet for ny funksjonalitet i form av brukstilfeller. Som modell for brukstilfeller kan kunden velge mellom brukerhistorier (’som X ønsker jeg å utføre Y fordi Z’) eller klassiske brukstilfeller (i strippet versjon). Føringer fra kundens IT-strategier og eksisterende tekniske plattform dokumenteres for seg, slik at leverandørene kan ta hensyn til det i sine løsningsforslag.

I gjennomføringen av behovsanalysen vil kunden identifisere brukstilfeller og/eller brukerhistorier på et overordnet nivå. Disse holdes på et overordnet funksjonelt nivå for ikke å begrense løsningsrommet. Videre bør de grupperes logisk i funksjonsgrupper, hvor man i teorien kunne produksjonssatt en funksjonsgruppe på egenhånd (gitt avhengigheter mellom gruppene). Hovedpoenget er at konkurransegrunnlaget skal være tilstrekkelig godt definert til at leverandørene skal kunne forstå oppgavens omfang og vanskelighetsgrad og kan estimere et løsningsforslag med rimelig grad av sikkerhet.

Utfordringen er å beskrive oppgaven godt nok, uten å bruke lang tid og mye ressurser og uten at leverandørenes handlingsrom blir unødig beskåret. Det er vesentlig å unngå å bruke for

mye krefter på arbeid som ikke vil komme til nytte, jamfør prinsippet i Lean Management om 'Eliminate Waste' [19].

4 Løsningsbeskrivelsesfasen

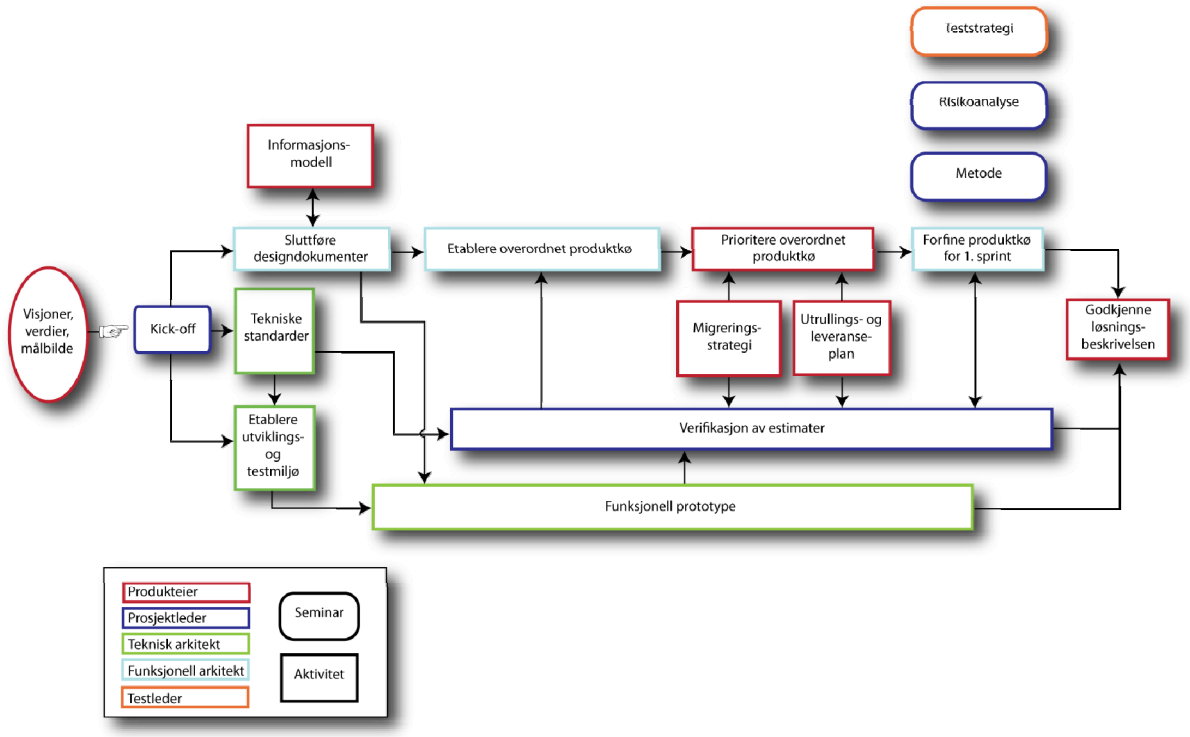
Løsningsbeskrivelsesfasen (LBF) skal legge grunnen for en vellykket gjennomføring av prosjektet. Dette er den fjerde store iterasjonen knyttet til kravgrunnlaget og arbeidet i LBF vil følgelig være sterkt farget av det som har kommet ut av de tre foregående iterasjonene:

- De første delene av behovsfasen med utarbeidelse av konkurransegrunnlaget, som beskrevet i kapitttel 3
- Tilbudsfasen, der leverandøren dokumenterer hvordan et grovt løsningsforslag imøtekommer behov og krav. Tilbudet vil være en ytterligere tolkning og spesifikasjon av kravgrunnlaget og danne grunnlaget for at kunden velger de løsninger/leverandører som anses best egnet for å løse oppgaven og som man ønsker å gå videre med
- Forhandlingsfasen, der partene avstemmer behov og løsning, risiko og kostnader. Mistolkninger og forbehold reduseres eller fjernes, slik at forventningene blir mest mulig samstemt før signering. En av delene som blir særlig detaljert i sluttforhandlingene, er planen for hvordan LBF skal gjennomføres, hvor lang tid som skal settes av til den, hvem som skal delta og hva leveransene er

For smidig metodikk gjelder det å gjøre minst mulig som ikke har umiddelbar nytteverdi. Det innebærer å dokumentere behov, krav og løsninger bare til det detaljnivået som er strengt nødvendig for den fasen man befinner seg i. Fram til signering vil det si at detaljnivået skal tilfredsstille bestemmelsen av en målpris, eller måltimetall, med rimelig grad av sikkerhet. Dette prinsippet vil så bli videreført i LBF.

4.1 Et forslag til prosessmodell for LBF

Figur 4-1 gjengir et forslag til prosessmodell for LBF, som angir avhengigheter og rekkefølge mellom de største aktivitetene. Hovedstrømmen er arbeidet med Produktkøen. Mye av det som sies i det følgende vil være relevant for PS2000-prosjekter generelt, enten de er basert på smidig metodikk eller ikke.



Figur 4-1: Forslag til prosessmodell for LBF

4.2 De viktigste leveransene fra LBF

4.2.1 En omforent produktkø

Den absolutt vesentligste leveransen fra LBF er en omforent produktkø. Produktkøen skal inneholde alle elementer fra behovsanalyse og løsningsforslag som har timekostnader knyttet til seg. Dersom estimeringsmodellen som ligger til grunn for avtalen er bygget rundt dokumenterte brukstilfeller, er mye av arbeidet allerede utført.

Til hvert element i produktkøen skal det knyttes minst to egenskaper: et estimat og en prioritet. Estimatet kan være i timer eller en annen enhet for arbeid (funksjonelle punkter, poeng, hele eller halve dagsverk). Det må uansett ved utgangen av løsningsbeskrivelsesfasen foreligge en kobling mellom de estimerte produktelementene og målprisen i kontrakten. Prioriteten kan være 3- 4- eller 5-delt, eventuelt kontinuerlig som en innbyrdes ordning av samtlige produktelementer.

Det er flere forfattere i det smidige miljøet som advarer mot lange produktkøer. Mary Poppendieck anbefaler for eksempel at man har en lengde på 3 leveransesykler eller ideelt sett tre sprinter per produktkø [19]. Ting som ikke kommer med i køen til de neste tre syklene kan man ta inn igjen i senere leveransesykler.. På grunn av kontraktskravene må det lages en produktkø for hele prosjektperioden, men man kan lage produktkøelementene lengst ut i horisonten rimelig store. Det vil si typisk 100 funksjonelle punkter eller større.

I tillegg må produktkøen ordnes i en tenkt leveranseplan. Ved utløpet av LBF vil de egenskapene ved løsningen som antas å ha størst nytteverdi for kunden og/eller størst risiko for prosjektet bli plassert ut i de første sprintene. Erfaring tilsier likevel at hypotesen om hva

som har størst nytteverdi imidlertid vil endre seg utover i prosjektperioden, slik at produktkøen og den tilhørende leveranseplanen, vil bli gjenstand for korreksjoner etter hver sprint.

Det er ikke nødvendig å plassere ut produktelementer på de sprintene som ligger et stykke ut i prosjektet. Det er tilstrekkelig med en hypotese om antall sprinter, og volumet av arbeid pr. sprint, slik at det totale volumet tilsvarer volumet av samtlige produktelemeter som inngår i løsningen.

For å kunne etablere en produktkø med de egenskapene som er nevnt, vil det ofte være nødvendig med en ytterligere detaljering av prosessmodell, brukstilfelle og løsningsforslag. Dette kan være nødvendig både for å avdekke avhengigheter i løsningen, og for å kunne verifisere estimatene med rimelig grad av sikkerhet. Man skal ikke detaljere mer enn det som er absolutt nødvendig.

Det kan være krav (og tilbud) i bilag A som ikke omfattes av produktkøen, for eksempel krav eller føringer til løsningen som går på tvers av flere produktelementer og som derfor ikke er priset separat. Vi anbefaler imidlertid at produktkøen formelt inngår i løsningsbeskrivelsen på en slik måte at den definerer omfanget av leveransen på alle de områdene der den erstatter bilag A. Øvrige føringer på løsningen opprettholdes som de er beskrevet i kravtabeller og lignende. Enhver omforent (og eventuelt signert) oppdatering av produktkøen underveis i prosjektet vil med dette være en korreksjon av kontraktens omfang. Med et forenklet endringsregime kan partene avtale at dette er tilstrekkelig der oppdateringen verken har konsekvenser for kostnader eller fremdrift. For kontraktuelle konsekvenser for del 1, se kapittel 5.8.

Noen prosjekter rapporterer god erfaring med at produktkøen kun inneholder det som har forretningsverdi. I tillegg kommer infrastrukturelementer og timebasert arbeid som ikke direkte knytter seg til brukstilfellene. Disse kan legges i en egen kø, på samme format og med samme verktøy som produktkøen. Man bør imidlertid vite hvilke avhengigheter det er mellom slike rent tekniske produktelementer og elementer med forretningsverdi. Disse avhengighetene, for eksempel element X må være produsert før element Y kan produseres, må dokumenteres slik at kunden kan foreta prioriteringer basert på et mest mulig reelt kostnadsbilde.

Produktkøen bør registreres i et dertil egnet verktøy. Vi har gode erfaringer med å bruke et såkalt ’issue tracking system’ eller oppgavehåndteringssystem. Felles for disse er at de er basert på en database. Vi anbefaler ikke å bruke regneark til produktkøen, fordi både historikk, vedlegg, flytting av eierskap, prosessflyt, statussetting og gjensidige koblinger er svært nyttige egenskaper, som bare lar seg realisere fullt ut med en databaserepresentasjon i bunnen.

4.2.2 Verifiserte estimater

En annen viktig leveranse fra LBF er verifikasjon av estimatene som lå til grunn for signert avtale. Gjennom detaljering av brukstilfeller i behovsanalysen, arbeidet med elementene i produktkøen og funksjonell prototyping (Proof of Concept, PoC) skaffer prosjektet til veie informasjon som kan påvirke estimatene i positiv eller negativ retning. En ordentlig kalibrering av valgt gjennomføringsmodell får man imidlertid ikke før man har kjørt noen sprinter. Da vil man ha funnet mengden av funksjonalitet man kan ta unna i en sprint, og med det kunne gi en prognose med enda større sikkerhet.

I produktkøen er alle elementene som skal produseres listet opp med estimater. Når man legger til faste kostnader, som kan være rollebaserte pådrag og kostnader til testfasene, får man et kostnadsoverslag som skal mappes til målpristimetallet i kontrakten. Dette kostnadsoverslaget skal legges til grunn for det periodiserte budsjettet i prosjektet.

Dersom leverandøren under LBF oppdager at det er endrede forutsetninger for estimatene, kan han i henhold til avtalen utstede en endringsordre på vanlig måte. Kunden på sin side står fritt til å akseptere eller underkjenne endringen.

Når det gjelder fremgangsmåte for å verifisere estimatene fra tilbuds- og forhandlingsfasene, vil vi anbefale å gjøre dette i flere skritt:

- For å verifisere estimatene pr. brukstilfelle, er det av stor verdi med en prototype av ett eller flere brukstilfeller (PoC) som inkluderer alle lagene i arkitekturen i det tenkte løsningskonseptet (GUI, portallag, applikasjonstjener, databaselag, tjenestebuss, regelmotor med videre). En slik kjørende prototype vil ikke bare fungere som en verifikasjon av anbefalt arkitektur, men vil også gi verdifull informasjon om kostnadene ved å produsere det aktuelle brukstilfellet
- Gå grundig gjennom ett eller flere nært relaterte brukstilfeller og vurder kompleksitet, avhengigheter etc. og estimer disse bottom-up med de erfaringene om ressurspådrag som PoC har skaffet til veie. Dette innebærer detaljering av brukstilfellene på et fokusert område til et slikt nivå at det vil være mulig å estimere kostnadene ved å frembringe dem med høy grad av sikkerhet. For å oppnå tilstrekkelig detaljering, er det nødvendig å avdekke all funksjonell kompleksitet i forbindelse med det utvalgte brukstilfellet. Det vil si GUI og interaksjonsdesign, arbeidsflytbetingelser, statusendringer, logging av hendelser, tjenestekall til regelmotor, utvikling av regler, øvrige tjenestekall til eksterne kilder, datamodell-elementer og kodeverk, behov for migrering fra dagens applikasjoner, programmering av automatiserte tester med videre
- Øvrige delprosesser og brukstilfeller i løsningen vil også bli gjort til gjenstand for funksjonell analyse, men bare til et nivå der det gir mening å estimere dem ved bruk av analog estimering. Dette gjøres ved å etablere noen arketypiske standard brukstilfeller fra løsningen med karakteristisk ulike volumer og sammenholde de øvrige brukstilfellene med disse standard estimatene. Selve estimeringen kan med fordel utføres i workshops der begge parter deltar, for eksempel med bruk av Planning Poker
- Estimatene vil bli lagt inn i en estimeringsmodell som legger på prosesskostnader, testkostnader og overhead etter en bestemt fordelingsnøkkel, sammen med enkelte faste prosjektkostnader som en funksjon av teamstørrelse og kalendertid. Til dette kan for eksempel en Use Case Point analysemodell benyttes [20]

4.2.3 Øvrige leveranser fra LBF

I prioritert orden:

- En verifikasjon av arkitekturen med alle lagene implementert (PoC). Denne kan suppleres med et arkitekturdokument som verifiserer eventuelle valg som gjøres med

hensyn til arbeidsdeling mellom komponentene i løsningen. Arbeidet med PoC bør utføres i form av en eller flere sprinter, blant annet slik at prosjektet skaffer seg erfaringer om forholdet mellom funksjonspoeng, timer og fart (jfr. kapittel 5.1)

- En klargjort produktkø for første sprint. Når det gjelder kvaliteten på produktelementene som skal være kandidater for sprintplanlegging, må dette avklares mellom partene. Se kapitlene 5.2og 7.1
- Etablering av utviklings- og testmiljøer som skal benyttes i sprintene og verifikasjon av disse. Som et minimum vil vi anbefale en felles byggetjener for utviklingsprosjektet der det bygges kontinuerlig, samt en mer stabil testtjener der det bygges daglig, ukentlig eller en gang pr. sprint
- En leveranseplan, som er en hypotese i forhold til antall sprinter i prosjektet, lengden av sprintene, når kontrollpunktene er, antall sprintteam og volumet av produktelementer som planlegges levert i hver sprint. Leveranseplanen vil inneholde en plan for mulige produksjonssettinger underveis. Den vil fungere som et periodiserte budsjettet for prosjektperioden som opptjent verdi kan måles mot i rapporteringen kapittel 5.7). Leveranseplanen blir oppdatert i forbindelse med kontrollpunktene
- Konseptuell informasjonsmodell, samt en logisk modell til et slikt nivå at det understøtter første sprint i konstruksjonsfasen
- Retningslinjer for samhandling i prosjektet. Vi har gode erfaringer med et heldagsseminar med alle involverte for å etablere slike retningslinjer. Det er spesielt viktig å avtalefeste aktiviteter, rollefordeling og ansvar i forbindelse med inngangen til og utgangen fra sprintene, inkludert kontrollpunktet. Under samhandling inngår også en omforent teststrategi for prosjektet, som nedfeller retningslinjene for testarbeidet i prosjektet, krav til roller, ansvar og leveranser både på kunde- og leverandørsiden
- Dokumentasjon av standarder som skal følges i prosjektet. Dette er for eksempel kodestandard, bruk av verktøy, GUI-plakat med prinsipper for interaksjonsdesign og eventuelle grafiske elementer, konfigurasjonsstyringsprinsipper, kvalitetssikringsregimer med videre
- En omforent risikoanalyse av prosjektet. Denne bør inneholde identifiserte risikoforhold, kvantifisering av sannsynlighet og konsekvens, eventuelle risikoreduserende tiltak med ansvarlige og frister med videre.. Vi anbefaler et heldagsseminar der samtlige nøkkelroller hos kunden og leverandøren deltar. Seminaret bør fasiliteres av spisskompetanse på kvalitativ og kvantitativ analyse av risiko

5 Gjennomføringsfasen

I denne veilederen legger vi til grunn at hver sprint i det smidige utviklingsløpet blir et trinn i PS2000. Dette er en mulig måte å gjøre det på, men det finnes også eksempler på prosjekter der man har kjørt flere sprinter pr. trinn.

5.1 Innholdet i en sprint

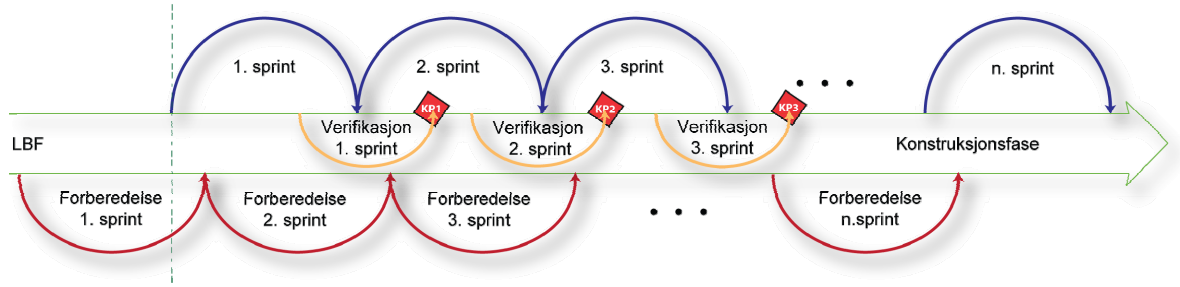
Vi har best erfaringer med de prosjektene der man lykkes med å sammenføye kravene til gjennomføring av sprinten med kravene til gjennomføring av PS2000-trinnet. Det vil si at hver sprint inneholder tre ulike aktiviteter:

1) Detaljplanlegging, analyse og design
2) Utvikling
3) Testing

Merk at smidig utvikling i praksis innebærer at detaljplanlegging/analyse/design, utvikling og testing skjer kontinuerlig innenfor en sprint og ikke nødvendigvis trinnvis i ovennevnte rekkefølge.

Ved avslutning av sprinten skal det gjennomføres et kontrollpunkt.

En utfordring med denne modellen er å motvirke hull i framdrift og ressursbruk på grunn av det administrative arbeidet med å gjennomføre kontrollpunktet. Med bakgrunn i dette vil vi foreslå en modell der sprintteamet gjennomfører kontinuerlige sprinter gjennom hele gjennomføringsfasen slik som vist i figur 5.1. For å oppnå dette, må partene utføre forberedelser til neste sprint og verifikasjon av forrige sprint i parallell.



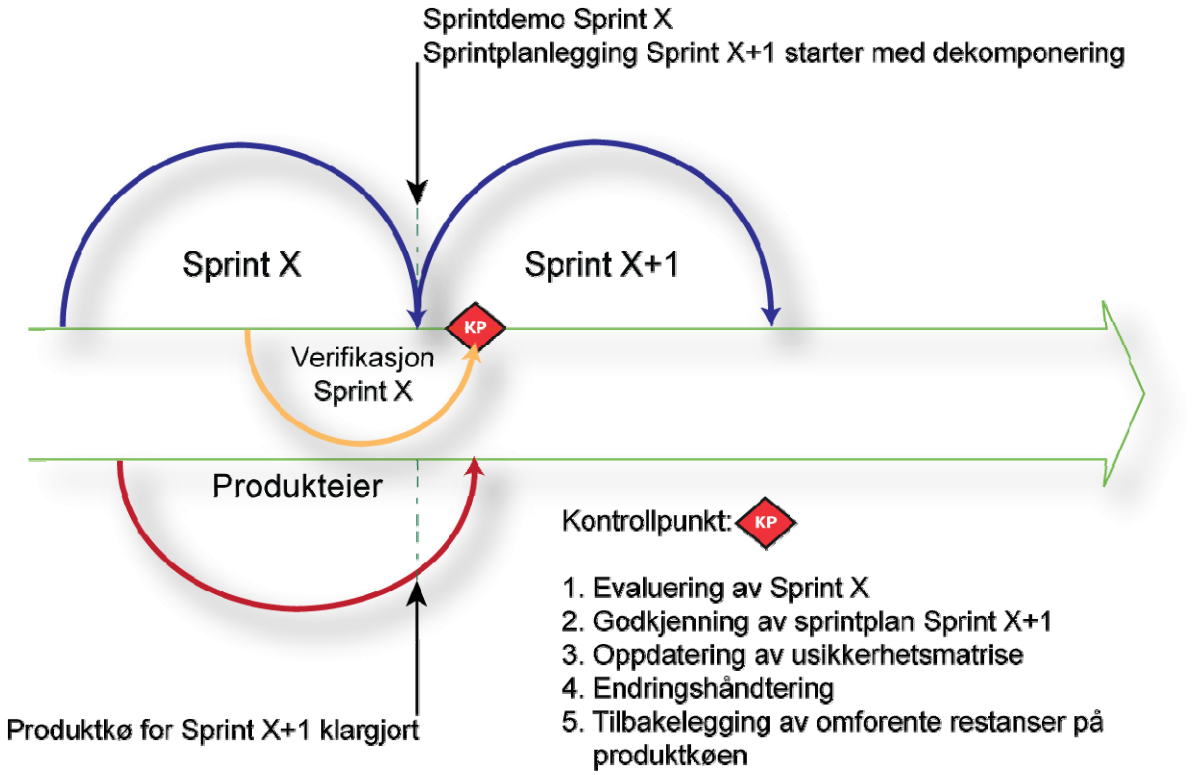
Figur 5-1: Kontinuerlige sprinter i gjennomføringsfasen

I figur 5.2 er selve innholdet i en tenkt sprint fremstilt i detalj. Hver sprint har i tillegg til de aktivitetene som er beskrevet i forbindelse med et PS2000 trinn også en aktivitet med forberedelser og en hale fra sprintens avslutning til og med kontrollpunktet. Lengden av disse for- og etter-aktivitetene kan variere fra 1-2 dager og opp til en uke, alt etter størrelsen på prosjektet eller spesifikke forhold hos kunden og leverandøren. I noen prosjekter har vi erfaring med at forberedelser av produktkøen for en sprint tar like lang tid som hele den pågående sprinten.

Merk at det kan innebære en risiko å ha ovennevnte hale fra sprintens avslutning til kontrollpunktet. Det anbefales derfor på det sterkeste å redusere halen så mye som mulig gjennom meget tett involvering av oppdragsgiver, testpersonell og øvrig mottaksapparat i selve sprinten slik at testing og andre kontrollpunktaktiviteter kan gjennomføres på en mest

mulig kontinuerlig basis. Dette er også noe som er understreket i veilederen til PS2000 (v 2.9, s.23):

"Det er viktig å arbeide effektivt både på kundens og leverandørens side i forbindelse med kontrollpunktet, slik at det ikke oppstår forsinkelser og unødig dødtid for prosjektets personell. I størst mulig grad bør kundens evaluering av gjennomført trinn og leverandørens planlegging av neste trinn gå i parallell."



Figur 5-2: Sprint i detalj

I figur 5-2 ser vi at kontrollpunktet for sprint X sammenfaller med godkjenning av sprintplan for sprint X+1. I de tilfeller der sprintteamet må påbegynne arbeidet med sprint X+1 før kontrollpunktet for sprint X er passert, for eksempel i de tilfellene der kundens verifikasjon av sprint X tar lenger tid enn forutsatt, introduseres en risiko for det arbeidet som utføres inntil kontrollpunktet er oppnådd. Vi mener at partene bør kunne enes om å dele denne risikoen. Den enkleste kontraktsmessige måten å håndtere dette på er at eventuelt merarbeid kan medføre overskridelser av målpris med de risikofordelinger som ellers gjelder i kontrakten. Dersom årsaken til slike overskridelser er forhold som kunden i hovedsak har ansvar for, kan leverandøren kreve å utstede endringsordre, også dette etter kontraktens generelle vilkår. Alternativt kan det avtales en særegen kostnadsdeling for akkurat dette, se kapittel 5.6. Risikoen kan unngås dersom kundens representanter i sprintteamet arbeider tett sammen med leverandørens ressurser i sprintplanleggingen for sprint X+1, slik at kundens godkjenning av sprintplanen kan skje samme dag som den foreligger.

5.2 Forberedelse

Når det gjelder produkteiers forberedelse av produktkøen forut for sprintplanlegging, er erfaringene litt sprikende. Dette gjelder også fra Scrumbaserte prosjekter som ikke kjører under leveransekontrakt. I modellen vår har vi tatt hensyn til dette. Vi mener at det er vesentlig at partene avtalefester kvalitetsnivået på den delen av produktkøen som kan være aktuell for sprinten og som skal foreligge ved inngangen til en ny sprint. Her kan det for eksempel inngå krav til at interaksjonsdesignet er definert med et visst nivå på skjermbildeskisser. Det kan vedlegges maler som eksemplifiserer nivået. Det kan videre være krav til faglige avklaringer, som regelsett og rutinebeskrivelser som gjelder for de brukstilfellene som ligger øverst i produktkøen. Videre bør det være krav til utforming av akseptansekriterier for de samme brukstilfellene.

Det er ikke avgjørende om produkteiers eller produktteamets arbeid med klargjøring av produktkøen for neste sprint skal inngå i de enkelte sprintplanene eller ikke. Dersom det bare er ett sprintteam i prosjektet, er det antagelig naturlig at det inngår. I tilfeller der produkteier skal betjene flere sprintteam, blir det ikke like naturlig.

Det er viktig at forberedelsene skjer tilstrekkelig tidlig slik at avklaringer som tar tid, for eksempel ved at mange i kundeorganisasjonen må høres, blir ferdige til sprintoppstart.

I arbeidet med å klargjøre produktkøen forut for sprinten kan følgende oppgaver inngå:

1) Forfining av høyt prioriterte brukstilfeller eller brukerhistorier, for eksempel ved å utforme varianter og unntak fra hovedscenarie - 'happy day case'
2) Arbeid med interaksjonsdesign, dersom dette er aktuelt for produktene øverst i køen. Dette kan for eksempel være i form av skjermbildeskisser
3) Avklaring av faglige forhold knyttet til kvalitetsstyring av virksomhetsprosesser, rutinebeskrivelser, regelsett, rettslige spørsmål og lignende
4) Utforming av testbeskrivelser og akseptansekriterier for produktelementene øverst i produktkøen. Hvert brukstilfelle bør ledsages av en prosedyre for verifikasjon (et testcase), som beskriver forventede resultater til et sett med inputverdier eller hendelser

I begynnelsen av sprinten ser vi at produktteamet bruker noe tid på å bistå sprintteamet med avklaring på de oppgavene de jobber med. For øvrig bør kundens representanter i sprintteamet kunne håndtere det meste av nødvendige avklaringer som oppstår. Produktteamet bør derfor kunne konsentrere seg om forberedelsene til neste sprint.

5.3 Detaljplanlegging, analyse og design

Med utgangspunkt i den definerte produktkøen gjennomføres sprintplanlegging. Tverrfaglige sprintteam med deltakelse både fra kunde og leverandør vurderer og beslutter hvilke elementer fra produktkøen som skal inngå i neste sprint for teamet. Disse settes opp i en prioritert liste, kalt sprintkø. Sprintkøen representerer det realistiske omfang som teamet skal levere i løpet av en sprint. En sprint er en fast, kortere tidsperiode (timebox) som typisk defineres til 4 ukers varighet.

Sprintplanleggingen tar utgangspunkt i de høyest prioriterte elementer i produktkøen og splitter hvert element opp i flere deloppgaver (dekomponering). Deretter gjør man en nærmere vurdering av arbeidsmengde og antatt tidsforbruk knyttet til realisering av hvert

element. I tillegg til selve utviklingen av enhetene skal forberedelser og gjennomføring av testarbeid og andre kvalitetssikringsaktiviteter inngå som en del av de aktiviteter som utgjør arbeidsomfanget. Det bør også legges inn et buffer for feilretting, både fra foregående og inneværende sprint.

Basert på sprintens definerte, faste varighet, prioritet og antatt arbeidsmengde for hvert element, samt teamenes ressurssituasjon, defineres et utvalg elementer som skal inngå i sprintkøen for realisering i neste sprint. Kunden og leverandøren må sammen vurdere underlaget og utarbeide et omforent innhold i sprintkøen. Kunden bidrar med vurderinger av omfang og prioritet mens leverandøren gjør betraktninger basert på utredet arbeidsmengde og tilgjengelige ressurser.

Ressurstilgangen vurderes også i forhold til teamdeltakernes antatte mulighet til å fokusere på arbeidet i sprinten, såkalt fokusfaktor. Planlagt og ikke planlagt fravær samt andre forstyrrende elementer, for eksempel behov knyttet til vedlikeholdsforpliktelser mot produksjonssatte moduler, er forhold som må tas hensyn til i denne vurderingen.

Et viktig premiss for sprintplanleggingen, er teamenes erfarte hastighet. Dette er et mål på hvor mye teamet klarer å produsere i løpet av en sprint. Så lenge kontrollpunktet med kundens evaluering av foregående sprint ikke foreligger, vil det hefte noe usikkerhet ved farten fra den foregående sprinten. Det kan være at det identifiseres restanser og feil ved leverte produkter, som tilsier at farten i foregående sprint var noe lavere enn først antatt. Denne usikkerheten vil imidlertid reduseres etter hvert som prosjektet får erfaringsmateriale fra flere sprinter.

I større prosjekter vil det være aktuelt å definere flere sprintkøer som kan danne grunnlag for parallelle sprinter for ulike sprintteam som fokuserer på ulike deler av en større leveranse.

Sprintplanlegging omfatter videre sammensetting av sprintteam for neste sprint. Man kan velge å beholde teamsammensetting fra tidligere sprinter knyttet til de samme moduler og funksjonsområder i koden. Dette kan gi fordeler knyttet til å bevare effektive samkjørte grupper, utnytte spisskompetanse og lignende. Det er klare fordeler og muligheter i å endre sammensetting av sprintteamene. Hyppige rollebytter og rullering bidrar til kompetansespredning og god kommunikasjon på tvers av teamene. Omrokkering av team kan være et effektivt virkemiddel for å forebygge mot og eventuelt løse opp i samarbeidskonflikter.

Konkret resultat etter sprintplanlegging bør være:

1) Et definert mål for sprinten
2) En liste av teamdeltakere både fra leverandøren og kunden og deres allokering
3) En sprintkø brutt ned i estimerte oppgaver
4) Retningslinjer for evaluering av sprinten, dato for sprintdemo og sprinttilbakeblikk
5) Definert tid og sted for daglig scrum

Det er av avgjørende betydning at representanter fra kunden kan prioritere deltakelse i Sprintplanlegging. I tillegg til at kunden har en klar rolle i forhold til løpende prioritering av elementer, er det viktig at kunden etablerer forståelse for at det er aktiviteter som må utføres for realisering av elementer i køen i tillegg til selve utviklingsarbeidet. Behov for spesiell tilrettelegging i byggesystemer, tekniske miljøer, konfigurasjonsstyringsverktøy, spesielt omfattende testforberedelser, testgjennomføring med videre kan påvirke arbeidsmengden for aktuelle elementer i en sprint. Slike forhold kan skape grunnlag for å prioritere innholdet i en

sprint på en annen måte. Kundens nærvær er også viktig i forhold til å raskt kunne fremskaffe avklaringer på kundefaglige detaljspørsmål.

Det er viktig at det skapes forståelse hos kunden for at en for omfattende og innholdsrik sprint kan gi tidspress og dermed innebære risiko i forhold til testarbeidet og kvaliteten. Metoden fremhever viktigheten av korte sprinter (maksimalt 4 uker), korte akseptansesykler og en sprintkø med et omfang som oppleves som oppriktig realistisk av hele sprintteamet.

Sprintplanen skal formelt godkjennes av kunden (PS2000, del II, § 3.4.5:

"*Arbeidet med etterfølgende trinn skal først igangsettes når Kunden innen en frist som er definert i Bilag C, har godkjent planen og den oppdaterte usikkerhetsmatrisen.*"

Vi har anbefalt at denne fristen settes så kort som mulig (se kapittel 5.1). Det aller beste er om partene enes om at slik godkjenning kan gis samme dag som sprintplanen foreligger. Både sprintplanen og godkjenningen bør foreligge skriftlig eller elektronisk, for eksempel på e-post eller ved signatur i prosjektweben. De dekomponerte arbeidsoppgavene i sprinten behøver ikke omfattes av dette formelle regimet. Her mener vi at gule lapper eller tilsvarende er tilstrekkelig.

5.4 Utvikling

En sprint kjøres av et definert sprintteam. Denne arbeidsgruppen kan bestå av representanter både fra kunde og leverandør og ledes av en scrumleder. Metoden understøtter god kommunikasjon og samhandling i teamet gjennom daglig scrum. Disse møtene skal ha fast agenda og kort varighet, typisk 15 minutter. For å sikre at de daglige scrummøtene blir korte og effektive avklaringsmøter er det viktig at de gjennomføres stående.

Det legges vekt på at teamet bør være mest mulig samlokalisert med lett tilgang til et fellesareal. Det anbefales videre at sprintens fremdrift visualiseres ved bruk av en sprintaktivitetstavle på veggen i fellesarealet. Denne skal vise hvilke oppgaver som ligger i køen, hvilken prioritet de har, hvilke som er påbegynt og hvilke som er ferdige. Aktivitetstavlen plasseres lett tilgjengelig og godt synlig og utgjør et samlingspunkt for sprintteamets daglige scrum. I tillegg foreslås en grafisk fremstilling av fremdrift i form av et brenndiagram, også dette fremstilt på tavlen.

Scrum er basert på timeboxing. Det vil si at sprinten alltid avsluttes og evalueres når den på forhånd fastsatte tidsperioden utløper. Gjenstående eller uferdige elementer fra sprintkøen dokumenteres i en restanseliste. Disse elementene vil sammen med resten av produktkøen inngå som grunnlag for å planlegge neste sprint.

Smidig metodikk foreskriver testdrevet utvikling. I dette ligger det blant annet at man skal tilstrebe å skrive testen før man programmerer ny funksjonalitet. Prosjektet bør legge til rette for dette og for mest mulig automatisering av enhetstester, som kan kjøres hver gang det sjekkes inn ny kode på byggetjeneren.

Hvis det viser seg at man blir ferdig med innholdet i sprintkøen før den faste tidsperioden utløper kan ny funksjonalitet legges til fra produktkøen i samarbeid med produkteier.

Metoden åpner for at en sprint kan avbrytes, omdefineres og restartes basert på at det identifiseres uventet høy kompleksitet, ny risiko eller at det oppstår andre forhold som tilsier behov for avbrudd. Dersom partene ønsker å ta høyde for dette, må det beskrives i bilag C hvordan dette skal håndteres.

Sprinten avsluttes med sprintdemo, der produktene fra sprinten demonstreres for prosjektets nøkkelressurser og andre interessenter. Det viktigste på dette møtet er å få tilbakemeldinger på det som er produsert. Ved avslutning av sprinten gjennomføres et sprinttilbakeblikk der sprintteamet gjør en evaluering for å identifisere muligheter for å forbedre prosessen i kommende sprinter.

God framdrift sikres ved at arbeidet med produktkøen gjøres delvis overlappende med gjennomføring av sprintene. Dette vil si at partene begynner å forberede produktkøen for neste sprint i god tid før denne skal starte og at dette glir over i sprintplanlegging, se figur 5-1 og 5-2. Dette forebygger forsinkelse ved at gjennomganger av produktkøen må ferdigstilles før gjennomføring av neste sprint kan starte.

5.5 Testing

Det anbefales at partene enes om en teststrategi i løpet av forhandlingene. Teststrategien bør senest være på plass i løpet av LBF. En teststrategi må si noe om hvilke tester som skal utføres til hvilke tidspunkt i prosjektperioden og den må beskrive roller og ansvar for testarbeidet. Testing omfatter verifikasjon også på andre måter enn kjøring av programkode. Verifikasjon kan for eksempel utføres ved formelle dokumentinspeksjoner, teknisk review, kodegjennomganger, og lignende.

Generelt anbefaler vi at kunden tar på seg mer ansvar for testarbeid enn det som er tradisjonelt. Ingen er bedre skikket til å skrive testtilfeller og verifisere levert funksjonalitet enn representanter for forretningssiden og sluttbrukerne. Dette potensialet bør utnyttes fullt ut i prosjektorganisasjonen. For å få til dette i praksis kreves det at kundens og leverandørens testressurser arbeider tett sammen i hele prosjektperioden.

Vi anbefaler at kunden ved produkteier og produktteamet tar på seg et stort ansvar for planlegging og gjennomføring av verifikasjon i den enkelte sprinten av funksjonalitet, faglig innhold og brukskvalitet. Leverandøren har imidlertid fortsatt hovedansvaret for kvaliteten i løsningen og det er leverandøren som planlegger og gjennomfører all enhetstesting av utviklede komponenter.

Vi anbefaler mest mulig automatisering av funksjonelle tester. Det finnes mange gode testverktøy å velge mellom. Det er flere faktorer som avgjør valget:

- Fordelen med samme verktøy som sprintteamet og resten av testprosjektet
- Verktøyvalg kan allerede være foretatt av kunden
- Vedlikeholdbarhet
- Krav til stabilitet
- Eventuelle supportavtaler
- Pris

Det kreves en viss modenhet på testprosessen før man vil lykkes med bruk av verktøy. Prosesser og rutiner må være på plass for å kunne utnytte verktøyene fullt ut og få ut den ekstra testdekningen man ønsker. En samling av automatiserte funksjonelle tester er nyttig for all regresjonstesting. Erfaring tilsier imidlertid at man i tillegg vil ha behov for manuell testing av sammensatt funksjonalitet og lengre verdikjeder.

Når det gjelder egenskaper ved programvaren som ytelse, robusthet, portabilitet, vedlikeholdbarhet, sikkerhet, grensesnitt og lignende, så bør partene gå opp ansvarsforholdet i detalj. I mange situasjoner vil det være kunden som må stå for tilrettelegging av testgjennomføringen. For eksempel i forbindelse med inngåelse av avtaler med tredjepart som skal stille eventuelle testdata og testmiljøer til rådighet for grensesnittesting. Flere av disse testene er erfaringsmessig så dyre å gjennomføre at de ikke kan planlegges inn i ordinære sprinter. I verste fall må de utsettes til leverandørens avsluttende systemtest, eller enda mer dramatisk til godkjenningsprøven. Det bør alltid utføres en enkel kost/nytte vurdering når man planlegger hvordan slike egenskaper skal testes.

Systemintegrasjon og testing av dette er en egen utfordring. Antall integrasjonsfeil er økende fordi systemene som utvikles har mange grensesnitt å forholde seg til og det foregår svært ofte flere parallelle prosjekter både internt og eksternt. Det er derfor viktig å tidlig fokusere på integrasjonstester da det i henhold til de smidige prinsippene skal leveres testbar kode. Dette er også vesentlig ut fra risikobetraktninger. Systemintegrasjon innebærer høy risiko og det er derfor viktig å få dette med i så tidlige sprinter som mulig.

Den tradisjonelle 'V-modellen' er fast inventar i de fleste lærebøker om test. Modellen foreskriver at generelle akseptansekriterier for leveransen kan klargjøres i forbindelse med kravgrunnlaget (test-førstprinsippet). V-modellen har den samme svakheten som tradisjonell fossefallstenking: at detaljerte testbetingelser er like gyldige ved oppstarten av prosjektet som ved avsluttende akseptansetesting. Det tas like lite hensyn til læring underveis som i fossefallsmodellen. Dette gjelder om man tenker på V-modellen som gyldig for hele prosjektperioden. V-modellen kan imidlertid forenkles og benyttes som 'Mange-V-modellen', med en V pr. sprint. Slik gir V-modellen et greit bilde på validering og verifisering innenfor den enkelte sprinten. Uavhengig av utviklingsmetodikk anbefales det å fokusere på å finne feilene i den fasen de oppstår. Det er kostnadseffektivt å finne feilene så fort og så tidlig som mulig.

En generell modell for testarbeidet må ta inn over seg at testbetingelsene vil endre seg i takt med alle andre endringer i prosjektperioden, enten det gjelder omprioritering av produktelementer, tillegg eller fjerning av slike. Dette tilsier at detaljert testplanlegging ikke bør påbegynnes før scopet er frosset, det vil si i forbindelse med hver enkelt sprintplanlegging (jfr. arbeidet med produktkøen i forkant av sprintplanlegging). I tillegg har vi erfaring for at selve testingen også innebærer læring og at noe av det viktigste tilfanget til forfining av testbetingelsene kommer fra selve testingen.

Scrum gjøres testdrevet ved at man i starten av sprinten definerer og planlegger de kvalitetssikringsaktiviteter som produkter fra en sprint skal være gjenstand for. Dette kan omfatte tilrettelegging av testverktøy, skriving av automatiserte tester, spesifisering av manuelle tester, definering av omfanget av kodegransking, kundeinspeksjoner, demonstrasjoner, dokumentgjennomganger med videre.

Etter hvert som produktene blir levert i sprinten, blir disse evaluert. I løpet av den første halvdelen av sprinten bør de viktigste produktelementene være realisert og det bygges da en første leveransekandidat som legges ut i et testmiljø. Alle i sprintteamet og produktteamet, utviklere, testere og faggrupperepresentanter har tilgang til utviklingsmiljøet som bygges daglig gjennom hele sprinten. Likevel, ved å legge ut en tidlig versjon til testmiljøet åpner man for at flere fra kunden kan starte tidlig testing og gi tilbakemelding mens det ennå er tid til å gjøre små forandringer til det som utvikles. Feil og kommentarer til det som utvikles tas hensyn til så langt det er tid, men ikke slik at det truer datoen for sprintdemo.

Det er som sagt vesentlig at alle roller, ansvar og myndighet knyttet til forberedelse og gjennomføring av tester er avklart mellom partene på forhånd. Vi anbefaler et sterkt engasjement fra kundesiden når det gjelder verifikasjon av funksjonalitet, faglig innhold og brukskvalitet i hver enkelt sprint. Det kan for eksempel avtalefestes at det skal delta en testansvarlig fra kundesiden i hvert enkelt sprintteam, som har ansvaret for å forberede og lede kundens deltakelse i sprinttesten.

Den strukturerte sprinttesten avsluttes med en sprintdemo. Perioden frem til kontrollpunktet benyttes til videre verifikasjon av leveransen og forberedelser av kontrollpunktet.

Erfaringsmessig er det mest krevende arbeidet å filtrere listen av testobservasjoner med hensyn til feil, mangler, endringer, omforente restanser og ikke-omforente restanser. Her kommer prosjektlederne på begge sider inn, på basis av innstillingene fra produkteier på kundesiden og løsningsansvarlig på leverandørsiden. Basert på erfaringer fra andre PS2000-prosjekter, vil det kunne gå 1-2 virkedager til å ferdigbehandle dette. En samlet vurdering tilsier at hele prosessen vil kunne strekke seg til 3-5 virkedager etter sprintdemo, dersom kundesiden arbeider godt og planmessig med dette.

Eventuelle feil og omforente restanser legges tilbake på produktkøen og prioriteres på samme måte som andre produktelementer.

5.6 Hva skjer i kontrollpunktet

Kontrollpunktet nås når evaluering av sprinten er avsluttet og det er avholdt et møte mellom partenes prosjektledere, produkteier og løsningsansvarlig. Som grunnlag for gjennomføring av kontrollpunktet trenger partene

- En protokoll som viser resultat av leverandørens test av sprinten
- En protokoll som viser resultat av kundens test av sprinten
- En liste over restanser/avvik som skal legges i produktkø og prioriteres av produkteier
- Eventuelt endringsanmodninger som har vært utstedt og utredet i gjennomførte trinn
- En plan for etterfølgende trinn basert på overordnet leveranseplan og erfaringer fra gjennomført trinn

Det formelle kontrollpunktmøtet mellom partene må behandle

- Hvilke restanser som er utestående
- Kundens evaluering av leveransen i kontrollpunktet, med de forbehold som er listet i foregående punkt
- Merkantile avklaringer i forbindelse med eventuelle endringer
- Gjennomgang av usikkerhetsmatrise
- Nødvendige justeringer av leveranseplanen og prosessen

Det er viktig å være presis på hva det er som verifiseres i kontrollpunktet. Det kan for eksempel være tilfelle at det i tidlige leveranser ikke er mulig å verifisere slike forhold som vedlikeholdbarhet, robusthet, ytelse, portabilitet og sikkerhet. Slike forhold må likevel bygges inn i designet fra begynnelsen av og det bør planlegges med testing av dette så tidlig som mulig. Vi vil foreslå som et minimum at kontrollpunktet vektlegger verifikasjon av funksjonalitet, faglig innhold og brukskvalitet. Signert verifikasjon underveis av disse

forholdene vil langt på vei sikre reell fremdrift i utviklingsprosjektet. De forholdene som man eksplisitt ikke tar stilling til i kontrollpunktene, vil kunne adresseres i en senere sprint, i leverandørens avsluttende systemtest eller i godkjenningsprøven. Ideen er at produkter (features, brukstilfelle og lignende) som er verifisert i et kontrollpunkt skal oppfattes som utført. Kunden skal ikke senere i prosjektet kunne påberope seg behov for ytterligere videreutvikling av funksjonelt og faglig innhold på verifiserte produkter som mangler ved leveransen. Dette blir i så fall å betrakte som endringsanmodninger, som må håndteres etter bestemmelsene om endringshåndtering og eventuelt tilføres produktkøen dersom de blir godkjent.

Ikke omforente restanser utgjør et spesielt forhold i kontrollpunktet. Dette oppstår for eksempel når kunden identifiserer et forhold som en feil eller mangel ved leveransen, men som leverandøren på sin side mener er utenfor kontraktens omfang. I en slik situasjon, der kunden vil flytte dette forholdet tilbake på produktkøen, og leverandøren mener det vil kunne påvirke prosjektets kostnader eller gjennomføringstid, bør leverandøren utstede en endringsanmodning. Denne behandles i henhold til bestemmelsene om endringshåndtering i de generelle kontraktsvilkårene (se kapittel 5.8).

Et spesielt forhold knytter seg til godkjenning av sprintplanen for kommende sprint (se kapittel 5.1). Vi anbefaler som tidligere beskrevet at partene avtalefester å behandle godkjenning av sprintplanen samme dag som den foreligger. I de tilfeller der partene likevel avtalefester at formell godkjenning av sprintplan for sprint X+1 skal skje i forbindelse med kontrollpunktet for sprint X, kan partene vurdere følgende tillegg til de generelle kontraktsbestemmelser: Del II § 3.4.5, 3. avsnitt settes inn i kontraktens del I:

”Eventuelt merarbeid som leverandøren påføres ved at sprintplanen underkjennes i kontrollpunktet skal endringshåndteres. Slik endring godtgjøres med halvparten av dokumentert merarbeid, eller halvparten av et omforent estimat for merarbeidet.”

5.7 Rapportering (brenndiagram og s-kurven)

Man kan vurdere å slå sammen kontrollpunktmøte og månedsrapportering fra prosjektet, dersom man kjører månedlige sprinter.

Sentralt i kravene til rapportering i bilag B i PS2000 kontrakten er S-kurven og det tilhørende begrepet opptjent verdi (earned value). Dette er et svært nyttig verktøy for å vurdere fremdrift og prognoser for kostnader. Vi anbefaler at partene legger dette til grunn for kravene til rapportering i smidige prosjekter.

I denne veilederen vil vi forutsette at opptjent verdi defineres som ”estimert kostnad for utført arbeid.” Det vesentlige er at det foreligger et periodisert budsjett, der estimerte produktelementer er lagt ut i tid. Leveranseplanen fra LBF er en hypotese på hvor mye som vil bli produsert i den enkelte sprinten. Leveranseplanen kan bli gjenstand for endringer og innholdet i de enkelte sprintene kan bli gjenstand for omprioriteringer underveis. Så lenge målprisen og datoen for HMP2 står fast, skal man imidlertid ikke endre på det opprinnelige periodiserte budsjettet i løpet av prosjektperioden. Det periodiserte budsjettet er et uttrykk for ”estimert kostnad for planlagt arbeid” (planned value).

Et annet springende punkt er hva vi legger i begrepet utført. Her er det viktig at partene er enig i den definisjonen som legges til grunn. Det er flere muligheter

1) Sprinttestede produktelementer levert til kundens verifikasjon

2) Levert og verifisert funksjonalitet i kontrollpunktet
3) Systemtestet funksjonalitet levert til godkjenningsprøven
4) Godkjent og produksjonssatt

Det smidige idealet er alternativ 4). V anbefaler likevel at man legger til grunn en definisjon av utført som lar seg følge opp internt i sprintene og i forbindelse med sprintdemo og kontrollpunktet knyttet til sprintene, som i alternativ 1). Dette betyr at 'utført' utgjør samlingen av produktelementer som er levert til kundens verifikasjon. Dette innebærer videre at kostnader til testing av ikke-funksjonelle forhold, samt leverandørens avsluttende systemtest estimeres for seg, som et påslag på øvrige utviklingskostnader. Videre innebærer det at estimatene for de enkelte produktkø-elementene må inkludere alle kostnader til analyse, design, utvikling, enhets- og integrasjonstest, samt administrasjon. Endelig betyr det at kostnadsoverslagene til omforente restanser legges til den globale ETC etter hvert kontrollpunkt, sammen med de opprinnelige estimatene for produktelementer som enda ikke er levert.

Ulempene ved å legge definisjonene i 3) og 4) til grunn, er at det blir rapportert kunstig lite fremdrift inntil henholdsvis HMP2 og HMP3 oppnås. Det betyr tilsvarende lite støtte for å gjøre grep underveis. Ulempen ved alternativ 2) er at det også kan bli rapportert kunstig lite fremdrift, for eksempel dersom det i forbindelse med kontrollpunktet blir identifisert enkelte små restanser knyttet til flere leverte brukstilfeller. Med alternativ 1) risikerer man noe overrapportering i perioden forut for kontrollpunktene. Dette vil likevel utgjøre et mye sikrere mål på fremdrift enn rapportering fra tradisjonelle IT-prosjekter basert på for eksempel RUP. Med RUP vil man bruke skjønnsmessig ETC på samtlige påbegynte brukstilfeller som ligger til grunn for beregningen av opptjent verdi.

Man estimerer produktelementer i form av funksjonelle punkter eller poeng. For å sammenholde dette med meteret i s-kurven, som er timer, må prosjektet etablere en oversettelsesfunksjon mellom funksjonelle poeng og timeverk. På grunn av læring underveis i prosjektet vil farten øke i de første sprintene. Man kan regulere oversettelsesfunksjonen slik at de funksjonelle poengene som er planlagt i tidlige sprinter har en høyere timeverdi. Den samlede timeverdien for hele prosjektet må være avstemt med avtalt målpris.

Det er vanlig å følge opp fremdrift internt i sprintene med ETC på de enkelte sprintoppgavene som er i produksjon. Ofte er disse uttrykt i timer og figurerer på gule lapper. Etter hver daglige scrum legger scrumleder sammen disse verdiene, sammen med verdiene for ikke påbegynte sprintoppgaver og ikke dekomponerte produktelementer. Det er dette som fremkommer i brenndiagrammet for det enkelte sprintteam. Ved å summere verdiene fra hvert team, vil leverandørens prosjektleder automatisk ha en daglig kontroll på fremdriften i prosjektet fordi den samlede ETC er et invers uttrykk for utført. Produksjon av ukentlig eller månedlig s-kurve for prosjektet som helhet er kun snakk om formler, dersom prosjektet også tar hånd om timefangst "faktisk kostnad for utført arbeid" (actual value) pr. rapporteringstidspunktet.

5.8 Endringshåndtering

Sammenhengen mellom smidig metodikk og formell endringshåndtering blir ofte misforstått i smidige prosjekter. I denne sammenheng er det viktig å fastholde at de formelle endringsprosedyrer som PS2000 legger opp til må følges også når prosjektet gjennomføres etter smidig metodikk. Dette kan medføre utfordringer ettersom smidig metodikk på et

grunnleggende nivå på mange måter driver frem endringer for å øke leveransens kvalitet og relevans, samtidig som dette krever formell oppfølging innenfor endringsregimet.

Flytting av brukstilfeller, funksjonalitet eller oppgaver mellom sprinter behøver ikke å anses som endringer dersom de ikke innebærer en endring til totalkostnad, milepælsplan eller systemets/leveransens virkemåte. Et eksempel på dette kan være at man innenfor planlegging av en sprint beslutter å flytte et brukstilfelle til en senere sprint.

Prioriteringer som innebærer at totalkostnad, milepæler eller systemets/leveransens virkemåte endrer seg, skal i utgangspunktet endringshåndteres. Et eksempel på dette kan være at man prioriterer inn et nytt krav til fordel for et eksisterende element fra produktkøen. Selv om dette gjøres kostnadsnøytralt, det vil si at kostnadsøkningen fra det nye kravet kompenseres ved å ta ut et annet krav at tilsvarende verdi, skal det i henhold til de generelle kontraktsbestemmelsene endringsmeldes ettersom beslutningen innebærer at sluttleveransens virkemåte endres.

Vi vil imidlertid anbefale å håndtere disse kostnadsnøytrale endringene på en enklere måte. Dette kan gjøres ved å utnytte at produktkøen fra LBF er en kontraktsfestet beskrivelse av det funksjonelle omfanget i prosjektet. Partene kan legge ved oppdatert og versjonert produktkø fra hvert kontrollpunkt som vedlegg til kontraktsgrunnlaget. Produktkøen med versjonsnummer må i så fall signeres av begge parter.

Fremgangsmåten beskrevet her må gjøres eksplisitt klart i kontraktsteksten som en endring eller tillegg til de generelle kontraktsbestemmelsene i del I.

5.9 Systemtesten

Et av kjennetegnene ved smidig metodikk som scrum er at testingen kan utføres med en gang det foreligger en demonstrerbar versjon av systemet. Dette samsvarer med prinsippet om å teste så tidlig som mulig for tidligst mulig å avdekke feil, noe som igjen fører til lavere totalkostnad. I og med at produksjonsklare komponenter foreligger ved hver sprintslutt, og gjerne hyppigere, så vil deler av det vi tradisjonelt har forbundet med systemtest kunne skje tilnærmet fortløpende gjennom hele utviklingsprosjektet. Dette betyr at man med smidig gjennomføring bør kunne forvente at leverandørens avsluttende systemtest blir mindre omfattende enn det vi tradisjonelt har opplevd.

Likevel vil det i forkant av en større leveranse være nødvendig å konsolidere løsningen og utføre en mer gjennomgående test. Denne testen skal adressere både funksjonelle og ikke-funksjonelle aspekter ved den utviklede løsningen. Livssyklus-testing, verdikjedetesting og lignende skal bidra til å identifisere feil og mangler i grensesnittene og sammenhengene mellom forskjellige områder.

Systemtesten kan med fordel gjennomføres som en eller flere sprinter.

En del av arbeidet i utviklingssprintene er å utforme testscenarier. Dette utføres fortløpende i sprintteamene og parallelt med utviklingen. I forkant av den avsluttende systemtesten vil det følgelig foreligge en samling av testscenarier og automatiserte tester som skal dekke testbehovet til den foreliggende løsningen. Testscenarier og ikke-funksjonelle elementer som skal testes, kan inngå i en egen logg kalt test-produktkø. Alle elementene i test-produktkø skal være koblet med og understøtte testing av elementer i produktkøen.

Elementene i test-produktkøen bør i likhet med elementene i produktkøen være prioritert. Vi anbefaler en risikobasert prioritering av testobjektene. I korthet går dette ut på at hvert

testobjekt vurderes med hensyn til sannsynligheten for at det vil forekomme feil på testobjektet og konsekvensen av at det oppstår kritiske feil på testobjektet i produksjon. Testprioriteringen gjøres i henhold til sum eller produkt av sannsynlighet og konsekvens.

Planleggingen av testene blir mer effektive dersom det under utarbeidelse av utviklingsdokumentasjon (brukstilfeller, brukerhistorier, forretningsregler med videre) fokuseres på at det skal være mulig å teste etter beskrivelsene senere. Mye tid er spart dersom prosjektet slipper å skrive det samme i både utviklingsdokumenter og testdokumenter.

Gjennomføringen av systemtesten kan altså organiseres som flere systemtest-sprinter. Man bygger opp sprinttestloggene ved å ta fra toppen av test-produktkøen. Det bør være en egen systemtest-sprint som ivaretar ikke-funksjonell test, samt en egen systemtest-sprint som utfører feilretting i parallell med testingen. På denne måten vil testteamene avdekke feil som går inn i feilloggen og feil i feilloggen tas unna av feilrettingsteamene etter prioritet. Restansene fra test-sprintene vil danne grunnlaget for vurdering av oppstart av godkjenningsprøven.

6 Godkjenningsprøven

PS2000 åpner for at kunden kan produksjonssette og ta i bruk deler av løsningen underveis i prosjektet (delvis overtakelse), før partene har nådd kontraktsmilepælen HMP2. I det følgende beskriver vi noen råd for gjennomføring av den formelle godkjenningsprøven mellom HMP2 og HMP3.

På samme måte som med systemtesten, er mange av de tradisjonelle aktivitetene i kundens godkjenningsprøve også distribuert ut over i de enkelte sprinttestene i konstruksjonsfasen. Dette tilsier også at gjennomføring av godkjenningsprøven burde forløpe mer smertefritt enn det man har vært vant til.

I motsetning til tidligere er leverandørens kostnader til godkjenningsprøven nå et relativt påslag til målprisen. Dette påslaget bør ta høyde for følgende aktiviteter:

- Støtte til kundesiden i tilrettelegging for godkjenningsprøven (testmiljøer, bygging og konfigurasjonsstyring, overføring av kompetanse innen migrering/deployment/ installasjon, testdata, samt siling og prioritering av testobservasjoner om mangler og feil). Partene kan avtale et timetak på denne bistanden, for eksempel 10 timer pr. uke i godkjenningsprøven
- Det bør arrangeres et felles møte mellom testledere på kunde- og leverandørsiden jevnlig (gjerne daglig) for å gå gjennom meldte feil og mangler. Ofte er det snakk om dubletter eller forhold som har vært oppe tidligere. Det kan videre være snakk om byggefeil, nedetid på servere og grensesnitt eller andre tekniske forhold, feil på testdata eller feil på testprosedyrer. Testobservasjoner som ikke skal følges opp som ordinær feil eller mangel, er det viktig at man får identifisert så tidlig som mulig. Reelle feil og mangler må konsekvensvurderes og prioriteres så raskt som mulig. De daglige møtene mellom testlederne kan arrangeres som daglig scrum
- Det kan være snakk om testobservasjoner som går ut over omforent funksjonelt omfang. Av denne grunn bør produkteier og leverandørens løsningsansvarlig involveres i verdikjeden så tidlig som mulig med sikte på avklaring om meldingen skal håndteres i godkjenningsprøven, eller prioriteres for vedlikeholdsløpet. Deltagelse på de daglige møtene mellom testlederne kan anbefales
- Det bør planlegges med feilrettingspatcher i løpet av godkjenningsprøven, for å ta høyde for å kunne levere retting av høyt prioriterte feil og mangler. Hvor ofte dette skal skje, er litt avhengig av størrelsen på prosjektet og kompleksiteten i løsningen. Vi har erfaring fra prosjekter der det har vært lønnsomt med opp til to slike patcher ukentlig. Med fullt ut smidig gjennomføring i prosjektet vil det etter all sannsynlighet ikke være behov for så tett patching, men det er ikke noe problem å avlyse en planlagt patch dersom det ikke er behov for den
- I forbindelse med planlagte feilrettingspatcher gjelder selvsagt alle kvalitetskrav til leveransene: at hver melding er ledsaget av en verifikasjonsprosedyre, at hver leveranse er enhetstestet og systemtestet fra leverandørens side (inkludert regresjonstest), med videre

Kunden må avveie hvor mye av leveransen som skal regresjonstestes. Som for leverandørens systemtest, vil vi anbefale en risikobasert tilnærming til prioritering av testobjektene. Som et minimum må alle tester gjennomføres som ikke tidligere er blitt gjennomført (kan for eksempel dreie seg om fullt ut realistiske tester av eksterne grensesnitt, ytelse, konvertering, vedlikeholdbarhet, og avbrudd/gjenoppretting). Når det gjelder samlingen av manuelle

testtilfelle som er aggregert i løpet av prosjektperioden, bør de gjentas i henhold til en prioritert orden. Testobjekter med stor feilfrekvens fra systemtesten er kandidater for høy prioritering. Det bør også i godkjenningsprøven gjennomføres fritesting med nye sluttbrukere for å avdekke eventuelle nye forhold eller problemer med å ta i bruk programvaren. Alle automatiserte tester som er produsert i prosjektperioden kjøres selvsagt som tidligere i forbindelse med bygging.

Et særegen aktivitet i godkjenningsprøven kan være installasjonstesting ved kundens eget IT-personell eller driftspartner, dersom vedkommende ikke har vært involvert tidligere i prosjektet. Denne testen kan for eksempel gjennomføres i forbindelse med installasjon av selve testmiljøet for godkjenningsprøven. Dette er viktig for å verifisere installasjonsdokumentasjonen og for innfasing av drifts- og forvaltningspersonell.

Godkjenningsprøven bør være så produksjonslik som mulig. Det gjelder miljø, eksterne grensesnitt og testdata. Dersom det er snakk om en helt ny løsning, vil man ofte kunne kjøre godkjenningsprøven på det kommende produksjonsmiljøet. Hvis det er snakk om videreutvikling på en eksisterende plattform, må det tilstrebes å gjøre godkjenningsprøven så produksjonsnær som mulig. Ikke minst gjelder dette bruk av produksjonsdata som testdata.

Godkjenningsprøven må ta høyde for verifikasjon av øvrige leveranser i tillegg til selve programvaren. Det gjelder slike ting som systemdokumentasjon, brukerdokumentasjon (dersom leverandøren har hatt ansvar for dette) og driftsdokumentasjon inkludert installasjonsdokumentasjon, utrullingsplaner, opplæringsplaner med videre.

I godkjenningsprøven må kunden verifisere at egne leveranser er tilfredsstillende for produksjonsstart. Det gjelder slike ting som for eksempel etablert forvaltningsplan (se kapittel 7.3), definerte prosedyrer og roller for drift, forvaltning og brukerstøtte, at opplæring er i henhold til plan, at brukersteder er informert om og forberedt for utrulling.

7 Produksjonssetting

Mye av det som står i dette kapitlet er gyldig for IKT-prosjekter under PS2000 generelt. Det som er spesielt for smidig systemutvikling er blant annet

1) Den sterke kundeinvolveringen gjør at man kan tenke forvaltning allerede tidlig i utviklingsprosjektet
2) Større mulighet for å produksjonssette programvare tidligere i prosjektperioden
3) Det at produkter skal være kjørbare, gjør at man tidligere får drifts- og forvaltningserfaring enn i tradisjonelle prosjekter
4) Det vil ved avslutningen av prosjektet foreligge en levende produktkø, som kan ligge til grunn for fortsatt smidig metodikk i vedlikeholdsløpet

Som allerede nevnt i tidligere kapitler, kan partene avtale hyppige produksjonssettinger, blant annet for å redusere risiko knyttet til systeminnføring. Det vi skriver i avsnittene om systeminnføring og forvaltning er ment å kunne anvendes enten prosjektet har en eller flere produksjonssettinger.

7.1 Systeminnføring

Systeminnføring har en teknisk, en funksjonell og en organisatorisk side. Det skal fases inn ny teknologi, kanskje med ny infrastruktur, driftsmønstre og lignende. I tillegg skal det forvaltes et eller flere nye systemer i organisasjonen med blant annet systemeierskap og brukerstøtte. Videre skal det muligens innføres nye prosesser og måter å arbeide på, samtidig som det skal arrangeres opplæring og trening av et stort antall medarbeidere. Det meste av dette er oppgaver som skal gjøres i linjen.

Vi har gode erfaringer med at kunden organiserer et eget mottaksprosjekt, eventuelt et eget mottaksteam i utviklingsprosjektet, for å forberede og gjennomføre innføring av systemleveransen i linjen. Under mottaksprosjektet kan det ligge ansvar for

- Opplæring av forvaltningsroller og sluttbrukere
- Etablering av forvaltningsorganisasjon
- Plan og gjennomføring av utrulling av programvaren
- Plan og gjennomføring av drift og overvåkning av ny løsning
- Plan og gjennomføring av gevinster

Noen steder har mottaksprosjektet hatt ansvar for forberedelser og gjennomføring av kundens godkjenningsprøve.

Organiseringen av et eventuelt mottaksprosjekt setter imidlertid ikke linjeansvaret til siden. Det er linjen som har det overordnete ansvaret og ofte også det operative og utførende ansvaret, for innføringen av ethvert system. Det er kun linjen som sitter med helhetsansvaret, et ansvar som går utover både ansvaret til hvert prosjekt og leverandørens ansvar.

Vi vil anbefale at kundeorganisasjonen etablerer klare retningslinjer for innføring av nye løsninger, dersom det ikke allerede er på plass. Som et minimum må det stilles krav til at operasjonell risiko er vurdert og akseptert før produksjonssetting, likeledes at avtaleverk og forvaltningsplan (se kapittel 7.3) er kvalitetssikret og godkjent. I tillegg kommer slike ting som godkjenning av opplæringsopplegg, utrullingsplaner, gevinstplaner med videre.

7.2 Garantiperioden

Som under godkjenningprøven skal leverandøren estimere sitt bidrag til garantiperioden som et fast påslag til målprisen. Dette påslaget vil blant annet avhenge av garantiperiodens lengde. Vi vil anbefale en standard garantiperiode på 3 måneder og at partene har tegnet opsjoner på vedlikeholdsavtale og rammeavtale for videreutvikling. Rammeavtalen for videreutvikling kan da typisk utløses i forbindelse med utgangen av garantiperioden, eventuelt konkurranseutsettes den for seg i god tid før utløpet av gjeldende avtaler. I de tilfellene der partene tegner opsjon på vedlikeholdsavtale og rammeavtale bør alle de sentrale merkantile betingelsene, som tjenesteinnhold, tjenestenivåer, priser og sanksjonsbestemmelser, være avtalt før kontrakt om utvikling inngås.

Når leverandøren priser det faste tillegget for garantiperioden, bør det kalkuleres med retting av feil og mangler, samt test og overlevering av disse. Hvilke rutiner som gjelder i garantiperioden bør avklares med kunden før signering av kontrakten. Dette er, for eksempel rutiner i forhold til felles møter, rapportering og eventuell annen administrasjon.

7.3 Forvaltning

Før produksjonssetting av enhver løsning bør det foreligge en forvaltningsplan. Denne planen bør minimum inneholde alle viktige roller og ansvar, prosesser og rutiner som gjelder for forvaltning og drift av løsningen.

Vi har opplevd at forankring av forvaltningsoppgavene på fagsiden eller forretningssiden, er et svært sentralt suksesskriterium. Vi vil derfor sterkt anbefale at forvaltningsregimet legges til en systemeierrolle eller tilsvarende fra fag/forretningssiden. Denne systemeierrollen bør ha det overordnede ansvaret for strategisk planlegging, avtalehåndtering med tredjepart (blant annet leverandøren og driftsansvarlig), første- og andrelinje brukerstøtte, opplæring av sluttbrukere, prioritering, bestilling, godkjenning og beslutning om produksjonssetting av endringer med videre.

Prosesser og rutiner bør basere seg på beste praksis, slik de blant annet er nedfelt i ITILs rammeverk. Det gjelder maler for tjenestenivå-avtaler SLA, rollen som service level manager, samt prosesser og rutiner for helpdesk, incident, problem, change, release, configuration, continuity og security management. De generelle retningslinjene i rammeverket må tilpasses organisasjonens særegne behov. Egen erfaring gjør at vi understreker viktigheten av å opprettholde forretningsperspektivet på all forvaltning, slik at ikke prosesser, roller og ansvar for fagapplikasjoner blir delegert bort til IKT-funksjonene i organisasjonen.

Første-, andre- og tredjelinje brukerstøtte må være godt definert og dokumentert på en lett tilgjengelig måte, for eksempel på intranettet. Fra førstelinje skal det være enkelt å rute videre til riktig adresse i andrelinje avhengig av hendelsens art. Det vil vanligvis være tilstrekkelig å skille mellom funksjonelle forespørsler og rent tekniske forespørsler. Tekniske forespørsler omfatter nettverk, svartider, tilgjengelighet, serverbrudd, problemer med kryptering, grensesnittog tilsvarende.

Det vil ofte være regningssvarende å benytte et verktøy for hendelseshåndtering i forvaltningen. Sluttbrukere og andre kan melde inn hendelser, feil og mangler via et webgrensesnitt og meldinger rutes videre pr. mail eller annen varsling til de rette

mottakerene. Verktøyet bør som minimum støtte historikk, statussetting og arbeidsflyt for flytting av eierskapet på meldingen underveis i verdikjeden. Forvaltningsapparatet vil da ha full sporbarhet med endringers historie, status, eierskap og eventuelle vedlegg med konsekvensvurderinger, designdokumenter og testdokumenter.

Utviklingsprosjektets opparbeidede rutiner for test og kvalitetssikring må videreføres i forvaltningsregimet. For større prosjekter vil kundens testressurser fra prosjektet ofte kunne inngå som superbrukere, brukerstøtte og testere i forvaltningsregimet.

8 Referanser

[1] K. Moløkken-Østvold and M. Jørgensen, "A Review of Surveys on Software Effort Estimation," in *2003 ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2003)*, Frascati, Monte Porzio Catone (RM), ITALY, 2003, pp. 220-230.

[2] K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. E. Hove, "A Survey on Effort Estimation in Norwegian Software Industry," in *10th International Symposium on Software Metrics*, Chicago, Illinois, USA, 2004, pp. 208-219.

[3] K. Moløkken-Østvold and M. Jørgensen, "A Comparison of Software Project Overruns – Flexible vs. Sequential Development Models," *IEEE Transactions on Software Engineering*, vol. 31, pp. 754-766, 2005.

[4] M. Iansiti and A. MacCormack, "Developing Products on Internet Time," *Harvard Business Review*, pp. 107-117, 1997.

[5] C. Larman and V. R. Basili, "Iterative and Incremental Development: A Brief History," *IEEE Computer*, pp. 2-11, 2003.

[6] E. L. May and B. A. Zimmer, "The Evolutionary Development Model for Software," *Hewlett-Packard Journal*, vol. 47, pp. 39-45, 1996.

[7] D. R. Graham, "Incremental Development and Delivery for Large Software Systems," in *Software Engineering for Large Software Systems*, B. A. Kitchenham, Ed.: Elsevier, 1990.

[8] T. Gilb, "Estimating Software Attributes: Some Unconventional Points of View.," *ACM Sigsoft Software Engineering Notes*, vol. 11, pp. 49-59, 1986.

[9] H. D. Mills, "Software Development," *IEEE Transactions on Software Engineering*, vol. 2, pp. 265-273, 1976.

[10] A. Cockburn, "In Search of Methodology," in *Object Magazine*. vol. 4, 1994, pp. 52-56.

[11] A. Cockburn, *Surviving Object-Oriented Projects*: Addison-Wesley., 1998.

[12] B. Boehm, C. Abts, A. Winsor Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, *Software Estimation with COCOMO II*: Prentice-Hall, 2000.

[13] A. D. Teasley, L. A. Covi, M. S. Krishnan, and J. S. Olson, "Rapid Software Development through Team Collocation," *IEEE Transactions on Software Engineering*, vol. 28, pp. 671-683, 2002.

[14] K. Moløkken-Østvold and M. K. Furulund, "Analyzing Agile Practices - The Relationship between Customer Collaboration and Software Project Overruns," in *Agile 2007*, Washington DC, USA, 2007, pp. 72-83.

[15] K. Beck and M. Fowler, *Planning Extreme Programming*: Addison-Wesley, 2001.

[16] K. Schwaber, *Agile Project Management with Scrum*: Microsoft Press, 2004.

[17] M. Cohn, *Agile Estimating and Planning*: Addison-Wesley, 2005.

[18] J. Sutherland, C Jackobsen, K Johnson; *Scrum and CMMI Level 5: The Magic Potion for Code Warriors*, http://jeffsutherland.com/scrum/Sutherland-ScrumCMMI6pages.pdf

[19] Mary Poppendick, *Lean Software Development*: Addison-Wesley, 2003.

[20] G. Schneider and J.P. Winters, *Applying Use Cases*: Addison-Wesley, 1998

INNHOLDSFORTEGNELSE VEDLEGG

FIGURFORTEGNELSE VEDLEGG

1) Differensiering mellom forskjellige typer utviklingsmetodikk

Terminologien som benyttes for å beskrive utviklingsmetodikker kan være forvirrende. For eksempel kan det være forvirring relatert til hva som skjuler seg bak begreper som iterative, evolusjonære, smidige og inkrementelle utviklingsmetodikker. Et forsøk på å differensiere mellom disse på et overordnet nivå er presentert i dette vedlegget.

I litteraturen gjøres det ofte et skille mellom de tradisjonelle sekvensielle utviklingsmetodikkene, som regel manifestert gjennom fossefallsmetoden, og de andre mer fleksible, ikke-sekvensielle utviklingsmetodikkene. Dette er en grov grenseoppgang, for eksempel benyttet av Iansiti og MacCormack [4] for å redusere klassifiseringsproblemer. De smidige metodikkene tilhører sistnevnte kategori av fleksible, ikke-sekvensielle metodikker.

Fleksible utviklingsmetodikker har eksistert en god stund. Grunnleggende prinsipper ble faktisk formulert så tidlig som på 1930-tallet. Imidlertid har ikke disse utviklingsmetodikkene blitt adoptert i stor grad av utviklere, forfattere og kunder inntil nylig [5].

2) Smidig utvikling og prosjektstyring

Siden 1970-tallet er det blitt hevdet at bruken av fleksible utviklingsmetodikker kan bidra til å redusere overskridelser sammenlignet med sekvensielle utviklingsmetodikker [6-9]. Et nylig publisert forskningsarbeid fant at prosjekter som benyttet en fleksibel utviklingsmetodikk (smidig, evolusjonær og/eller inkrementell) hadde gjennomsnittlige overskridelser langt lavere enn for sammenlignbare prosjekter som benyttet en sekvensiell utviklingsmetodikk [3]. Gjennomsnittlig overskridelse i arbeidsmengde var 24% for prosjektene som benyttet en fleksibel metodikk, sammenlignet med 55% for de som benyttet en sekvensiell metodikk.

Prosjektene i de to sammenlignede gruppene var for øvrig like med tanke på viktige parametere som estimeringsmetode, prosjektstørrelse, kundemodenhet og mengde av levert funksjonalitet. Intervjuer med prosjektledere fant at de fleksible prosjektene skapte et samarbeidsklima som ga gode relasjoner mellom kunde og leverandør i langt større grad enn de sekvensielle prosjektene [3]. Dette støtter opp om de som har hevdet at utviklingsmetodikker som baserer seg på en inkrementell tankegang, herunder de smidige, medfører bedre ledede prosjekter [7, 10-12]. En viktig effekt at man kan benytte tilbakemeldinger man får fra tidlige inkrementer til å justere kursen [7].

Studier har vist at opptil 70% av total tid i IT-prosjekter benyttes til samarbeid og kommunikasjon [13] (avhengig av prosjektets størrelse og omfang). Smidig prosjektstyring er orientert mot samarbeid mellom kunde og leverandør for å oppnå prosjektmål og unngå uønskede hendelser, som overskridelser og prosjektavbrudd. Disse aspektene ved smidig utvikling er også de mest sentrale i denne veilederen, ettersom den er ment å understøtte samarbeid mellom kunde og leverandør.

Et studie nylig gjennomført hos et norsk konsulenthus tok sikte på å undersøke effekten av samarbeid mellom kunde og leverandør [14]. Der ble det avdekket at i de prosjektene hvor man hadde en daglig interaksjon mellom kunde og leverandør, hadde man gjennomsnittlige overskridelser i arbeidsmengde på 9%. Tilsvarende hadde man overskridelser i arbeidsmengde på 58% i de prosjektene hvor man ikke hadde daglig interaksjon. Den samme studien avdekket også positive erfaringer ved å benytte seg av kontrakter basert på målpris.

Viktigheten av samarbeid mellom kunde og leverandør gjennomgås og forklares i sentrale tekstbøker om smidig utvikling, for eksempel XP [15], Scrum [16] og smidig estimering og planlegging [17]. Mike Cohn utdyper:

”*Samarbeid med kunden er verdsatt fremfor kontraktsforhandlinger fordi smidige team ønsker at alle parter i det samme prosjektet arbeider mot de samme settene av mål”* [17].

Beck og Fowler hevder at utviklingsprosjekter må justere kursen kontinuerlig. De skriver videre at:

”Kommunikasjon er poenget. Vi har sett alt for mange nedskrevne kravdokumenter som ikke involverer kommunikasjon” [15].

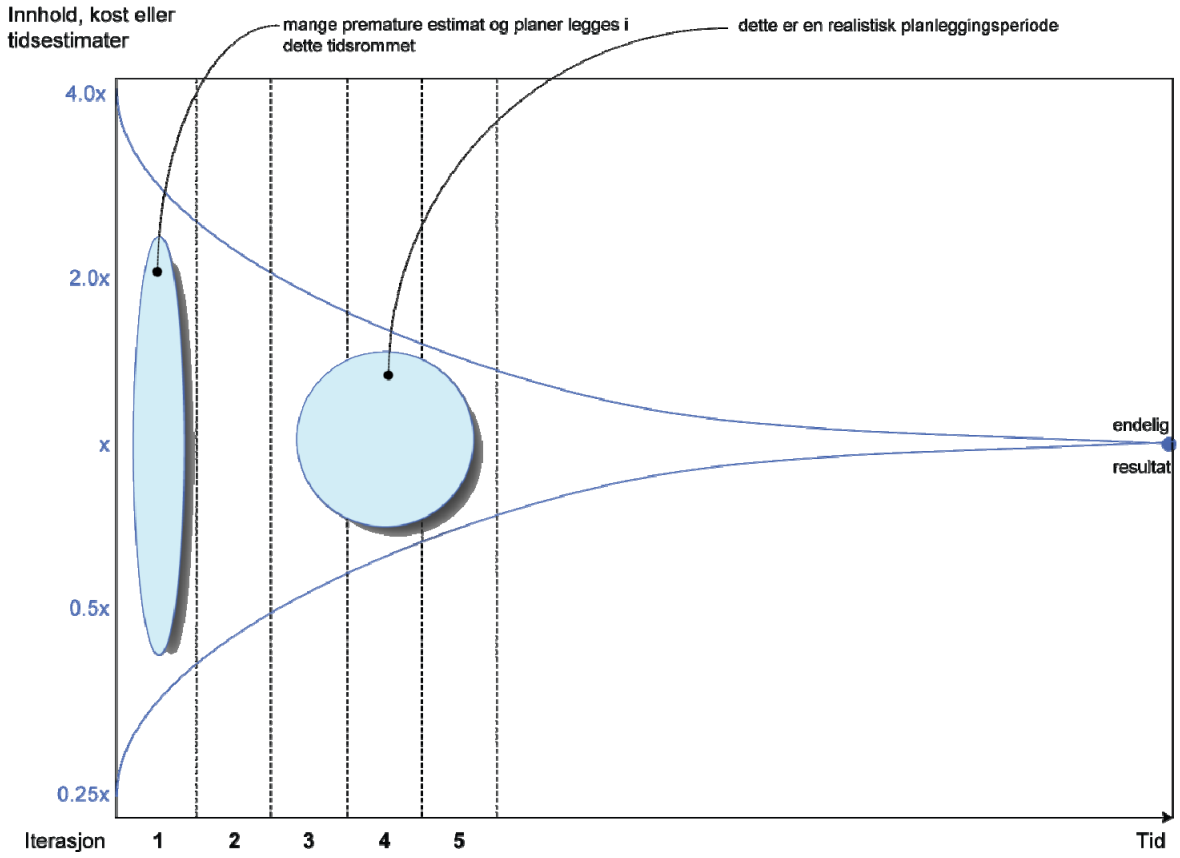
Hyppig kommunikasjon kan benyttes til å prioritere funksjonalitet, sette fokus på feilretting eller å inkludere mer funksjonalitet [15].

3) Om planlegging

Estimering av løsningsforslagene, utført av leverandørene, er ett av underlagene til planlegging. Uten estimater kan planer ikke legges, men planlegging er mer enn bare å estimere arbeidets omfang og tidsforbruk.

Planlegging er vanskelig og planene er ofte feil. Det er vanlig å angripe denne problemstillingen på ekstreme måter, enten ved ikke å planlegge i hele tatt eller å legge så mye arbeid ned i planlegningen at alle impliserte ender opp med å være overbevist om at planene må være riktige.

At estimering og planlegging er vanskelig er ikke nytt. Det kommer blant annet til uttrykk i usikkerhetstrakten (’cone of uncertainty’), jamfør figur 1. I 1981 tegnet Barry Boehm første versjon av hva Steve McConnell (1998) senere kalte ”cone of uncertaincy” (usikkerhetstrakten).



Figur V1: Usikkerhetstrakten

Iterativ, og spesielt smidig, utvikling innebærer at krav, planer, estimater og løsning utvikler seg eller forfines over iterasjonene, framfor at de er fullt ut definert og frosset etter et tidlig, stort spesifikasjonsarbeid før utviklingen begynner.

Adaptiv utvikling innebærer at elementer tilpasser seg som respons på tilbakemeldinger fra tidligere arbeid; tilbakemelding fra brukere, utviklere, testere også videre. Hensikten er den samme som for iterativ utvikling, men navnet hentyder at det legges mer vekt på tilbakemelding-responsmekanismen i utviklingen.

I iterativ og adaptiv utvikling er det ikke slik at krav for alltid er konstant i endring. De fleste krav defineres og forfines i de tidlige iterasjonene. Videre legges det vekt på å forstå de kravene som har størst arkitekturmessige konsekvenser eller som gir høyest forretningsverdi først. Partene er likevel klar over at enkelte krav vil endres eller kommer til underveis i utviklingen, uten at dette skaper annet enn regulær endringshåndtering.

I en tidlig fase er usikkerheten høy, men den minker etter som tiden går og informasjon akkumuleres. Det er dette som illustreres av usikkerhetstrakten**.** Den smidige løsningen på denne usikkerheten er å vente med å fastsette forventningene til kostnader, innsats og plan til man har gjennomført noen Sprinter, kanskje 10 % til 20% inn i prosjektet. Dessverre kan dette være vanskelig å få til i praksis, da det forventes at de som har ansvaret for prosjektet er rimelig sikre på sine anslag innen det gis klarsignal innen oppstart.

Videre oppfordrer vi til adaptiv framfor prediktiv planlegging. Det vil si at det lages bare detaljerte planer for en kort tidshorisont, slik at detaljeringsnivå og forpliktelser er i samsvar med kvaliteten på informasjonen som foreligger.

I PS2000-kontrakten er LBF svaret på utfordringene i usikkerhetstrakten. Det er viktig at denne fasen brukes for det den er verdt for å bli virkelig kjent med løsningen som skal lages og for å redusere usikkerheten i estimater og planer som man nødvendigvis vil ha med seg inn fra kontraktsfasen.

4) Oversikt over ulike metoder

I løpet av det siste tiåret har smidige, iterative utviklingsmetoder blitt svært populære og det er skapt mye hype omkring dem. Overfloden av metoder som er utviklet kan virke skremmende: agile, incremental, iterative, evolutionary, lean og extreme er noen av termene som er brukt. Det er derfor ikke enkelt å velge riktig metode til problemstillingen man skal løse. Et prosjektteam bør ha fokuset på å være i stand til å takle prosjektets risiki. Valg av riktig utviklingsmetodikk er en viktig aktivitet tidlig i utviklingsløpet.

Sammenlignet med fossefallstilnærmingen og RUP har alle smidige metoder det til felles at leveransen etter hver sprint er en integrert, testet kjørbar løsning. Alle utviklingsprosesser (analyse og design, koding, integrasjon og test) gjøres innenfor hver iterasjon.

Forskjellene mellom metodene finner vi innenfor hvilke risiki de fokuserer på, lengden på iterasjonene, mengden dokumentasjon og hvor viktig modellering anses å være. Oversikten under presenterer kort de mest kjente metodene, ikke bare de smidige, deres hovedfokus og og referanse til mer informasjon.

I Spiral Model (Boehm 1985)

Selv om Barry Boehm ikke var den første til å lage en modell for iterativ utvikling var han den første til å forklare hvorfor iterasjoner er viktig. Spiralmodellen har eksplisitte aktiviteter for å identifisere risiki, håndtere dem med prototyping og evaluere dem før man går videre til neste iterasjon.

For mer informasjon:

A Spiral Model of Software Development and Enhancement, by Barry Boehm, IEEE Computer, May 1988

Wikipedia: Spiral Model

II Incremental Development

Ofte brukt sammen med iterativ utvikling. Hovedfokus er etappevis integrasjon. Hovedrisikoen med ”big bang” integrasjon mot slutten av utviklingsløpet håndteres. I en tradisjonell fossefallorientert organisasjon har inkrementell utvikling den fordelen at man kan nøye seg med å arbeide iterativt i de siste fasene av utviklingsløpet, koding, integrasjon og test, mens krav, analyse og design gjøres på tradisjonell fossefallmåte. Dette er lett å selge inn både i egen organisasjon og mot forretningsforbindelser. Ulempen med denne tilnærmingen er at man blir svært fastlåst i forhold til endringer i krav til løsningen.

Med en fullstendig iterativ prosess kan man også gjøre kravaktiviteter i hver iterasjon, men det vil kreve mer involvering fra kunden.

For mer informasjon:

Wikipedia: Iterative and incremental development

III Rational Unified Process (RUP)

RUP er en svært utbredt iterativ utviklingsprosess utviklet av IBM-Rational. Prosessen er arkitektur-, brukstilfelle- og risikodrevet i riktig balanse. Sammenlignet med smidige metoder er det lagt mer vekt på dokumentasjon og visuell modellering. IBM-Rational tilbyr en komplett pakke av verktøy, malverk og webbasert prosessbeskrivelse for å underbygge prosessen. RUP kan skreddersys og skaleres og det er kanskje også det største ankepunktet mot prosessen. RUP oppleves overveldende for små prosjekter.

For mer informasjon:

The Rational Unified Process: An Introduction, Philippe Kruchten,

Wikipedia: IBM Rational Unified Process

IV eXtreme Programming (XP)

XP er den mest kjente av de smidige metodene. Målet er å håndtere skiftende krav. Det legges vekt på kommunikasjon, forenkling, tidlige leveranser og testing. Metoden baserer seg på noen grunnleggende prinsipper, testdrevet utvikling (TDD), kontinuerlig integrasjon, parprogrammering, små leveranser, kundetest, enkelt design, refaktoring, metaforer, kollektivt eierskap, kodestandard, "planning game", hele team og utholdelig tempo. XP er svært effektivt og produktivt for små, godt kvalifiserte team. Metoden er relativt vanskelig å skalere til store distribuerte team.

For mer informasjon:

www.xprogramming.com (Ron Jeffries)

eXtreme programming - a gentle introduction

Extreme Programming Explained, Kent Beck

Wikipedia: Extreme Programming

V Scrum

En metode som legger vekt på prosjektledelse og teamarbeid. Scrum er et enkelt "inspiser og tilpass" rammeverk som har tre roller, tre seremonier og tre artefakter utviklet for å levere fungerende programvare i sprinter, vanligvis 30-dagers iterasjoner.

1. Roller: produkteier, scrumleder, scrumteam
2. Seremonier: sprintplanlegging, sprinttilbakeblikk, og daglig scrum
3. Artefakter: produktkø, sprintkø, og brenndiagram

Scrum er den metoden vi fokuserer på i denne veilederen og vi går derfor lenger i å definere scrum i dette vedlegget.

For mer informasjon:

www.controlchaos.com/
Agile Project Management with Scrum, by Ken Schwaber, Microsoft Press

Wikipedia: Scrum (development)

VI Dynamic Systems Development Method (DSDM)

PS2000 bygger på hovedprinsippene i denne metoden, fokus på å holde milepæler og målpris (time-box prinsippet). DSDM har 9 prinsipper som vi kjenner igjen i andre smidige metoder;

1. Aktiv brukerinvolvering
2. Team-medlemmene må ha beslutningsmyndighet
3. Hyppige leveranser,
4. 'Fitness for business purpose' – akseptansekriteriet er at leveransen er skreddersydd til organisasjonens behov
5. Iterativ og skrittvis utvikling
6. Alle endringer er reversible
7. Krav spesifiseres på et høyt nivå
8. Testing integreres ut over hele utviklingsløpet
9. Det kreves en atmosfære av gjensidig tillit og samarbeidsvilje

Prosessrammeverket er basert på "Rapid Application Development" (RAD). Metoden har likheter med RUP, men det er mindre fokus på verktøy og teknikker.

For mer informasjon:

DSDM Consortium

Wikipedia: DSDM

VII Crystal Methods (Alistar Cockburn)

Alistar Cockburn er kjent for sitt brennende engasjement for strukturerte brukstilfeller (use case). Hovedvekt på å skreddersy metoden for prosjekter fra 6 til 60 utviklere og på kritikalitet. Det rettes også oppmerksomhet på versjonskontroll og konfigurasjonsstyring.

For mer informasjon:

Crystal Clear: A Human-Powered Methodology for Small Teams, by Alistair Cockburn

Wikipedia: Crystal Clear

VIII Feature-Driven Development (FDD, Peter Coad)

Kombinerer smidige metoder med modelldrevne teknikker med fokus på sluttbrukerfunksjonalitet (feature) "*plan by, design by, build by, and report by feature*"**.** Domenemodellering gjøres i hovedsak på forhånd. Iterasjoner er definert som "feature sets". En feature utvikles etter fossefallsmetoden. Fordi metoden legger vekt på domenemodellering skalerer metoden bra til større team

A Practical Guide to Feature-Driven Development (The Coad Series),

Wikipedia – Feature Driven Development

IX Test-Driven Development (TDD)

Fokuserer på testbare krav og automatiske tester. Utviklingssyklusen er snudd rundt: start med et testtilfelle og en feilende enhetstest før man implementerer koden. Bruk enhetstestverktøy og bygg en samling med regresjonstester mens man går.

For mer informasjon:

Test Driven Development: A Practical Guide, by Dave Astels

Test Driven Development: By Example, by Kent Beck

Introduction to TDD, by S.Ambler

Wikipedia – Test Driven Development

X Component Based Development (CBD)

CBD handler om å lage programvare ved å sette sammen og integrere programvarekomponenter, i stedet for å kode. Fokus er på å redusere utviklingskostnader og å få til rask systemsammensetning. Aktivitetene omfatter; komponentutvelgelse (egnethetstesting) komponenttilpasning, sette sammen komponenter til systemer og systenevaluering

For mer informasjon:

Component-Based Software Development / COTS Integration

XI Lean Software Development

LSD er en oversettelse av lean manufacturing prinsipper og praksis, først utviklet av Toyota, til programutviklingsdomenet. Fokus er på effektivitet og å fjerne unødvendige elementer. scrum bygger på mange av prinsippene fra Lean.

For mer informasjon:

Lean Software Development: An Agile Toolkit for Software Development Managers
Lean Software Development. Deliver Value Quickly, Efficiently, Reliably - Every Time

Wikipedia: Lean Software Development

XII Evo - Evolutionary Project Management

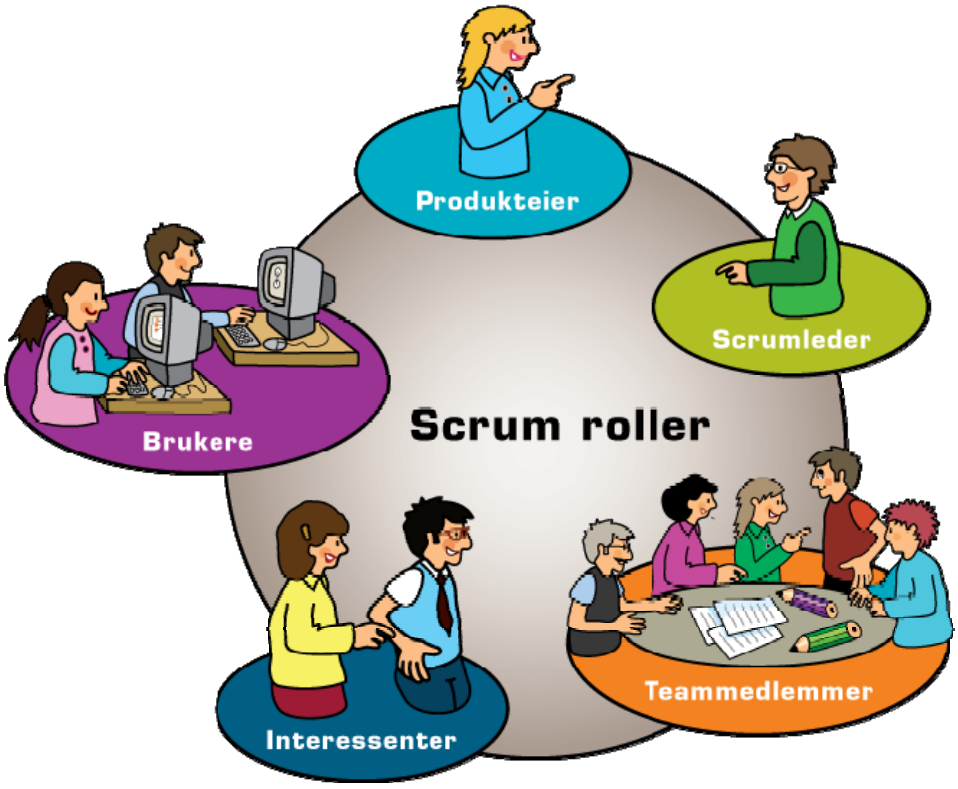
EVO er godt kjent i Norge, i og med at foregangsmannen selv er bosatt her, og at flere bedrifter i Norge har tatt i bruk metoden. Hovedprinsippet i EVO er at alle mål kan kvantifiseres, ikke minst effektmålene i programutviklingsprosjekter.
For mer informasjon:

Tom Gilb www.gilb.com

XIII Waterfall lifecycle - Fossefallmetodikk

Fossefallsmetodikken er bare egnet på lette prosjekter med oversiktlige krav og kjent løsning. Slike prosjekter er det få av nå til dags. Eller som Fred Brooks forklarte I sitt nøkkelinnlegg '95 ICSE keynote og sin berømte bok "Mythical Man-Month": "The waterfall model is wrong!".

5) Roller i scrum



Figur V2: Roller i scrum

Scrum har 3 klart definerte roller: produkteier, scrumleder og sprintteam.

I scrum er deltakerene som er ansvarlig for å levere løsningen ofte kalt "griser" (produkteier, scrumleder, sprintteam), mens de som er interessert i resultatet kalles "kyllinger[2]" (Interessenter, brukere). Disse begrepene kommer fra en historie som beskriver forskjellen mellom å være ansvarlig for en oppgave kontra det å være interessert i resultatet.

Produkteier, scrumleder og sprintteam [grisene] er altså de viktigste rollene for å nå et vellykket resultat. En del av scrum prosessen er å ta hensyn til de andres (kyllingenes) behov, deres ønsker, ideer og påvirkning, men ikke på en slik måte at det innvirker eller uroer eller kommer i veien for selve scrum prosjektet. Kyllinger omfatter interessenter, brukere, systemarkitekter og alle andre som er interessert i prosjektet og som kan ha motstridende interesser med prosjektet.

[2] Kylling på norsk er ikke direkte det samme som 'chicken' på engelsk, men kylling er allerede etablert som begrep så det ville bli feil å innføre høne som norsk oversettelse ☺

En av de viktigste egenskapene med en smidig tilnærming er praksisen med å gjøre brukere, forretning og andre interessenter delaktige i prosessen. Det er kritisk at de er engasjerte og gir tilbakemelding på det teamet produserer ved å delta på sprintdemo og ved å teste nye funksjoner i produktet etter hvert som de ferdigstilles.

I Produkteier

Figur V3: Produkteier

Produkteieren er bestiller og påser at sprintteamet jobber med de riktige tingene sett ut fra et forretningssynspunkt (figur 3). Produkteieren kan være en kunderepresentant eller en som representerer en sluttkunde. For et firma som lager produkter er kunden ”markedet” og produkteieren tjener som en representant for markedet. Produkteieren kan gjerne ha en gruppe av støttespillere eller opptre som en gruppe av personer, men da er det gjerne leder av gruppen som kalles produkteier.

En produkteier er ansvarlig for formulering av hensikten med programvaren. og gjerne en plan som viser hvilke gevinster som kan forventes fra produktet innenfor gitte tidsrammer.

Produkteieren har følgende funksjoner:

- Eier av produktkøen, som definerer den funksjonaliteten som skal leveres
- Er ansvarlig for å samle innspill fra brukere, interessenter, og andre med interesse for systemet for å kunne prioritere og innarbeide dette i produktkøen
- Er ansvarlig for å knytte gevinstplanene i prosjektet til prioritering av funksjonalitet i produktkøen

- Prioriterer funksjonalitet med hensyn på forretningsverdi
- Er ansvarlig for å oppdatere produktkøen underveis og før hver ny sprint. Dette gjelder ikke minst ved behov for endringer som medfører ny funksjonalitet, eller ved at tidligere krav må gå ut eller prioriteres lavere

Godkjenner eller underkjenner resultater produsert underveis, eventuelt i en rådgivende funksjon for kundens prosjektleder.

II Produktteamet

I en PS2000 kontrakt er det naturlig at produkteier er en kunderolle, men vi har erfart at produkteier trenger ressurser rundt seg for å kunne forberede produktkøen for sprinting. Vi har derfor plassert leverandørens løsningsansvarlig eller hva vi også kan kalle faglig ansvarlig i produktteamet.

Ansvarsområder for produktteamet

- Definere produktkøen i form av produktelementer
- Klargjøring av produktkøen i forkant av hver sprint. De øverste produktelementene bør i tillegg til beskrivelse, forretningsverdi, prioritet og estimat også inneholde en beskrivelse av hvordan elementet skal verifiseres, en demobeskrivelse, eventuelle testscenarier, krav til testdata og lignende
- Holde produktkøen oppdatert til enhver tid
- Prioritere nye produktelementer i forhold til virksomhetsmessig betydning og forretningsverdi
- Bidra til en fornuftig utviklingsmessig rekkefølge, blant annet ved å planlegge til samme sprint elementer som henger sammen funksjonelt, vurdere avhengigheter til tekniske produktelementer med videre
- Forklare innholdet i, krav og spesifikasjoner som er knyttet til et produktelement for sprintteamet

Leverandørens hovedansvarsområder

Leverandøren har framdriftsansvaret i en PS2000-kontrakt og leverandørens representanter i produktteamet bør derfor ta hovedansvaret for noen av oppgavene:

- Detaljert implementasjonsbeskrivelse og design, for eksempel interaksjonsdesign
- Løpende estimering av nye produktelementer
- Presentere forslag til hvordan de funksjonelle produktelementene skal realiseres i løsningen
- Gi råd om hvordan man kan få til sammenhengende funksjonalitet innenfor hver sprint

- Holde fokus på gjenbruk av løsningselementer
- Avklare tekniske avhengigheter mellom produktelementer og andre komponenter som må produseres
- Gi råd om tekniske endringer som bør foretas og hvilke funksjonelle konsekvenser de har
- Løpende vurdering om oppdateringen av produktkøen har merkantile konsekvenser i form av endring

Kundens hovedansvarsområder

- Løpende beskrivelse av produktelementer, inkludert hvordan de skal verifiseres med krav til demo og testscenarier
- Løpende avklaringer og forankringer i kundeorganisasjonen
- Prioritering av produktelementene
- Godkjenning av produktelementer

Løsningsansvarlig

Dette er en rolle på leverandørsiden, som er ansvarlig for løsningens utforming. Vi har gode erfaringer med at leverandørens Løsningsansvarlig bidrar sammen med produkteier i produktteamet med å tilrettelegge produktkøen i forkant av hver sprint. Andre navn på denne rollen har vært funksjonell arkitekt, ’business analyst’ og lignende.

Andre roller i produktteamet

Det er vanlig i PS2000 prosjekter at både kunden og leverandøren har hver sin testleder. I et større prosjekt der det er flere sprintteam, vil det være naturlig at både kundens og leverandørens testleder inngår i produktteamet, for å tilrettelegge for akseptansekriterier og generering av testbetingelser knyttet til elementene øverst i produktkøen som er kandidater for neste sprint.

III Sprintteamet

FigurV4: Sprintteamet

Sprintteamet er en tverrfaglig gruppe av personer med de ulike ferdighetene som trengs for å skape sluttfunksjonalitet ut av kravene som er beskrevet i produktkøen. Leveransen fra sprintteamet skal være ferdig – funksjonelt komplett, testet, dokumentert og uten kjente feil, i stand til å bli produksjonssatt om ønskelig. Et slikt team består gjerne av noen som kan analysere kravene, lage design, programmere, lage og gjennomføre test, dokumentere med videre. I et PS2000 prosjekt hvor kunde og leverandør samarbeider om å ta fram løsning vil typisk kunden inneha analyse- og test-roller i og med at de er de som kjenner fagområdet best, mens leverandøren typisk innehar de andre rollene som er typiske fagroller innen programvareutvikling, arkitektur og lignende.

Det er sprintteamet som forplikter seg ovenfor produkteier at de vil gjennomføre hver iterasjon/sprint og ferdigstille funksjonaliteten teamet påtar seg ved inngangen til sprinten. På slutten av sprinten viser de fram resultatet av arbeidet som er gjort for produkteier og andre interessenter og produkteier tar dette med seg inn i planleggingen for påfølgende sprinter.

I organisasjoner som utvikler programvare er det vanlig å opprette ekspertgrupper som jobber på tvers av prosjekter og som samler personer med spesialkompetanse innenfor for eksempel brukervennlighet, database, infrastruktur, sikkerhet, systemarkitektur, integrasjon. I den grad slik kompetanse er nødvendig i et scrumprosjekt så bør disse personene gå inn i sprintteamet på vanlig måte. Det er ikke tilrådelig at de står utenfor og kun gir råd, de blir del av teamet når det er nødvendig å lage arkitektur, lage infrastruktur eller etablere databasemodell. De gjør enten jobben selv eller jobber med kompetansebygging innenfor teamet overfor de som gjør arbeidet.

Hovedforskjellen er at de ikke lenger opptrer som kyllinger. Vi får dem inn i teamet når iterasjonen starter og de forplikter seg på samme måte som resten av teamet til å løse oppgavene teamet har påtatt seg i denne sprinten. Som teammedlemmer er de griser på linje med de øvrige.

- Sprintteamet velger blant de høyest prioriterte produktelementene for å bestemme innholdet i neste sprint
- I sprintplanlegging dekomponeres produktelementene til oppgaver
- Sprintteamet er ansvarlig for utvikling og enhetstest
- Sprintteamet velger mål for sprinten og spesifiserer resultatet av arbeidet

- Sprintteamet er ansvarlig for å rapportere daglige resultater, estimert gjenstående på løpende oppgaver og eventuelle hindringer
- Sprintteamet har rett til å gjøre alt som skal til innenfor prosjektets retningslinjer for å nå målet med sprinten
- Sprintteamet organiserer seg selv og arbeidet som skal gjøres
- Sprintteamet demonstrerer resultatet av arbeidet gjennomført i sprinten til produkteier og andre interessenter i sprintdemo
- Sprintteamet oppsummerer sine egne erfaringer og gjør justeringer av prosesssen i sprinttilbakeblikk

IV Scrumleder

FigurV5: Scrumlederen

Scrumlederen (ScrumMaster) er den tredje, viktige rollen. Tidligere var dette prosjektlederen, nå er det personen som er ansvarlig for at scrumprosessen brukes slik intensjonen er. Det er ulike måter å se scrumlederens rolle på. En måte å se på rollen er å sammenligne den med å være en forelder, fordi første gangen du får samlet sprintteamet vet de ikke hvordan de skal organisere seg selv, de vet ikke hvordan de skal jobbe tverrfaglig, de vet ikke hvordan de skal jobbe med prosjekteier eller arbeide innenfor tidsrammer. På samme måte som en forelder er scrumlederen ansvarlig for å lære og trene teamet i hvordan ulike ting gjøres til de klarer det uten hjelp, fra et tidlig uorganisert uselvstendig team til et modent selvfungerende team. Tilsvarende er scrumlederen lik en trener ansvarlig for å motivere, være leder og veileder. Scrumlederen er også lik en dommer som sikrer at spillereglene følges.

Scrumlederens viktigste oppgave er å sikre at alle følger reglene for metoden og at teamet får full fokus på å løse sine oppgaver uforstyrret. Ofte vil prosesser i organisasjon kunne skape hindringer for sprintteamet som igjen påvirker i hvilken grad teamet klarer å levere som planlagt. Det er lett å gi opp og akseptere at oranisasjonsprosesser ikke lar seg endre, og derav også ofre noe av teamets kapsitet, men dette er noe scrumlederen må kjempe i mot. For å kunne oppnå den høye produktivitet som scrum forventes å ha er det essensielt at reglene følges nøyaktig og gjennomføres uansett. Ved å gjøre det vil enhver ting i organisasjonen som ikke fungerer og som hindrer arbeidet med å utvikle programvare bli lett synlig. Det er scrumlederens jobb å fjerne disse hindringene, om de er innenfor eller utenfor teamet, slik at teamet klarer å nå målene de har satt seg ved inngangen til sprinten.

Scrumlederens oppgaver oppsummert:

- Å få til velfungerende sprintteam som har høyest mulig produktivitet. Det betyr opplæring, fasilitering, teamsammensetning, daglig oppfølging, sikre at prosesser følges med videre.
- Sprintplanleggingsmøtet, det medfører også å sørge for at de riktige personene er invitert og delaktige (for eksempel produkteier, medlemmer av produkøteam)
- Sprinttilbakeblikk og for å prioritere forbedringer og sørge for at forbedringer blir utført
- Å skjerme sprintteamet fra eksterne forstyrrelser og fjerne hindringer
- Å invitere relevante personer til daglig scrum, sprintdemo og sprintplanleggingsmøter
- Å lette kommunikasjonen mellom sprintteamet og produkteier slik at produkteier er i stand til mest mulig direkte å styre prioriteringene av funksjonalitet som utvikles
- Å coache prosjektmedlemmer på både leverandør- og kundesiden, slik at de ved å bruke scrum kan få mest mulig igjen for pengene og nå prosjektets mål
- Å få til tett samarbeid på tvers av alle roller og funksjoner
- Å forbedre utviklingspraksis og bruk av verktøy slik at hvert inkrement med funksjonalitet er potensielt produksjonsklar
- Å følge opp fart og fokusfaktor til sprintteamet, slik at forbedringspotensialet tas ut

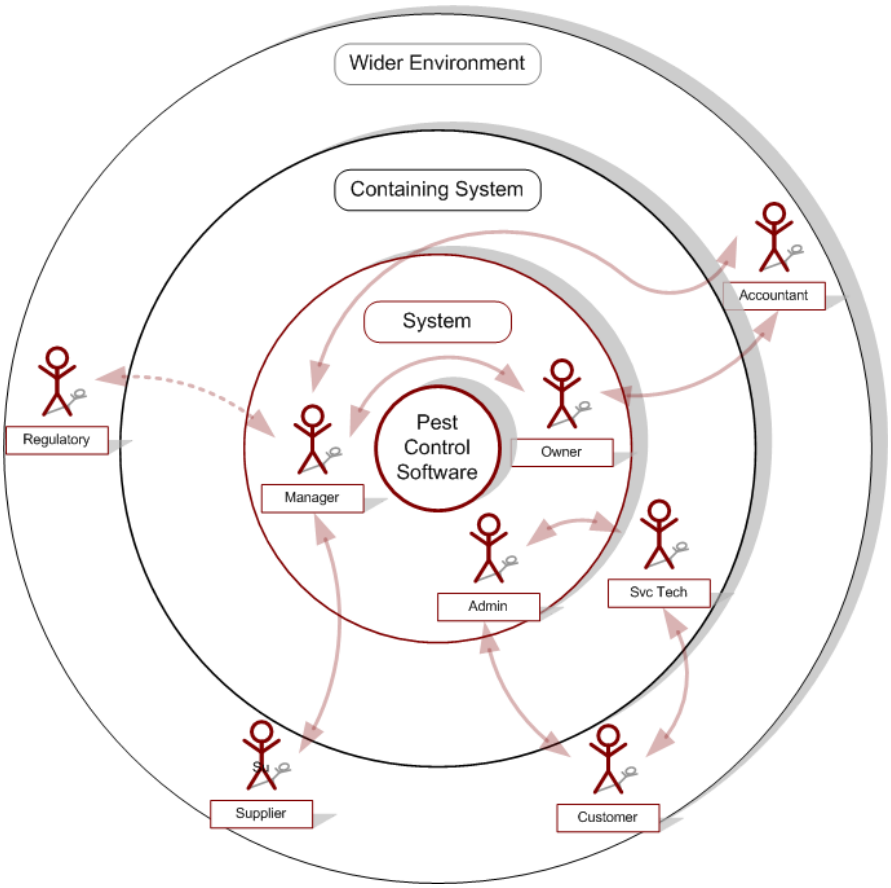
V Interessenter

Figur V6: Interessenter

Det første trinnet i kravanalysen før prosjektoppstart er å identifisere hvem som kan gi deg kravene til hva det planlagte systemet skal inneholde. Forretningsprosesser omfatter kommunikasjon mellom ulike folk i organisasjonen, men kommunikasjon skjer også utenfor organisasjonen. Når man samler krav til løsningen er det lett å overse de som ikke bruker programvaren direkte. Disse ressursene kan likevel være interessenter til løsningen.

Alle potensielle interessenter til prosjektet bør identifiseres så tidlig som mulig. Hvis de ikke har vært involvert i forberedende arbeid har prosjektet bruk for en eller annen form for kommunikasjonskanal for å informere dem om prosjektet og hvordan prosjektet vil gi dem fordeler eller påvirker dem på en eller annen måte.

En teknikk som anbefales for å hjelpe å identifisere interessenter er å bruke et løkdiagram som illustrert i figur 7.



Figur V7: Eksempel på løkdiagram for å identifisere interessenter til en løsning

Den beste måten å takle dette på er å først identifisere alle interessenter og deretter å lage en kommunikasjonsplan som dokumenterer hvordan hver enkelt skal informeres om prosjektet. Planen bør inneholde tidsangivelse, informasjonsinnhold og type kommunikasjon. Så fort planen er på plass er det på tide å gå og snakke med interessentene.

VI Brukere

Figur V8: Brukere

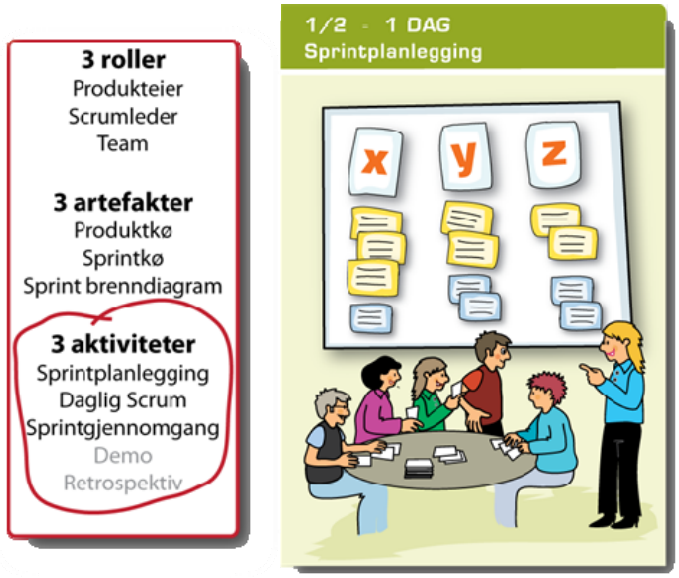
Brukerne er de som skal bruke systemet og er slik sett en spesielt viktig type interessent. Vi mener det er viktig at brukerne deltar i utviklingen av systemet som skal utvikles for at

brukerkrav blir tilstrekkelig identifisert, og at dette er et sentralt suksesskriterium for systemutvikling. Brukerne kan være representert både i sprintteamet og i produktteamet.

6) Aktiviteter i scrum

De tre hovedaktivitetene i scrum er sprintplanlegging, daglig scrum og sprintdemo. I tillegg kommer sprinttilbakeblikk.

I Sprintplanlegging



Figur V9: Sprintplanlegging

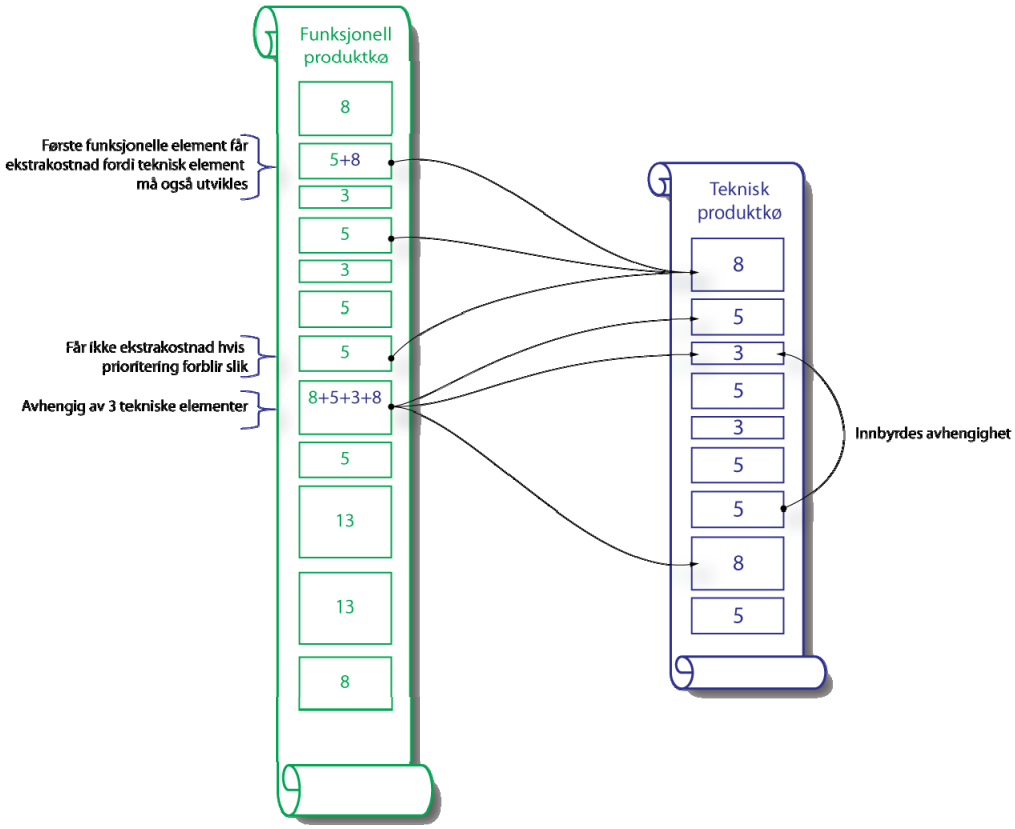
Planleggingsmøtet begynner med at produkteier gjennomgår visjon, veikart, leveranseplan og produktkø med teamet. Produktkøen er gjerne satt opp på veggen i form av lapper som er plassert slik at de viktigste oppgavene er plassert øverst, lengst til venstre eller lignende (se figur 9). Å plukke fra køen er da å faktisk ta lapper fra produktkøveggen og plassere disse over på hvert teams tavle. At dette skjer i samme rom og at alle gjør det samtidig har den effekten av man får diskusjon og at man forhåpentligvis får en optimal fordeling av oppgaver mellom teamene. Er prosjektet delt inn i flere team er det hensiktsmessig at de får plukke oppgaver samtidig for å ta hensyn til inviduelle ferdigheter eller tidligere erfaringer med lignende oppgaver.

Teamet vurderer estimatene på elementene i produktkøen og bekrefter at at de er så riktige som mulig. Teamet bestemmer seg for hvor mange elementer fra produktkøen de kan gjennomføre i sprinten basert på sin kapasitet – teamstørrelse, tilgjengelig tid og nivået på teamets produktivitet. Det er viktig at teamet selv plukker de oppgavene de har kapasitet til å løse innenfor sprintlengden fra toppen av produktkøen.

Når scrumteamet har valgt hvilke topprioriterte elementer fra produktkøen de skal levere i kommende sprint setter teamet seg sammen for å bryte ned produktelementene i sprintoppgaver. Dette er spesifikke utvikleroppgaver som er nødvendige å gjennomføre for å implementere de valgte produktelementene. Denne aktivitetslisten utgjør sprintkøen for

teamet. Denne fasen av sprintplanleggingsmøtet er tidsstyrt til maksimum 1 time pr. uke sprintlengde, det vil si 4 timer for en 4 ukers sprint. Oppgavene brytes ned i maksimum 2 dager, helst 1 dags arbeid. Når sprintkøen er ferdigstilt regnes totalt estimert arbeid sammen og sammenlignes med estimatene fra produktkøen. Er det vesentlige forskjeller må teamet forhandle med produkteier om enten å legge noe tilbake eller hente flere elementer fra produktkøen slik at teamet er sikrest mulig på å klare oppgavene de har tatt på seg i sprinten.

Produkteier har prioritert den funksjonelle produktkøen. For å realisere funksjonelle krav må man også realisere noen tekniske krav. Dette kan man vurdere å håndtere med en egen teknisk produktkø (se figur 10). Vi gjør imidlertid oppmerksom på at det er delte meninger om dette i det smidige miljøet. Noen mener at det å ha egne tekniske elementer bare er en kompensasjon for manglende hell i å formidle forretningsverdien for det man skal gjøre.



Figur V10: Funksjonell og teknisk produktkø

I sprintplanleggingsmøtet jobber man med å dele de funksjonelle produktelementene inn i delaktiviteter. Det er vanlig å bruke gruppeestimeringsprossesser som planning poker i sprintplanleggingsmøtet. Er det store avvik mellom estimatene teamet kommer fram til når man legger sammen alle delaktivitetsestimatene til et produktelement og det opprinnelige estimatet på produktelementet, må man gå tilbake til produkteier. Er oppgaven tolket på en annen måte enn produkteier mener, eller er det rett og slett flere ting som skal realiseres enn først antatt? Uansett må antall produktelementer som inngår i sprintkøen justeres slik at teamets sprintkapasitet overholdes.

I planleggingsarbeidet må man se på den tekniske produktkøen og sjekke ut om noen av de funksjonelle produktelementene krever tekniske produktelementer realisert. I figur V10 ser vi

hvordan enkelte funksjonelle elementer har kobling til ett eller flere tekniske elementer. Det første funksjonelle elementet med avhengighet til et teknisk element får en ekstrakostnad i form av at man må realisere dette tekniske elementet for å kunne gjøre det funksjonelle. produkteier må redusere sine funksjonelle ambisjoner for kommende sprint når slike situasjoner oppstår.

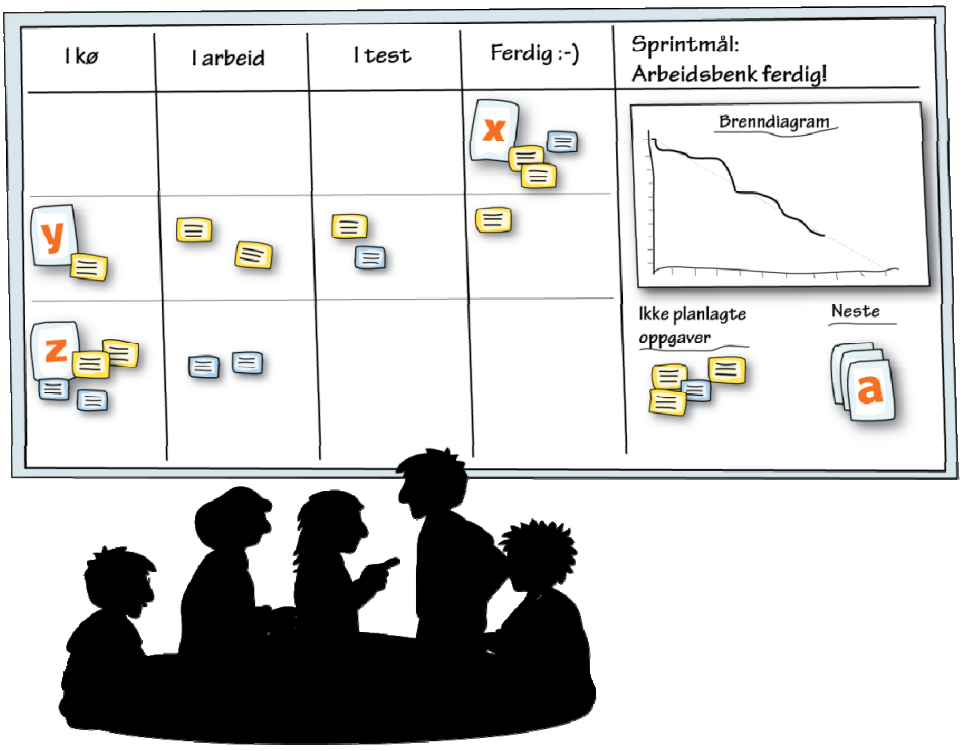
II Daglig scrum

Figur V11: Daglig scrum

Teamet møtes hver dag til samme tidspunkt og holder et 15 minutters møte. Alle deltakere svarer på følgende spørsmål etter tur

1) Hva har du gjort siden forrige møte?
2) Hva har du tenkt å gjøre til neste møte?
3) Er det noe som hindrer deg?

Samtidig oppdateres tavlen med sprintkøelementer og aktiviteter. Hver enkelt flytter lappene de har gjort noe med siden sist. Når alle er ferdige oppdaterer scrumleder brenndiagrammet.

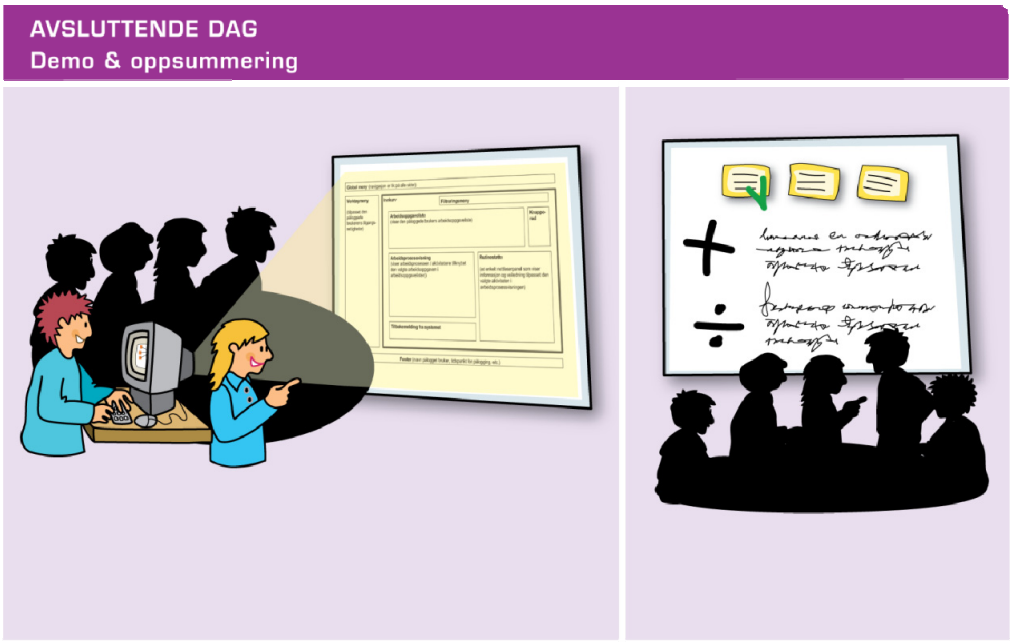


Figur V12: Brenndiagram

Oppgaver som kommer til underveis i sprinten har en egen plass på tavla. Der putter man også de køelementene man starter på hvis man får bedre tid enn planlagt.

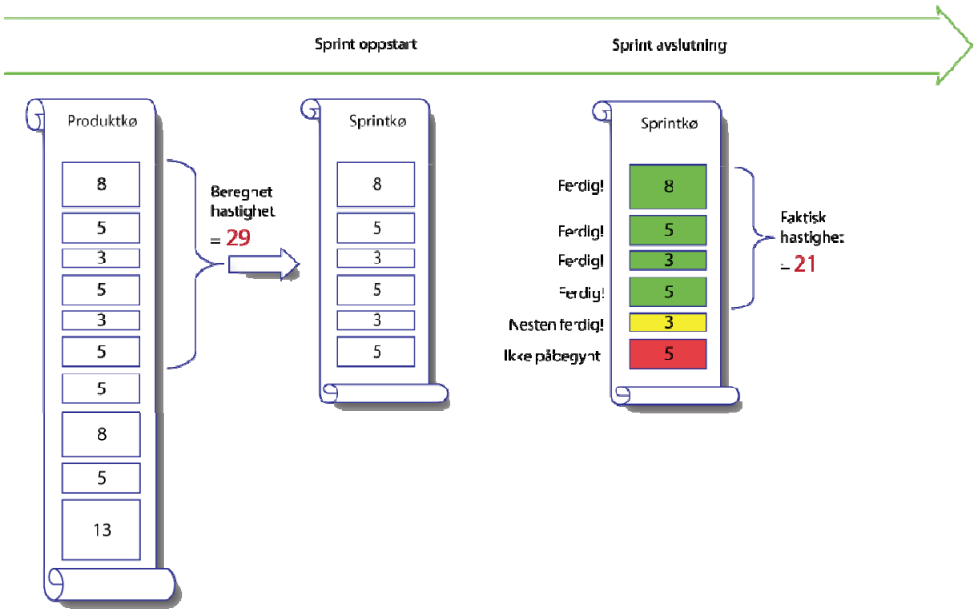
Det er ikke nødvendig med en egen kolonne for ’I test’ som vist i figur 12. En oppgave er ikke ferdig før den er enhetstestet og systemtestet, slik at den er fortsatt "I arbeid"/"Under arbeid". Hva som defineres som ’ferdig’ ble drøftet i kapittelet 5.7. Anbefalt definisjon er: ”Sprinttestede produktelementer levert til kundens verifikasjon”

III Sprintdemo



Figur V13: Sprintdemo

Siste dag i sprinten avholdes det demo for produkteier og alle andre interessenter. Hvert team demonstrerer den funksjonalitet de har utviklet i løpet av sprinten. Bare de produktelementene som er helt ferdige blir demonstrert. Er det oppgaver man ikke er ferdig med skal de tilbake i produktkøen.



Figur V14: Beregnet versus faktisk sprinthastighet

I figur 14 ser vi forløpet til et sprintteam. Siden dette var første sprint stilte man uten erfaring og måtte beregne sprinthastigheten til teamet. Hvis man tar utgangspunkt i arbeidsdager som enhet, vil beregnet sprinthastighet regnes ut fra formelen

(Σ antall arbeidsdager tilgjengelig) x (Fokusfaktor) = (Beregnet hastighet)

Beregnet hastighet var altså 29, gitt forventning om en fokusfaktor på for eksempel 70%. Resultatet var bare 21. Til neste sprint bør man da bruke justert fokusfaktor for å ha en mer realistisk plan for kommende sprint. Etter noen sprinter bør teamhastigheten øke noe for så å stabilisere seg etter hvert som teamet blir bedre kjent med hverandre og oppgavene som skal løses.

Et alternativ er å benytte fart som eneste målestokk, for eksempel i form av funksjonelle punkter eller poeng produsert pr. sprint. Se for øvrig kapittel 5.7.

IV Sprinttilbakeblikk

En viktig del av metodikken i scrum er at man skal lære underveis og forsøke å forbedre prosessen. Derfor avholdes det et retrospektivmøte i etterkant av demo, den siste dagen i sprinten. Møtet har vi kalt sprinttilbakeblikk. Møtet ledes av scrumleder og man bruker gjerne lapper og tavle i prosessen, se figur 13.

Hver enkelt skriver ned ett par lapper med ting som har fungert bra i sprinten, ett par ting som ikke har fungert bra og så forslag til konkrete forbedringspunkter for neste sprint. Alle lappene henges opp på veggen før man stemmer over hvilke forbedringsforslag man mener er viktigst. Teamet bør ta mål av seg til å gjennomføre de viktigste forbedringsforslagene i neste sprint.

7) Artefakter i scrum

I Produktkø

En produkteier stiller sammen alle krav til produktet og prioriterer blant mulig funksjonalitet. Resultatet av en produkteiers arbeid blir en produktkø – en gjøreliste over ønsket funksjonalitet som hele tiden får nye prioriteringer.

II Sprintkø

Innenfor hver sprint overføres de høyest prioriterte funksjonene til en sprintkø. Prosjektmedlemmene setter sammen scrumteam på 7, pluss/minus 2 personer. I diskusjon med prosjekteier blir man enige om målene for sprinten og bryter ned den prioriterte funksjonaliteten i detaljerte arbeidsoppgaver. Teamet er selvorganiserende og medlemmene deler ansvaret for resultatet av sprinten.

III Brenndiagram

Brenndiagrammet viser framdrift fra dag til dag innenfor sprinten. Det er vanlig å oppdatere brenndiagrammet i etterkant av daglig scrum og reglen er at bare oppgaver og aktiviteter som er ferdigstilt regnes med i ferdigstillesgrad. Målet er at man skal ”brenne” seg ned til null i løpet av sprinten og det er teamets ansvar å vurdere framdrift og treffe tiltak hvis det er nødvendig ved å ta inn eller ta ut produktelementer fra sprinten.

Vi gjør oppmerksom på at noen prosjekter benytter 'Burn-up-diagrammer' etter anbefaling fra blant annet Alistair Cockburn. Vi har imidlertid landet på Burn-down diagrammer i denne veilederen, fordi planlagt sprintvolum kan forflytte seg de første dagene i sprinten for eksempel fordi oppgaver kommer til eller produkter blir ytterligere ekspandert. Dette vil ikke kreve noe ekstra i oppdateringen av diagrammet. Samtidig er fokus på det som står igjen, fremfor fokus på det som er produsert hittil. Vi mener at dette samsvarer bedre med kjernen i fremdriftsrapporteringen.

8) Litteraturanbefalinger

Henry Kniberg: Scrum and XP from the trenches
http://www.infoq.com/minibooks/scrum-xp-from-the-trenches

Dette er kanskje den beste boka om Scrum på markedet (og den kan enda lastes ned gratis). I tillegg er den produsert av våre svenske naboer. I vår tillemping av Scrum i denne veilederen har vi basert oss tungt på denne boka.

Det smidige manifest
http://agilemanifesto.org/

Det smidige manifestet, og de 12 smidige prinsippene, slik de ble forfattet av 17 metodeeksperter i Utah, februar 2001.

Ken Schwaber: Agile Software Development with Scrum
http://www.amazon.co.uk/Agile-Software-Development-SCRUM/dp/0130676349/ref=sr_1_2?ie=UTF8&s=books&qid=1213628168&sr=8-2

Den originale Scrumboken. Overordnet om prinsippene og med elendige illustrasjoner, men et 'must-read' som innføring i Scrum.

Mike Cohn: Agile estimating and planning
http://www.amazon.co.uk/Agile-Estimating-Planning-Robert-Martin/dp/0131479415/ref=sr_1_7?ie=UTF8&s=books&qid=1213628082&sr=8-7

Den beste boka hittil på smidig estimering og leveranseplanlegging.

Kent Beck: Extreme Programming Explained: Embrace Change
http://www.amazon.co.uk/Extreme-Programming-Explained-Embrace-Change/dp/0321278658/ref=sr_1_1?ie=UTF8&s=books&qid=1213628335&sr=1-1

Den originale boken til Kent Beck i revidert utgave. Overlapper med Scrum, men med fokus på mer engineering praksiser.

James Shore og Shane Warden: The Art of Agile Development
http://www.amazon.co.uk/Art-Agile-Development-James-Shore/dp/0596527675/ref=sr_1_5?ie=UTF8&s=books&qid=1213628272&sr=8-5

En ny bok med mange praktiske erfaringer.

Alistair Cockburn: Agile Software Development
http://www.amazon.co.uk/Agile-Software-Development-Through-People/dp/0201699699/ref=sr_1_3?ie=UTF8&s=books&qid=1213628616&sr=1-3

Alistair Cockburn med fokus på de myke sidene.

Mary Poppendieck: Lean Software Development: An Agile Toolkit
http://www.amazon.co.uk/Lean-Software-Development-Agile-Toolkit/dp/0321150783/ref=pd_sim_b?ie=UTF8&qid=1213628616&sr=1-3

Mary Poppendieck's bok om Lean programutvikling.

Et par referanser til Planning Poker
http://www.planningpoker.com/detail.html

K. J. Moløkken-Østvold, N. C. Haugen, and H. C. Benestad. Using Planning Poker for Combining Expert Estimates in Software Projects, Accepted for publication in Journal of Systems and Software, 2008.

Craig Larman: Agile and iterative development
http://www.amazon.co.uk/Agile-iterative-development-Craig-Larman/dp/0131111558/ref=sr_1_12?ie=UTF8&s=books&qid=1213628233&sr=8-12

Craig Larmans bok som sammenligner ulike smidige metoder, men hvor kapitlene om hvorfor vannfall feiler er de mest nyttige.

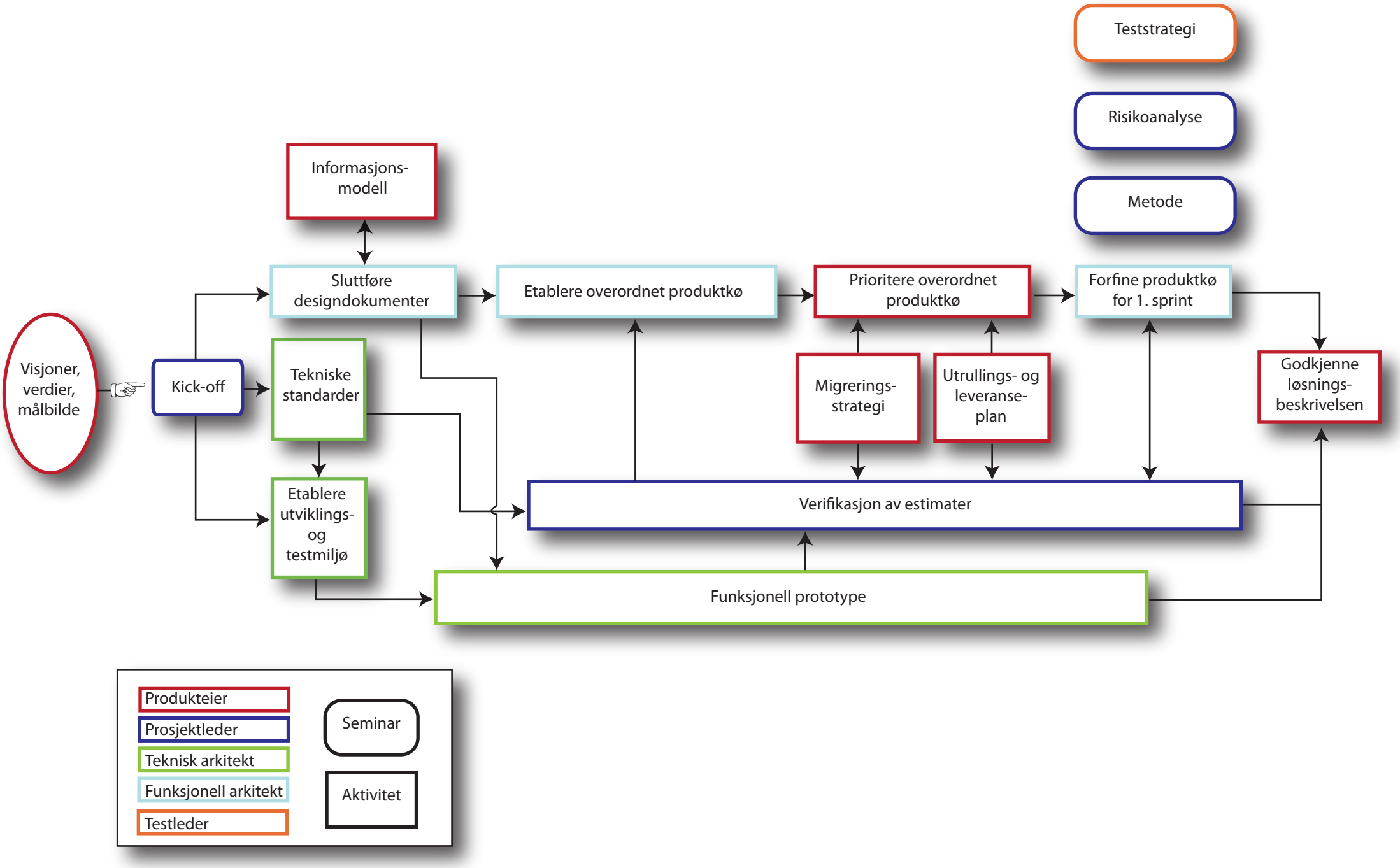
Jim Highsmith: Agile Project Management: Creating Innovative Products
http://www.amazon.co.uk/Agile-iterative-development-Craig-Larman/dp/0131111558/ref=sr_1_12?ie=UTF8&s=books&qid=1213628233&sr=8-12

Walker Royce: Software Project Management: A Unified Framework
http://www.amazon.com/Software-Project-Management-Unified-Framework/dp/0201309580/

En bok av opphavsmannen til 'fossefallsmetoden', som viser at han ufortjent har fått dette stempelet.

Barry Boehm og Richard Turner: Balancing Agility and Discipline: A Guide for the Perplexed
http://www.amazon.com/Balancing-Agility-Discipline-Guide-Perplexed/dp/0321186125/

En av forfatterne er mannen bak usikkerhetstrakten, som vi omtaler i avsnitt 8.3 ovenfor.


Teststrategi
Risikoanalyse
Metode
Informasjons-modell
Visjoner, verdier, målbilde
Kick-off
Sluttføre designdokumenter
Tekniske standarder
Etablere utviklings- og testmiljø
Etablere overordnet produktkø
Prioritere overordnet produktkø
Migrerings-strategi
Utrullings- og leveranse-plan
Forfine produktkø for 1. sprint
Godkjenne løsnings-beskrivelsen
Verifikasjon av estimater
Funksjonell prototype
Produkteier
Prosjektleder
Teknisk arkitekt
Funksjonell arkitekt
Testleder
Seminar
Aktivitet