stedet for kand.nr.)

FORSIDE

ved besvarelse av hjemmeeksamen, semesteroppgave, rapport, essay osv.

Emnekode:	**IS-402**
Emnenavn:	**Systemutviklingsprosesser og metoder I**
Emneansvarlig (normalt faglærer):	Hans Olav Omland og Even Åby Larsen
Eventuell veileder:	
Innleveringsfrist/ tidspunkt:	Fredag 23.11 klokken 12:00
Antall sider inkl. forside	37
Merknader:	

Jeg/vi bekrefter at jeg/vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	Ja [x]	Nei [ ]

Kopiering av andres tekster eller annen bruk av andres arbeider uten kildehenvisning, kan bli betraktet som fusk.

Gjelder kun gruppeeksamen:		
Vi bekrefter at alle i gruppa har bidratt til besvarelsen.	Ja [x]	Nei [ ]

Kan besvarelsen brukes til undervisningsformål?	Ja [x]	Nei [ ]

UNIVERSITETET I AGDER

IS-402
Systemutviklingsprosesser og metoder I

November 2007

Innholdsfortegnelse

Figurliste

Tabelliste

1 Innledning

I denne rapporten vurderer vi 3 ulike metoder som kanskje kan brukes av FIDO som en metode å jobbe ut i fra. FIDO bruker i dag ingen metode i sitt utviklingsarbeid, og sliter med et prosjekt som har blitt større og mer komplekst enn de først antok.
Etter at vi har valgt en metode som kanskje kan hjelpe FIDO i sitt utviklingsarbeid, vil vi beskrive hvordan FIDO bør innføre den valgte metoden inn i organisasjonen sin.

Rapporten starter med et sammendrag av hva vi har gjort, før vi fortsetter med å beskrive ETC, FIDO og hva som gikk galt. Deretter kommer det en beskrivelse og vurdering av metodene vi har valgt å se på. Ut i fra dette kommer vi med en konklusjon på hvilke metode vi mener kan passe best for FIDO og en plan for hvordan de bør innføre metoden. Til slutt har vi med refleksjoner på hvilke erfaringer vi har gjort ved å jobbe med oppgaven.

2 Sammendrag

Oppgaven er et case presentert for oss av foreleserne i faget IS-402, Systemutviklingsprosesser og metoder I. Meningen er at vi skal finne en passende metode som kunne ha løst problemstillingen i caset, beskrive metoden og hvordan vi ville innført den.

Vi valgte til slutt å bruke Scrum som utgangspunkt for metoden, men vi har gjort noen tilpasninger spesielt for det ene prosjektet som ligger i caset, nemlig utvikling av et faktureringssystem for bedriften ETC.
FIDO, som er utviklingsbedriften, må nok gjøre andre tilpasninger når de får nye prosjekter, men grunnlaget som nå er lagt har også tatt høyde for at man kan gjøre slike tilpasninger.

Vi har valgt å ikke løse selve utviklingen av systemet, dvs vi har ikke fordelt aktiviteter som er direkte knyttet til analyse, design og programmering av de aktivitetene vi kunne tenke oss dannet en foreløpig kravspesifikasjon for systemet. Vi har heller valgt å legge vekt på de aktiviteter, roller og tilpasninger som gjelder for metoden når slike system skal utvikles, og også prøvd å ta hensyn til de mellommenneskelige forholdene i dette tilfelle.

3 Analyse av ETC's organisasjon og informasjonssystem

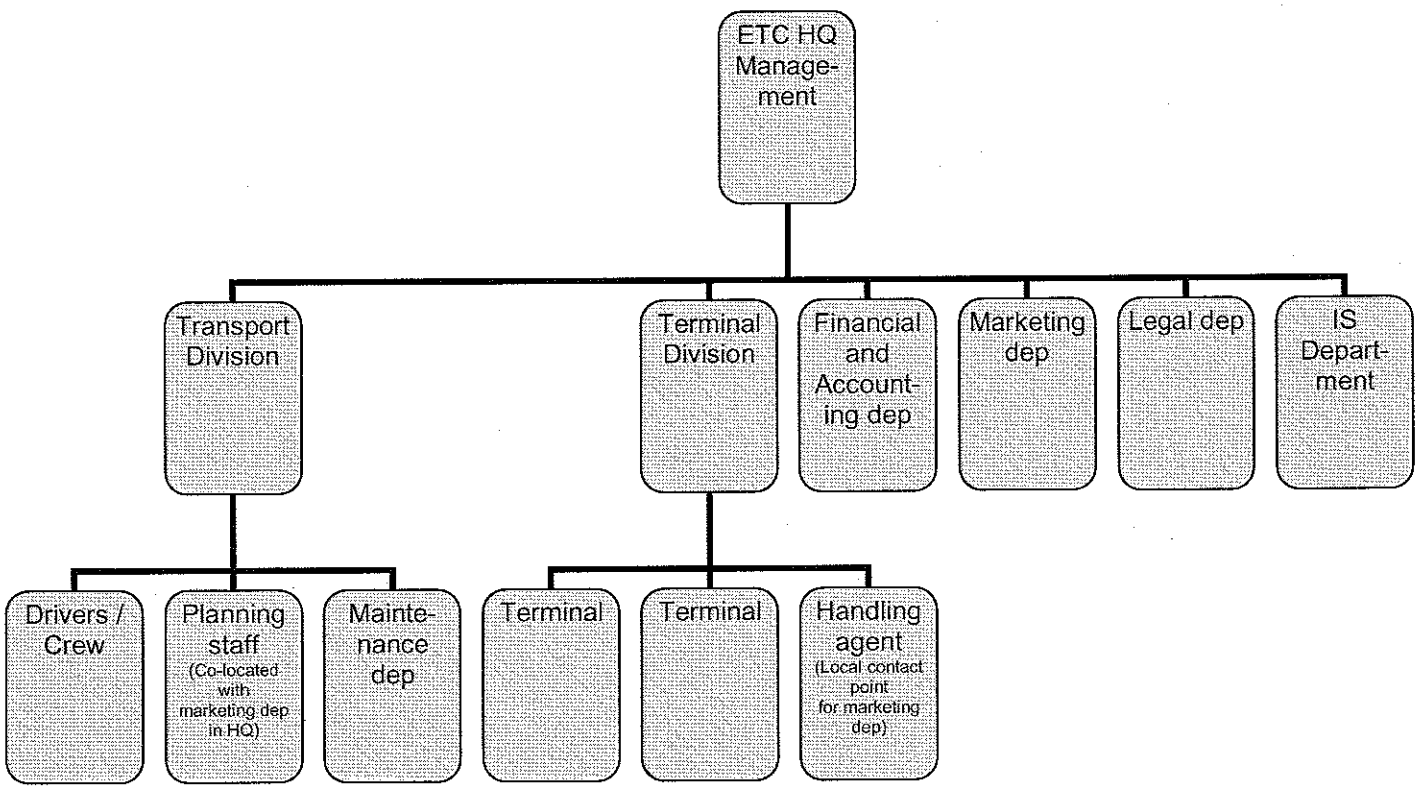
3.1 Organisasjonsbeskrivelser

ETC er et transportfirma som frakter forskjellig typer gods, med forskjellig typer transportmidler som tog, bil, fly etc.
Transporten skjer både nasjonalt og internasjonalt, som for eksempel Oslo, Gøteborg og Peking.
Hovedkontoret ligger i en liten by sør i Norge.
Kundene er hovedsaklig bedrifter som har behov for hjelp til å distribuere produktene sine ut til markedet, men ETC selger også restkapasitet på ad hoc basis til andre bedrifter og enkeltindivider.

ETC oppdaget etter hvert at fakturasystemet de brukte hadde flere mangler; det var arbeidskrevende, det var lett å gjøre feil pga mye manuell input og det var vanskelig/umulig for regnskapsfolkene å være sikre på at man fikk med seg alle aktuelle poster på hver faktura. De kom etter hvert til at et nytt automatisert fakturasystem ville løse problemene.

ETC satte i gang jobben med å utarbeide en kravspesifikasjon, og under arbeidet med denne innså de at de også ville få behov for et kontraktsystem hvis fakturasystemet skulle fungere så automatisk som de ville. Kontraktsystemet vil ellers bli et helt separat system som ETC har noen tilleggskrav til som *kanskje* skal implementeres, men da altså uavhengig av fakturasystemet.

ETC består hovedsaklig av 3 "avdelinger", nemlig hovedkontoret (HQ), transportavdelingen og terminalavdelingen. Der det ikke er regningssvarende å ha egne terminaler, har ETC "handling agents", agenter som i tillegg til å virke som mini-terminaler også fungerer som lokale kontaktpunkter for markedsavdelingen.
Organisasjonsstrukturen i ETC ser tilnærmet slik ut:



Figur 1: Organisajsonskart over ETC

3.2 Problemstillinger

Siden IS-avdelingen i ETC ikke har noen ledig utviklingskapasitet, bestemmer ETC seg for å be konsulentbyrået FIDO om hjelp til utviklingen, FIDO har tidligere utviklet ETC's transportplanleggingssystem og stått for vedlikehold av noen av komponentene i logistikksystemet. I tillegg har FIDO jobbet med en objektmodell for hele bedriften for å lette integrasjonen av de ulike systemene. De kjenner derfor ETC og systemene der rimelig godt.

Den eneste delen av ETC som de ikke har hatt noe med å gjøre, bortsett fra under arbeidet med å lage objektmodellen, er terminalavdelingen. Terminalavdelingen har brukt et annet firma enn FIDO til å hjelpe seg med sine systemer og terminalavdelingen er derfor i tvil om FIDO er i stand til å se behovene deres.

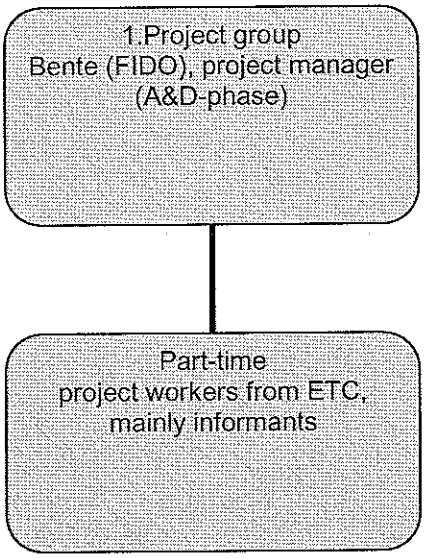
ETC og FIDO blir etter forhandlinger enige om at FIDO skal levere en komplett kravspesifikasjon og design av det nye systemet, betalt time-for-time.
Selve utviklingen av systemet skulle være til en fastpris og ETC stod i utgangspunktet fritt til å velge en annen leverandør enn FIDO til dette. Nå viste det seg etterhvert at FIDO fikk denne delen av prosjektet også, mye takket være Stig (FIDO ansatt) som hadde vært med på å utvikle ETC's logistikksystem før han ble ansatt i FIDO.

Første fase av prosjektet, analyse og design fasen (A&D) ble estimert til å ta ca 3 mnd, en tidsramme som viste seg å bli ettertrykkelig sprengt ettersom fasen tok mer enn 1 år.
Noe av grunnen var som følger:

- Faktureringsprosessen var mer kompleks enn først antatt
- Man avdekket behovet for et kontrakt system
- Siden faktureringssystemet var mer komplekst enn først antatt, så økte kompleksitetsnivået i kontraktsystemet også
- Terminal-divisjonen var i mot prosjektet. (De var egentlig ikke i mot selve prosjektet, de var bare redde for at deres behov ikke ville bli hørt eller forstått. De ville derfor ha en annen utvikler enn FIDO, siden de hadde brukt en annen før, og regnet med at denne kjente behovene deres bedre.)

Arbeide med kravspesifikasjonen var veldig vanskelig, slik at analyse- og designfasen strakk seg langt inn i programmeringsfasen.
Programmererne prøvde å håndtere dette ved å konsentrere seg om de delene av systemet de oppfattet som mest stabile, dvs de delene de trodde det ikke ville komme endringer på.
La oss se på hvordan selve prosjektorganisasjonen bygget seg opp over tid:

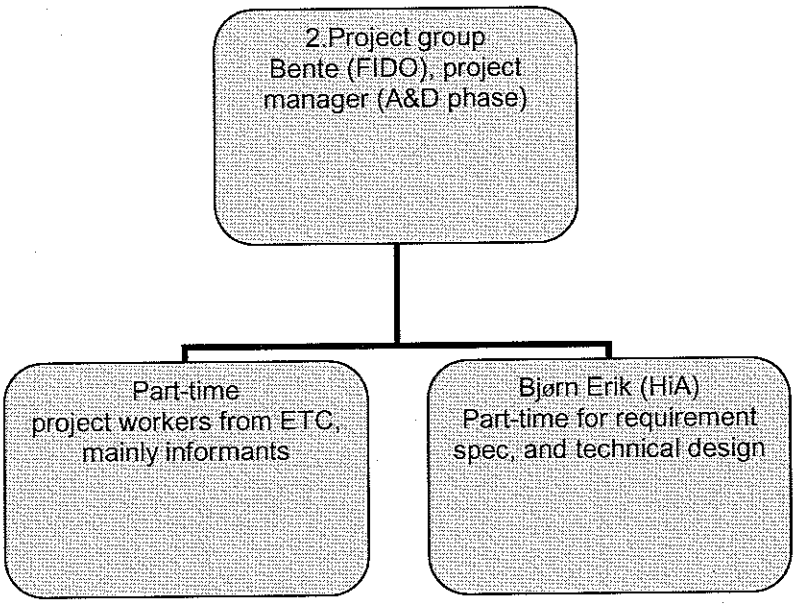
I første del av prosjektet, bestod prosjektet bare av Bente, prosjektleder fra FIDO, og deltids prosjektarbeidere fra ETC.



Figur 2: Organisasjonskart over prosjektgruppe 1

Man skjønte etter hvert at man hadde behov for flere folk, og leide inn en lærer fra HiA på deltid i prosjektet i Analyse- og Designfasen.

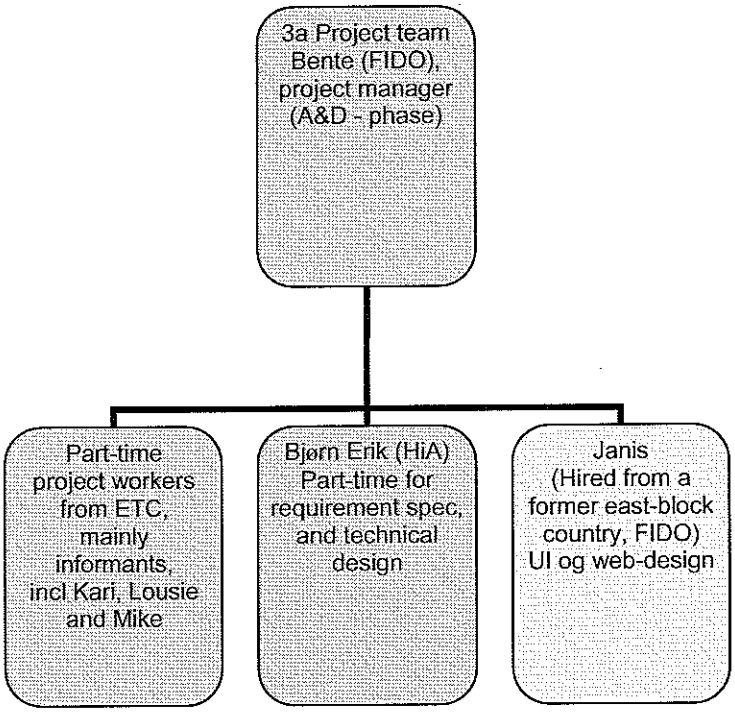
Organisasjonskartet til prosjektet ser nå sånn ut:



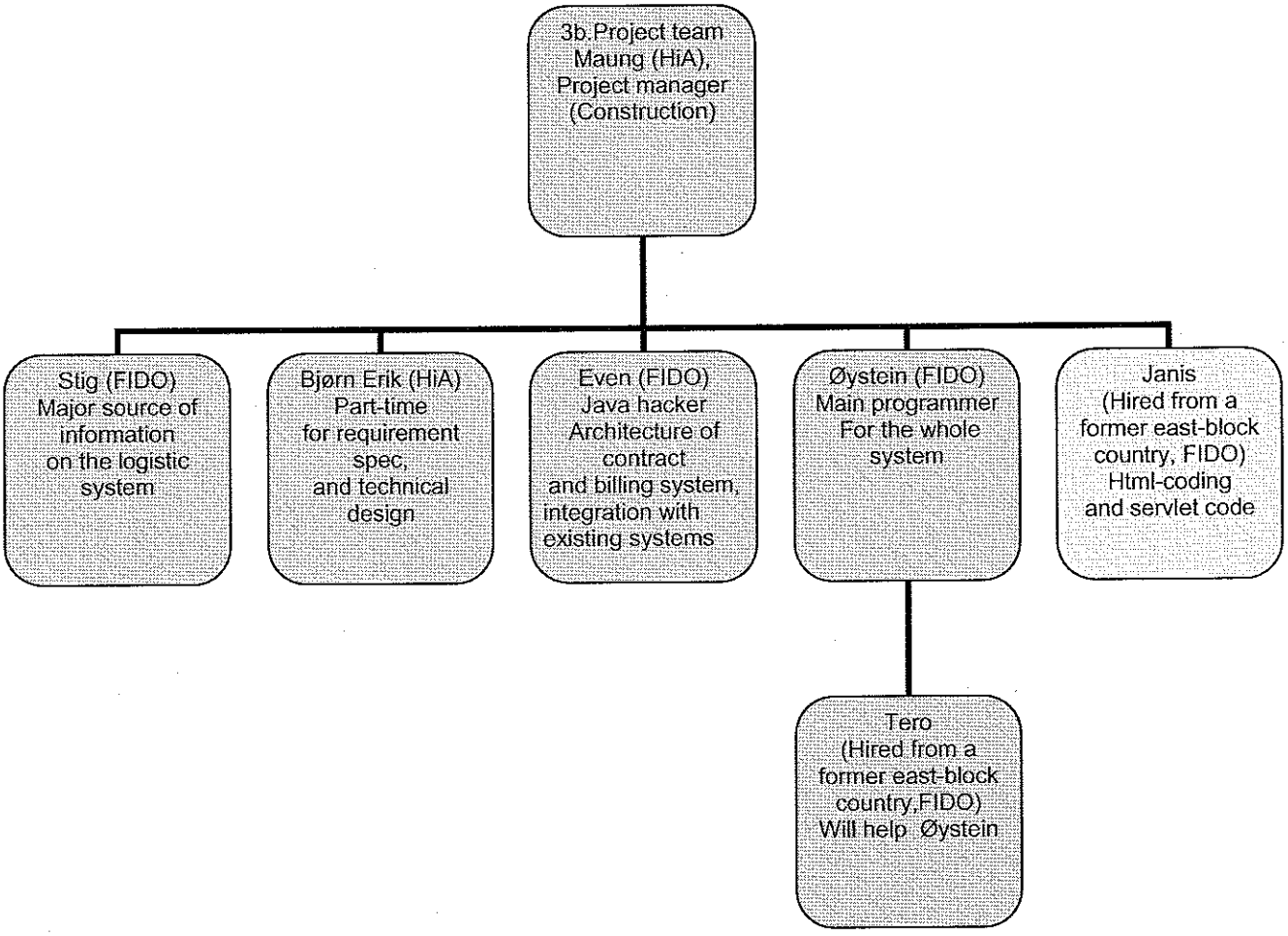
Figur 3: Organisasjonskart over prosjektgruppe 2

Da de så kom til konstruksjonsfasen, skjønte FIDO at de ville få behov for enda flere folk.

Vi får dermed 2 sidestilte prosjektorganisasjoner, med til dels overlappende medlemmer, og 2 prosjektledere:



Figur 4: Organisasjonskart over prosjektgruppe 3a



Figur 5: Organisasjonskart over prosjektgruppe 3b

3.3 Hva gikk galt?

Pga at man måtte utvikle et kontraktsystem i tillegg til fakturasystemet, fikk man flere stakeholders.
Siden A&D-fasen strakk seg langt inn i konstruksjonsfasen, fikk programmererne trøbbel med hva man kan kalle en noe flytende kravspesifikasjon, dvs. en spesifikasjon som levde og endret seg lenge etter at programmeringen var begynt. Resultatet var at programmererne valgte å konsentrere seg om de delene av kravspesifikasjonen som virket mest stabile, dvs. de delene som man minst forventet endringer i, og programmerte disse delene først. Det kan hende at dette tilfeldigvis ble den beste rekkefølgen å programmere i, selv om ingenting hadde forandret seg underveis, men det er nokså lite sannsynlig, og vanskeligjorde sannsynligvis programmeringen mer enn nødvendig.
En annen liten ting her: Programmererne "antok" at disse delene ikke ville forandre seg, noe som gir oss et hint om at kommunikasjonen mellom A&D-teamet og konstruksjonsteamet kanskje ikke var den beste. Men det kan også bety at de hadde grei kommunikasjon, men at A&D-teamet var såpass usikre ang kravspesifikasjonen at de ikke klarte å gi konstruksjonsteamet gode svar. Eller begge deler.

Man hadde time-for-time pris i A&D-fasen, noe som gjorde at Calle, leder og hovedeier av FIDO og bekymret for at FIDO skulle tape penger på prosjektet, *kunne* ha gode grunner for å strekke denne fasen med seg inn i programmeringsfasen som hadde en fast pris. På den måten kunne Calle "ta igjen" det tapte i fastpris-delen, gjennom å kjøre time-for-time-delen parallelt.

Mike liker ikke Ivan, vi vet ikke hvorfor, men vi vet at det er merkbart.
Terminalavdelingen motarbeider prosjektet, og det kan være pga Mikes holdning til Ivan, men også/eller fordi de er redde for at FIDO ikke vil se behovene deres, så de vil heller ha sitt "gamle" konsulentfirma på banen.

4 Metodebeskrivelser

4.1 STRADIS

STRADIS er en strukturert metode som har tatt i bruk flere forskjellige teknikker. Teknikkene finns i andre metoder, og mange oppfatter metoden som et sammendrag av de metoder som er basert på å skulle bryte ned problemstillinger i mindre deler. I tillegg gjør den seg bruk av dataflytdiagrammer.

Her følger en kort beskrivelse av STRADIS, hvor vi vektlegger å peke på de punkter som vil lette/vanskeliggjøre/mangle i forhold til FIDOs utvikling av ETC sine systemer:

Innledningsfasen

Innledningsvis ønsker man først og fremst å forsikre seg om at det systemet man velger å utvikle er det systemet som mest sannsynlig vil kunne gi bedriften en ekstra fordel i et konkurranseutsatt marked. Det viktigste kriteriet blir da kostnader og gevinster ved et nytt system, evt. ved hvert av alternativene.
Man starter med å samle inn informasjon fra ledere og brukere i bedriften. Videre går man gjennom eksisterende dokumentasjon og belyser alternativene i forhold til strategiske planer som måtte finnes i bedriften. Man lager et dataflytdiagram som viser en oversikt over det eksisterende systemet og dets brukergrensesnitt., og en oversikt over den tid og kostnad man vil bruke på å gjøre en mer detaljert undersøkelse. I tillegg kan man gjøre noen grovere overslag for det endelige resultatet. Denne fasen er beregnet til å ta mellom to dager og fire uker, avhengig av størrelsen og viktigheten av systemet.
Avslutningsvis ender denne fasen i en rapport som forelegges ledelsen, som så kan ta avgjørelsen om hvorvidt de vil gå videre til en mer detaljert studiefase, eller om de vil avslutte der og da. Velger de å gå videre, aksepterer de samtidig kostnadene for den detaljerte fasen, men ikke nødvendigvis og fortsette med utviklingen etter det.

Fordeler og ulemper i forhold til FIDO:

Ved å bruke innledningsfasen til STRADIS vil man kunne kartlegge ETC's eksisterende system, og også vite litt mer om den tid og kostnad man vil bruke på en noe mer detaljert undersøkelse.
Den gir ledelsen i ETC en mulighet til å få en fast pris på detaljkartleggingsfasen, og det passer bra siden ETC's ledelse helst vil bruke fast-pris-prinsippet. På den annen side vet vi at detaljkartleggingsfasen i dette tilfelle ikke ville avdekket alle behovene ETC har, fordi behovene ikke kom fram før på et mye senere tidspunkt, derfor vil ikke STRADIS metoden kunne gi de svarene FIDO trenger for å kunne fullføre fasen. Det betyr at FIDO må gjøre mye mer arbeid enn de først hadde trodd hvis de virkelig skal komme til bunns av problemstillingen i denne fasen, og de vil dermed tape penger på det. Og mest sannsynlig vil de allikevel ikke kunne gi ETC de svarene som trengs.
Konklusjonen til denne fasen blir allikevel at den kan brukes til en første kartlegging, men at å gi en fast pris på detaljkartleggingsfasen vil være vanskelig, særlig fordi vi vet at en detaljkartleggingsfase kanskje ikke engang er en fase som vil la seg gjennomføre i dette prosjektet.

Detaljkartleggingsfase

Etter denne fasen konsentrerer STRADIS-metoden seg om det eksisterende systemet.
Man prøver her å kartlegge hvem som er brukere av systemet, kostnader ved gammelt og nytt system og fordeler ved et nytt system.
Det lages et dataflytdiagram for hver viktige prosess og til slutt et overslag over tids- og kostnadsbruk for neste fase.

STRADIS kartlegger brukerne på 3 nivåer:

- De som bestilte systemet, dvs. de lederne som innså at behovet var tilstede og deretter foreslo å bestille et nytt.
- Mellomledere i de berørte avdelingene
- Sluttbrukerne, dvs. de som direkte vil bruke systemet.

Kort summert skal denne fasen resultere i:

- En detaljert definisjon av de forskjellige brukerne av systemet, funksjonene, sammenhengene, hvilke arbeidsoppgaver som vil bli berørt, jobbeskrivelser osv
- En logisk modell av eksisterende system
- En oversikt over de fordelene som kan gis av et nytt system, både i form av effektivitet og besparelser, men også en oversikt over evt. økninger av utgifter i form av vedlikehold, større lagringsbehov osv
- Oversikt over lovbestemmelser man muligens må ta hensyn til, konkurransedyktigheten til systemet, systemkostnader og et budsjettforslag for neste fase, gjerne med alternativer.

Fordeler og ulemper i forhold til FIDO:

Kartlegging av brukerne, og oversikt over kostnadene ved det gamle systemet er oppgaver FIDO vil kunne nyttiggjøre seg fra denne fasen. De vil også kunne lage dataflytdiagrammer over de prosessene som de til nå har avdekket, men vi vet fra oppgaveteksten at alle behov ikke *kan* avdekkes her.
Det blir derfor også vanskelig å gi et overslag over tids- og kostnadsbruk for *hele* neste fase. Spørsmålet er om de i det hele tatt kan kjøre neste fase som én fase.

Definering og design av de alternative løsningene

Her følger en fase hvor man definerer alternative løsninger på problemene til det eksisterende systemet.

Først tar man for seg de organisatoriske målene som ble kartlagt i innledende fase og bearbeider disse målene fram til systemmål. Et systemmål vil si å gi en beskrivelse av hva et nytt system burde kunne gjøre for å hjelpe ledelsen å nå de organisatoriske målene. Videre må systemmålene bør være spesifikke og målbare, ikke generelle.

Analytikere vil deretter bruke disse målene til å produsere logiske dataflytdiagrammer for det nye systemet. Også dataflyten mellom det som vil gjenstå som manuelle deler av systemet, slik som å ta i mot bestillinger via telefon/e-post, kundeoppfølging før kontraktsinngåelse,

hvordan tar man vare på dokumenter/e-post/samtaler som fører fram til en kontrakt, hvem legger inn kontrakten og dermed starter opp den nye automatiserte delen av systemet, osv.

Slike dataflytdiagrammer bør ende opp såpass detaljert at man ser at systemkravene er imøtekommet.

Analytikere og designere skal her jobbe sammen om å komme opp med 3 forskjellige designkategorier: (Avison, Fitzgerald (1995))

1. Lavbudsjett, rask implementering, møter ikke nødvendigvis alle krav
2. Middels stort budsjett, noe senere implementering, oppfyller de fleste kravene
3. Høyt budsjett, versjon som oppfyller alle krav.

Her gir man et røft estimat for kostnad, fordeler/gevinster, tidsforbruk, maskinvare, programvare.
Utfallet av denne fasen skal være en rapport som inneholder de tre forslagene over som skal presenteres for ledelsen, og ledelsen må bestemme seg for ett av alternativene. Rapporten bør inneholde følgende:

- et dataflytdiagram over eksisterende system
- begrensninger i eksisterende system
- logisk dataflytdiagram over nytt system

Til hvert av de 3 alternative løsningene bør følgende være med i rapporten:

- den delen av dataflytdiagrammet som vil bli implementert
- UI (brukergrensesnitt, terminaler, rapporter osv)
- Estimert kostnad og gevinst
- Implementeringstimeplan
- Risiko

Fordeler og ulemper i forhold til FIDO:

Idéen med 3 alternative løsninger virker forlokkende, da det gir ledelsen muligheten til å senke kravene til funksjonalitet mot at produktet leveres i rett tid, evt. at man utvider tidshorisonten og krever full funksjonalitet, eller at man oppfyller alle krav innenfor tidsfristen men dermed må øke ressursene og få et dyrere system.
Problemet i dette tilfelle er at FIDO ikke vil kunne gi ETC en full oversikt over *hele* systemet, men idéen burde kunne brukes på de delene av systemet man har oversikt over. Det betyr at STRADIS igjen gir oss noen detaljer vi kan bruke, men kun det, vi kan ikke følge metoden helt og holdent her heller.

Fysisk design

Det valgte alternativet blir så utviklet til et spesifikt fysisk design av designerne.
Denne fasen består av et antall aktiviteter, bl.a.:

- Detaljerte dataflytdiagrammer, med beskrivelse av feilhåndtering og unntak av alle slag, og all prosesslogikk, rapport og skjermbildeformater. Rapporter og skjermbildeformater bør valideres av brukerne..

- Fysiske filer el. databaser blir designet. De baseres på dataflytdiagrammene, som viser hvilke data man har behov for å få lagret i den gitte prosessen som beskrives. Mange av dataflytdiagrammene vil etter hvert overlappe hverandre med hensyn til data som har behov for lagring.
- Datalagrene rasjonaliseres, teknikker for normalisering blir brukt for å slå sammen og forenkle datalagrene i logiske grupperinger.
- Modellere et hierarki av funksjonene, på bakgrunn av innholdet i dataflytdiagrammene. Her prøver man å finne ett av to mulige strukturer for dataprosesseringen: "transform-centered", hvor alle transaksjonene følger veldig like prosesseringsveier(stier), eller "transaction-centered", hvor transaksjonene krever forskjellig prosessering.

Til slutt definerer man hvilke manuelle jobber det nye systemet vil kreve, dvs. hvor grensene for det automatiserte systemet går.

Alle oppgavene over utføres til et slikt detaljert nivå at man er stand til å gi et pålitelig anslag over kostnad for utvikling og betjening av det nye systemet.
De vesentligste sidene av disse kostnadene identifiseres ved:

- arbeidstimer som kreves for å utvikle de modulene man har bestemt
- Maskinvare og programvare som kreves
- Perifere enheter (skrivere og lignende) og datakommunikasjonskostnader
- Arbeidstimer til å utvikle brukerdokumentasjon og opplæring av brukere
- Tiden brukerne bruker på systemet
- Arbeidstimer som skal til for å vedlikeholde og videreutvikle/rette systemet i dets levetid

Fordeler og ulemper i forhold til FIDO:

I forhold til hovedingrediensen i denne fasen, nemlig at alle rapporter og skjermbilder osv skal designes til bunns, er umulig, fordi man enda ikke har brakt alle kravene fram i dagens lys.
Det er muligens aktuelt å bruke prinsippet, som beskrevet over, til å utvikle mindre moduler av systemet, men i den sammenheng vil det ikke være aktuelt å for eksempel designe databaser for *hver enkelt* modul eller funksjon. Flere av de andre kravene vil også være ting man gjør for hele systemet på en gang, så alt i alt vil vi konkludere med at denne fasen av STRADIS ikke lar seg bruke for FIDO's prosjekt.

Implementeringsfasen

STRADIS er en metode som tar for seg hovedsakelig analysedelen, noe mindre opptatt av designdelen, og nesten ikke har noe skrevet om implementering, men det finnes noen punkter som Gane og Sarson mener gir en pekepinn om hva som skal til for å fullføre systemet: (Avison & Fitzgerald, 2006)

- lage en implementeringsplan som inkluderer plan for testing og aksept av systemet
- utvikle applikasjonen og database/datakommunikasjons-funksjonene og samtidig
- "befolke"(populere) databasen
- Teste og forsikre seg om at man har aksept for hver del av systemet
- Forsikre seg om at systemet oppfyller de performance-kriterier som er definert i system-målene, og utsette det for realistisk datahåndteringsbelastning for så å måle responstid og "throughput" (mengden data som flyter gjennom).

- Sette systemet i gang i det virkelige liv, sjekke evt. flaskehalser og tune det.
- Sammenlikne systemet og performance-graden med det som opprinnelig var målet, og sørge for å rette alt som ikke svarer til forventingene.
- Analysere og se etter mulige forbedringsområder, prioritere disse, og så la systemet gå over i en vedlikeholdsfase.

Fordeler og ulemper i forhold til FIDO:

Etter hvert som STRADIS sammenliknes med behovet for systemutviklingsmetoder for FIDO, ser man at implementeringsfasen til STRADIS ikke så lett lar seg vurdere i den sammenheng. Dette er fordi at vi har konkludert de tidligere fasene med at man kan ta elementer fra dem, og kun det, fordi prosjektet til FIDO vil måtte brytes ned i mye mindre moduler. Dermed vil hver modul måtte ha sin egen test- og implementeringsplan, som vil skille seg fra det som er å teste hele systemet på en gang.
Implementeringsfasen i STRADIS kan derfor ikke brukes.

Forhold ved FIDO som ikke blir tatt hensyn til ved bruk av STRADIS metoden:
Det er mange forhold ved FIDO som ikke kan tas opp til vurdering hvis man bruker STRADIS. Det største problemet er at kompleksiteten ikke avdekkes med en gang, slik at det hele tiden kommer endringer underveis. STRADIS håndterer i liten grad endringer, i beste fall må man satse på å begynne kartleggingsfasen på nytt hver gang en endring oppstår.
Dette forholdet er av så stor betydning at andre forhold som ikke blir tatt hensyn til, slik som at brukerne sitter svært spredt, at man har brukere med store motsetninger internt i ETC osv ikke behøver å kommenteres punkt for punkt her, men kommer med i en endelig konklusjon, hvis nødvendig.

4.2. Feature Driven Development (FDD)

FDD ble utformet av Jeff De Luca i 1997 og først rapportert av Coad, P., Lefebvre, E. & De Luca, J. (1999). FDD er en metode som har fokus på å utvikle funksjoner som kan utvikles på under 2 uker. Metoden beskriver aktiviteter fra analyse til utvikling. FDD består av fem sekvensielle prosesser og inneholder roller, artefakter, mål og tidslinjer som trengs i et prosjekt.

Utvikle en overordnet modell

Man starter med å samle domain (domene/fagområde)- og utviklingsmedlemmene under veiledning av en erfaren objektmodellerer som innehar rollen "Chief Architect".

Det blir gjennomført en gjennomgang på toppnivå av omfanget til systemet og konteksten. Deretter blir det gjennomført en detaljert gjennomgang av hvert område, slik at det kan bli modellert. Etterpå bli små grupper bestående av fagpersoner og utviklere satt sammen. Gruppene tar så å utvikle sine egne modeller for å støtte opp om domenegjennomgangen. Gruppene presenterer så sine modeller for hver andre for gjennomgåelse og diskusjon. En av de foreslåtte modellene, eller en sammenslåing av modeller, blir så valgt av gruppen slik at modellen blir brukt på det enkelte domene/fagområdet. En sammenslåing av områdemodellen inn i en overordnet modell blir så utført.

Fordeler og ulemper i forhold til FIDO:

I analysefasen blir det nødvendig for FIDO å bruke flere av sine ansatte til å utføre selve analysen. Det positive er at man jobber side-ved-side med den som vet hva systemet skal brukes til. I denne fasen ville FIDO tidligere oppdaget behovet for et kontraktssystem. Ut i fra dette kunne man reforhandle fastprisen siden det ble avdekket et nytt behov.
Ulempen blir at FIDO trenger en person med erfaring innen objektmodellering, og er avhengig av å ha folk fra ETC hos seg under denne fasen.

Lage en funksjonalitetsliste

En gruppe av hovedprogrammerer blir dannet for å bryte ned funksjonaliteten i domenene i mindre emneområder, og hver av dem skal igjen brytes ned i forretningsaktiviteter som igjen brytes ned i enda mindre deler, slik at man utformer en kategorisert funksjonalitetsliste. Kategoriseringen av funksjonalitetslisten blir utført i samarbeid med domeneekspertene.

Fordeler og ulemper i forhold til FIDO:

Ved at FIDO bruker hovedprogrammererne til å gjøre om domenemodeller til funksjoner, får FIDO frem de ulike forretningsaktivitetene og de funksjoner som trengs. På denne måte får man noe å jobbe ut i fra.
Ulempen blir at FIDO er avhengig av å ha folk fra ETC hos seg under denne fasen.

Planlegg etter funksjonalitet

Prosjektlederen, utviklingslederen og hovedprogrammererne planlegger rekkefølgen som funksjonaliteten skal implementeres, ut i fra funksjonalitetsavhengighet, arbeidskapasiteten til utviklingsgruppen og kompleksiteten til funksjonaliteten som skal implementeres. Planen som blir utformet kan oppdateres etter behov.

Fordeler og ulemper i forhold til FIDO:

FIDO vil få frem en plan for når ting skal bli ferdig og i hvilken rekkefølge. De får også frem ansvar for forretningsaktiviteter og liste over klasseeiere.
Ulempen er at man kan bomme på rekkefølgen man skal utvikle funksjonaliteten på.

Designe etter funksjonalitet

Hovedprogrammererne velger funksjonaliteter fra sin *innboks*. Han kan velge mer enn en funksjonalitet, som bruker de samme klassene. Hovedprogrammererne danner så en funksjonalitetsgruppe ved å identifisere eiere av klasser som mest sannsynlig vil bli brukt i utviklingen av funksjonaliteten(e) han har valgt ut til utvikling. Gruppen lager så "Sequence Diagram(s)" av funksjonaliteten. Hovedprogrammereren raffinerer objektmodellen basert på innholdet i "Sequence diagram(s)". Utviklerne skriver klasse og metode innledninger. Til slutt blir det holdt en design inspeksjon.

Fordeler og ulemper i forhold til FIDO:

I dette stadiet får FIDO kartlagt akkurat hva en funksjon skal gjøre med tanke på utviklingen. FIDO får også samlet de personene som er ansvarlige for de ulike klassene slik at man kan unngå implementeringsproblemer.

Utvikle etter funksjonalitet

Ved å se på designpakken, tar utviklingseieren av klassen å implementere de nødvendige tingene for at klassen deres kan støtte designet av funksjonaliteten. Koden som så er utviklet gjennomgår en enhetstest og kodeinspeksjon. Etter at koden har gjennomgått en suksessfull inspeksjon, blir koden sendt videre til hovedsystemet.

Fordeler og ulemper i forhold til FIDO:

For FIDO vil nok dette passe godt inn med tanke på at de i dag må utvikle det de trur er ferdig analyser og designet. Ved å bruke FDD vil FIDO mest sannsynlig kunne utvikle korrekte funksjoner, samtidig som kravene kan endres. En ekstra styrke er at FIDO etter hver iterasjon kan vise til ny funksjonalitet.
Et problem kan være at FIDO bommer på hvor lang tid det tar å utvikle funksjonen, og dermed kan ende opp med å bruke mer tid på prosjektet enn anslått.

Konklusjon:

Selv om FDD klart kan hjelpe FIDO med å få en struktur på hvordan de skal utvikle systemer på. Er det visse ting som kan bli problematisk. For det første er FIDO avhengig av å ha en person med god kunnskap om objektmodellering. FIDO er også avhengig av at kunden kan stille med personer som innehar kunnskap om de ulike områdene som systemet påvirker. FDD har fokus på små grupper og ansvarsfordeling på roller. Her kan det være at FIDO mangler folk til å få en god fordeling av rollene og inndeling i grupper.
Det er ingen garanti for FIDO at man oppdager alle systemkravene ved bruk av FDD, men de kan håndtere endringer i kravspesifikasjonen som kommer etterhvert.

4.3 SCRUM

Forløperen til Scrum er Sashimi som ble utviklet av japanerne som hadde de samme problemene med Fossefallsmodellen som alle ellers, nemlig at de ikke kunne gå tilbake til en tidligere fase, derfor tilpasset de Sashimi til å passe deres egen stil. Tidsbruk og fleksibilitet var like viktig som høy kvalitet og lav pris, derfor reduserte de antallet faser til fire - krav, design, prototyp og mottakelse uten å fjerne noen aktiviteter, dette ført til overlapping av de forskjellige fossefallsfasene. Andre utviklet så Sashimi videre, slo sammen de fire fasene til én, og kalte det Scrum.

Andre mener at Scrum fremdeles kan deles inn i flere faser og allikevel opprettholde fleksibiliteten, vi har derfor valgt å presentere Scrum i 3 faser.

Scrum prosess i tre faser:

Forspillsfasen (Pre-game Phase) som inkluderer to underfaser:

- *Planlegging (Planning)* inkluderer beskrivelsen av systemet som skal utvikles, prosjektgruppe, redskaper og andre ressurser. Et produkts backlog (gjenstående arbeid) inneholder alle kjente krav. Kravene er prioritert og backlog'en er bestandig oppdatert med nye krav og med mer nøyaktig vurderinger. For å kunne gå til neste iterasjon, må backlog'en sjekkes først.

  Planleggingen inkluderer bl.a. definisjon av prosjektgruppen, verktøy, risikovurdering osv.

- *Arkitektur/Høy nivå design (Architecture/High level design)* planlegger det høye jevne designet av systemet, samt arkitekturen.

Fordeler og ulemper i forhold til FIDO

Ved å bruke forspillfasen til Scrum, vil man vite nok om systemet som skal utvikles, til å kunne gjennomføre en første iterasjon, og også vite litt om den tid og kostnad man vil bruke for det.

Utviklingsfasen (Development Phase) er laget slik at den er smidig (agil), og denne smidigheten er litt av kjernen i Scrum. Her er uforutsette hendelser forventet, og man vet hvordan man skal takle det ved hjelp av såkalte Sprinter.

Sprinter er iterative sykluser. Hver Sprint inkluderer de tradisjonelle fasene av systemutvikling: krav, analyse, design, evolusjon og leveringsfaser, men har kun en kravspesifikasjon som gjelder en liten bit av systemet (for eksempel én funksjon) som skal utvikles akkurat nå. En Sprint varer fra en uke til en måned. Det kan være et varierende antall personer som jobber på hver Sprint.

Scrums metodikk krever at hver person i gruppen forstår hele problemstillingen og alle trinnene ved utviklingen av systemet.

Under hver sprint holdes det daglige møter og man har et demonstrasjonsmøte på slutten av en sprint. Et demonstrasjonsmøte gir både kunden og utviklerne et godt bilde av hva som er utviklet og hvor langt man er kommet.

Man løser altså en oppgave i hver Sprint, og når det ikke lenger er oppgaver igjen å løse, eller man mener at det ikke vil lønne seg å løse de siste, er utviklingen ferdig.

Fordeler og ulemper i forhold med FIDO

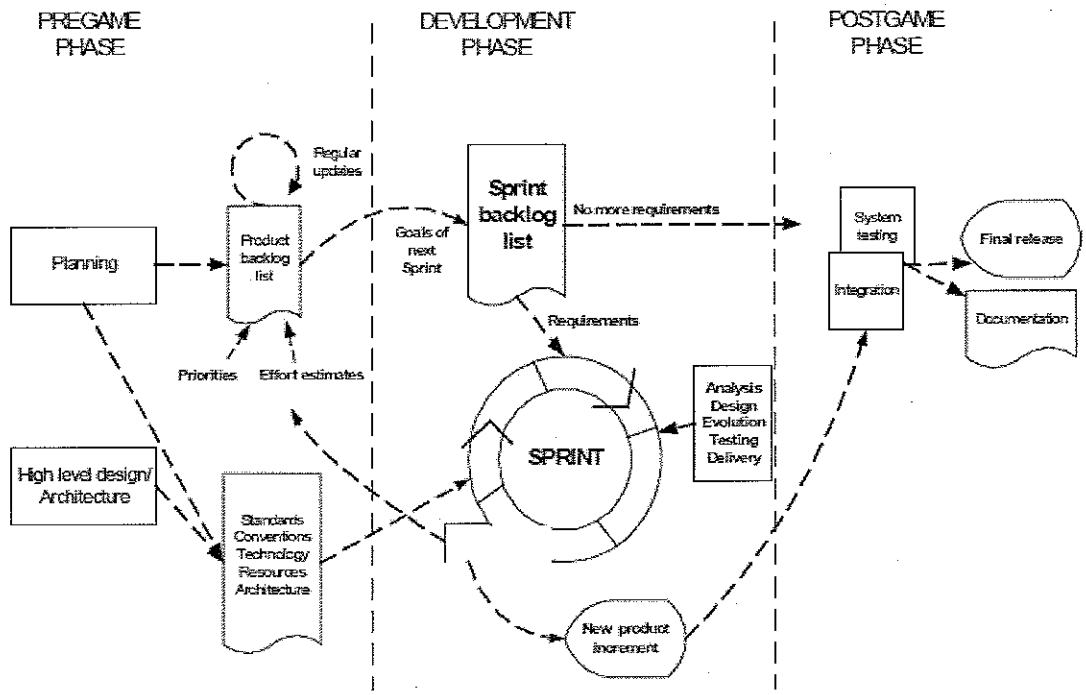
Kommunikasjon og interaksjon er et krav i Scrumprosessen. I FIDO har de personer som bruker hjemmekontor mye, kommunikasjonsmetodene bør derfor tilrettelegges nøye, og formålet og viktigheten av kommunikasjonen må forklares til alle som skal være en del av prosjektgruppen.

Etterspillfasen (Postgame Phase) inneholder oppgaver som testing, integrasjon av funksjonen i systemet og dokumentasjon.

Denne fasen går man inn i når et av kravene er fullført. (Se Figur 6)

Fordeler og ulemper i forhold med FIDO

Denne fasen øker sannsynligheten for at resultatet blir godkjent av kunden, siden kunden var så mye involvert i utviklingen av prosjektet.



Figur 6: SCRUMs lifecycle (Abrahamsson et al. 2002)

SCRUMs roller

- Scrum Master (Scrum Mester) er en ledelsesrolle som er ansvarlig for å sikre at prosjektet, verdiene og regel av Scrum er forstått og at det avanserer som planlagt. Scrum Master'en er også ansvarlig for kommunikasjonen mellom Scrum Team'et, ledelsen og kunden i løpet av prosjektet. Scrum Master'en er altså ansvarlig for å sikre at laget er beskyttet mot distraksjoner og hindringer.

- Product Owner (Produkteier) er ofte kunden, men kan også være en del av den intern organisasjonen. Product Owner'en representerer stemmen til kunden og er ansvarlig for å forvalte, styre og å synliggjøre prosjektets Product Backlog.

- Scrum Team (Scrum Lag) er prosjektets utviklingsgruppe som selv har myndigheten til å bestemme seg for de nødvendige tiltakene og til å selvorganisere seg slik at de

kan oppnå målene av hver Sprint. Scrum Team'et er også med på å vurdere størrelsen på Sprint Backlog'en som skal utvikles. Vist det vil ta mer en den angitte Sprint tidsrommet, blir dette fortalt til Scrum Master'en. Scrum Team'et er formet av 5 til 9 medlemmer som har kompetanse på ulike områder, og består blant annet av folk som analyserer, designer, programmerer og tester.

- Users (Brukere) er dem som systemet blir laget til.

- Stakeholders (Interessepersoner) er menneskene som muliggjør prosjektet, men som ikke er direkte involvert i prosessen. Dette inkluderer ledelsen.

- Konsulteringseksperter (Consulting Experts) gir ekspertise som er ikke krevd på hver sprint.

Scrums terminologi

- Story (Historie) er en kunde fokusert beskrivelse av den verdsatte funksjonaliteten.

- Product Backlog (Gjenstående arbeid) er en liste av Story'ene som skal utvikles i en prioriter rekkefølge (se Figur 6).

- Sprint (Iterasjon) er et tidsrom (vanligvis fra 2 til 4 uker) hvor Scrumteamet forplikter seg til å utvikle et sett med Story'er.

- Burn Down Chart (Nedbrytingsdiagram) viser sprintens daglige fremgang gjennom hele sprinten.

Scrums sprint prosess

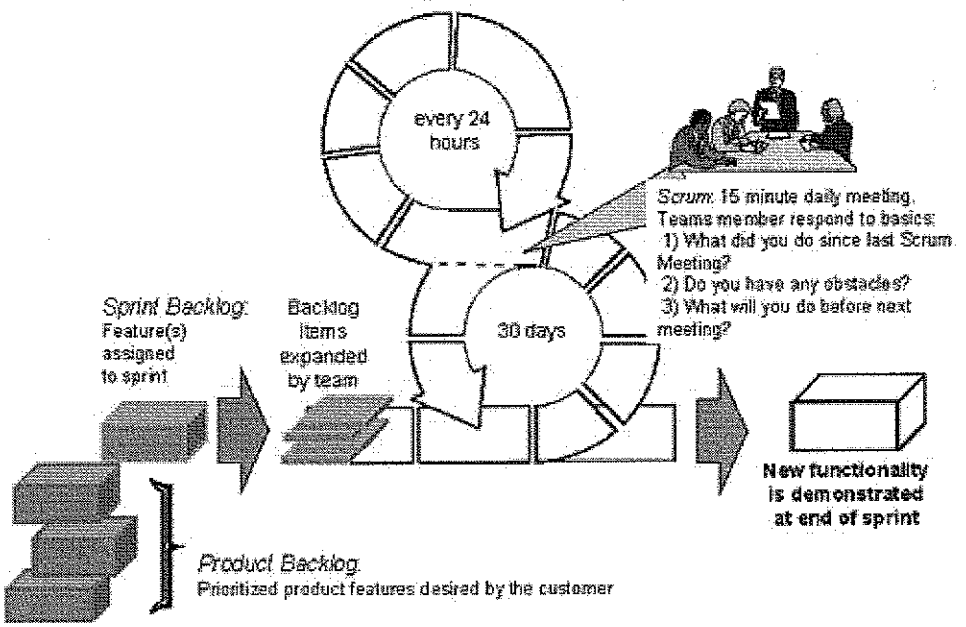
Scrums sprint prosessen inkluderer hovedsakelig stegene som er beskrevet herunder.

Scrumteamet starter med å velge ut krav fra toppen av backlog'en, dvs. det som er prioritert som viktigst, slik at de kan planlegge arkitekturen og designet. Dette danner grunnlaget for hva som skal utvikles i sprint'en. Det er så opp til Scrumteamet å organisere seg og arbeidet sitt i hver sprint.

Scrum har daglige formelle møter som er ikke varer mer enn 15 minutter.
I løpet av møtet svarer alle fra laget på tre spørsmål:

- Hva har gjort du siden forrige Scrum møte?
- Har du noen hindringer?
- Hva skal du gjøre til neste Scrum møte?

På slutten av en sprint, demonstrerer de ~~krav~~funksjonaliteten til kunden og ledelsen. (Se Figur 7)



Figur 7: SCRUMs sprint prosess

4.4 Konklusjon

Det største problemet for FIDO er at kompleksiteten i systemet hos ETC ikke kan avdekkes med en gang, slik at det hele tiden kommer endringer underveis.
Det finnes flere andre utfordringer også, men denne ene, det at det skjer stadige endringer underveis, er nok til at vi ikke bør velge en strukturert metode som Stradis, som er basert på fossefalls-tankegangen, og dermed gjør det til en forutsetning at man kan klare å avslutte f.eks design-fasen før man går videre til programmeringsfasen.
Både Scrum og FDD takler endringer underveis, så hvilken velger vi da?
Vi har sammenliknet alle 3 metodene i tabellen under, i forhold til om de kanskje kunne hjelpe FIDO med ETC-prosjektet, og vi ser at FDD og Scrum kommer nokså likt ut.
Erfaringene sier oss at det er heller usannsynlig at vi kommer til å følge én metode fullt ut. Vi må derfor prøve å velge den som gir oss det beste grunnlaget til å tilpasse en metode til FIDO på, og siden FIDO ikke har brukt noen metoder før, ser vi på det som viktig at metoden som velges i første omgang, er lett og sette seg inn i og få en viss grad av oversikt over.
Siden ingen i gruppen hos oss har førstehåndserfaring med noen av de to metodene, har vi prøvd å sette oss i FIDOS sted, for å se hvilken metode som virker mest tiltrekkende. Valget faller på Scrum. Vi har, i tabell 1 nedenfor, vist at den kanskje kan dekke de fleste utfordringene, og vi synes informasjon om Scrum har vært lettere tilgjengelig enn informasjon om FDD.
I tillegg mener vi at Scrum også egner seg som metodegrunnlag i en organisasjon som FIDO, som tar sikte på å arbeide med flere prosjekter av ETC-typen senere, og kanskje også større prosjekter.
I kapittel 5 kommer vi nærmere inn på hvert enkelt punkt, og vil forsøke å vise hvordan bruk av Scrum, med evt. tilpasninger, muligens ville kunnet løst noen av utfordringene FIDO hadde med ETC-prosjektet.

Kategorier	FIDO problemer	Metoder		
		STRADIS	FDD	SCRUM
Endringer	Kompleksiteten avdekkes et stykke ut i løpet	Nei	Ja	Ja
	Avdekket etter hvert behov for enda et system	Nei	Ja	Ja
	Det ekstra system økter også i kompleksiteten etter hvert	Nei	Ja	Ja
	A&D-fasen strekker seg langt inn i programmeringsfasen	Nei	Ja	Ja
	ETC får ikke noe skikkelig estimat for tidsbruk og kostnadsbruk	Ja/Nei	Ja	Ja
	Programmererne konsentrerer seg om de deler av kravene som virker mest stabile, uavhengig av hva som er viktigst å få til først, eller viktigst for kunden	Nei	Ja	Ja
Kommunikasjon	Terminaldivisjonen, representert ved Mike, har sine tvil og føler at de ikke får dekket behovene sine	Nei	Ja	Ja
	Kommunikasjonen mellom A&D-teamet og programmeringsteamet er muligens ikke god nok	Ja/Nei	Ja	Ja
	FIDO bruker egne folk og i tillegg innleide som flytter til Norge, og innleide som bor i Norge allerede	Ja	Ja	Ja
	Stig er bare på kontoret et par ganger i uka, jobber helst hjemmefra	Ja	Nei	Nei
	Brukerne sitter svært spredt	Ja	Nei	Nei
Pris	Time-for-time pris i A&D-fasen	Ja	Ja	Ja
	Fastpris for programmeringsfasen	Ja	Tja	Nei
Andre	FIDO mangler helt en metode for systemutvikling	Ja	Ja	Ja
	Mike liker ikke Ivan og vi vet at det er merkbart i organisasjonen	Nei	Tja	Tja
	FIDO mangler en prosjektleder som har ansvaret for HELE prosjektet, roller/ansvar?	Ja/Nei	Ja	Ja
	Janis har roller i begge prosjektorganisasjonene, behov for klargjøring av roller/ansvar?	Nei	Ja	Ja

Tabell 1: Metode sammenligning

5 Plan for innføring av valgt metode

Vi vil her ta utgangspunkt i de problemene FIDO og ETC hadde under systemutviklingen, og forklare hvordan vi ville løst disse problemene ved hjelp av metoden vi har valgt.
Vi mener at Scrum er et godt utgangspunkt som metode for de utfordringene man står overfor. Scrum vil kunne løse en god del av problemene til FIDO, samtidig som den gir FIDO et godt grunnlag for systemutvikling i videre prosjekter. Dette er fordi metoden er relativt enkel å få en oversikt over, så lenge du er åpen for tilpassninger som ikke nødvendigvis er en del av metoden.
Metoder blir som regel ikke fulgt til punkt og prikke i virkeligheten, men blir modifisert etter behov.

5.1 OPPLÆRING

Det er i første omgang viktig at alle som skal følge metoden gis tilstrekkelig innføring og opplæring. I Scrum er det dessuten et krav at alle medvirkende kjenner alle faser av metoden. Første bud blir derfor å tilrettelegge et kurstilbud i en eller annen fasong. Det kan være at man skal sende de som skal delta på i utviklingen, på et kurssted som ligger utenfor de daglige kontorfasilitetene. Erfaringsmessig får man mer ro da, og kan konsentrere seg om det som foregår på kurset. Det er jo dessverre ikke slik at all annen virksomhet i bedriften stopper opp fordi om noen har funnet ut at de trenger opplæring, de daglige oppgavene krever sitt og pågående prosjekter kan ikke settes på "hold", hvilket betyr at mange kan ha bruk for samtaler og lignende med de personene som egentlig skal konsentrere seg om opplæring.
Akkurat når det gjelder FIDO, er de så få personer i utgangspunktet, at hvis tilfeldighetene allikevel vil det slik at samtlige kan delta på likt, så kan det muligens svare seg å innhente kurskrefter som kan brukes på kontoret. Kursholderne får da samtidig den fordelen at de kan se hvordan arbeidsplasser og møterom er lagt opp, og kan hjelpe til med evt. tilpasninger som skal til for å få mest mulig effekt av metoden.
Selv om ETC innehar egen IS-kompetanse, så skal disse personene "kun" være med som kunderepresentanter i metoden. De skal derfor ikke være med på denne første kursingen, den er kun for FIDO. Dette virker logisk siden FIDO skal bruke denne metoden på systemutvikling hos mange forskjellige kunder i fremtiden, og det er først hvis man har behov for det at kunden gis en innføring i hvordan metoden fungerer. Det kan i ETC-tilfellet f.eks dreie seg om forklaringer i forhold til pris-setting og hvordan og hvorfor det gjøres slik eller sånn, men det kommer vi tilbake til.

5.2 FORDELING AV ROLLER

5.2.1 Scrum Master

Som Scrum Master burde FIDO velge Bente. Bente har lang erfaring som prosjektleder, og selv om en Scrum Master ikke er en prosjektleder så vil hun, med den erfaringen hun har, kunne fylle de oppgavene som rollen som Scrum Master pålegger henne. Hun skal sørge for at alle får den nødvendige opplæring, at de fordelte rollene er forstått, at prosjektoppgavene er forstått, og sørge for at ingenting forstyrrer gruppen når den arbeider, og at evt. hindringer blir ryddet av veien. Det kan for eksempel være at hun må leie inn flere folk, både pga

arbeidsmengde og fordi prosjektet trenger ekstra, eller annen, kompetanse enn den som er der fra før.

Scrum Master blir da:

- Bente

5.2.2 Product Owner

Her gjør vi en liten vri i forhold til det som opprinnelig er fastlagt i Scrum. Product Owner skal være kundens stemme inn i prosjektet, det er han som bl.a. skal velge fra backlog'en hva som skal prioriteres. Dette er en veldig viktig rolle, med mye ansvar, og det er nok ikke uten grunn at Scrum i utgangspunktet mener at det bare skal være én person. Det er muligens lettere for prosjektet å ha én person å forholde seg til fra kunden, særlig når det gjelder slike viktige saker som prioriteringsrekkefølge. Men i ETC sitt tilfelle, er det et annet forhold man må ta hensyn til, og det er samspillet mellom Mike og Ivan. I et forsøk på å få disse to til å samarbeide ville vi gitt dem et felles ansvar som Product Owner. I utgangspunktet ville det vært naturlig at Ivan, som er IS-ansvarlig i ETC, skulle hatt denne rollen alene. Valget av Mike i tillegg kan forsvares med at Mike representerer utekontorene. Han har også samarbeidet med andre leverandører før og har dermed en viss erfaring. Som et team vil Mike og Ivan dermed utfylle hverandre.
Det store spørsmålet er: *Vil* de jobbe sammen?
Kaufmann & Kaufmann (1996) sier at man kan ansvarliggjøre personene simpelthen ved å spørre dem om de er i stand til å jobbe sammen. Forutsetningene er at arbeidsoppgaven er attraktiv (og det passer med Product Owner), og den ene må ikke settes til å lede den andre, de må jobbe på likt nivå. Videre mener de at oppgaven må være klart definert slik at de begge har samme forståelse av hva målet er, og dette mener vi er perfekt for de avgjørelsene de skal ta når det gjelder prioriteringer. I Scrum kan hver lille del av backlog'en representere en funksjon, hver iterasjon tar forholdsvis kort tid, og til sammen er det lett å definere delmål for hver gang. Det er også lett å se at ting går framover, og både Mike og Ivan må stå sammen om å motta ros eller kritikk. Det er ikke noe mål at de skal like hverandre, de skal bare kunne jobbe sammen.

Product Owner blir da:

- Ivan og Mike

4.2.3 Scrum Team

Stig, som har jobbet for ETC før og var med å utvikle logistikksystemet der, og Even, som har greie på programarkitektur og integrasjon med andre systemer, danner sammen det grunnlaget FIDO har av indre kunnskap om systemene i ETC. De to sitter på en masse informasjon som prosjektet trenger. Problemet er at Even har vanskelig for å gi fra seg noe informasjon, og Stig har hjemmekontor mesteparten av tiden. Prosjektet er avhengig av at disse sitter lett tilgjengelig og at det er mulig å få ut den nødvendige informasjonen fra dem.
Når det gjelder Stig kan man kanskje redde situasjonen ved at man inngår en avtale spesielt tilpasset ham: Hans daglige Scrum-møter må settes opp ved hjelp av videokonferanse, selv om vi vet at det forringer verdien av møtene noe. Men vi tror det vil være vanskelig å overtale/pålegge ham å møte på kontoret hver dag, da kan man raskt få en konflikt til, i stedet for et samarbeid. Den forringelsen som skjer ved å bruke videokonferanse er nok her den

minste prisen å betale. I tillegg må han komme innom kontoret minst et par ganger i uken slik som før, og i tillegg må han forplikte seg til å komme inn ekstra hvis gruppen skulle ha behov for det.
Even derimot, sammen med alle de andre, må møte på Scrum-møter hver dag, der han vil måtte prøve å informere de andre om hva status er for hans del. At det vil bli vanskelig virker sannsynlig, og ivaretakelsen av informasjonsdeling kan muligens reddes ved at man setter ham i par med en av de andre under selve arbeidet med programmeringen. At han er stille betyr ikke at han ikke kan like å jobbe sammen med noen. Da kan den andre parten formidle status på møtene. Vi ville derfor valgt å parre ham med Tero, selv om Tero i utgangspunktet skulle hjelpe Øystein. Tero har den riktige kompetansen for dette.
Dermed kan Øystein og Janis sitte sammen, og fungere som et programmeringsteam.
Maung må sammen med Bjørn Erik sørge for at de for tak i de riktige kravene etter hvert og sammen må de lage en teknisk design for systemet. Dette gjør de litt etter litt, i følge med det som kreves i hver sprint.

Scrum-teamet ser da slik ut:

- Stig (Mye hjemmekontor, daglige Scrum-møter via videokonferanse, kan kalles inn mer enn før hvis nødvendig)
- Even og Tero programmerer i team, og sitter på kontoret hver dag. Deltar i daglige Scrum-møter
- Øystein og Janis programmerer i team og sitter på kontoret hver dag. Deltar i daglige Scrum-møter
- Bjørn Erik hjelper teamet med å definere kravene, og også med det tekniske designet av systemet
- Maung må hjelpe til med de tekniske løsningene og ha oversikten over det som skjer på dette området, og i tillegg kan han hjelpe til med programmeringen.

5.3 Konklusjon

Vi vurderer det slik at det er flere grunner til at vi velger at noen av personene skal sitte sammen og programmerer. Vi har forklart over det som er særskilt årsak til at vi setter Even og Tero sammen, men i tillegg er synes vi det er lurt at de innleide får sitte sammen med hver sin fast ansatte. For det første mener vi at de innleide raskere kan bli kjent med bedriften de er innleid i og dermed raskere kan komme inn i de oppgavene de skal gjøre. For det andre er det ikke så farlig om de fases ut etter at dette prosjektet er over fordi den kompetansen de har tilegnet seg gjennom prosjektet også sitter hos en av de faste. Til slutt er det også en forsikring dersom noen skulle bli syke eller av andre grunner blir borte underveis.
Teamfordelingen over, samt aktivitetene i Scrum sine sprinter, vil til sammen legge til rette for god kommunikasjon i prosjektet. Forståelsen for oppgaven vil øke, siden det blir lettere for alle å ha oversikt over hvilke oppgaver prosjektet egentlig har og hva som forsøkes løst akkurat nå. Tidsfrister og avhengigheter blir tydelig for alle siden de er satt i hver sprint.
Altså. Kommunikasjonsproblem løst.

Pga den gode oversikten over gjeldende og gjenstående problemstillinger, vil endringer også kunne takles bedre. Hver mulige endring vil bli diskutert, fastlagt og prioritert sammen med de andre oppgavene. Man vil kunne endre prioriteringslisten etter hver sprint, og til enhver tid ha oversikt over hva som er viktigst akkurat nå. Pga den korte tidsintervallet sprintene opererer med, er det mulige å gjøre endringer raskt hvis det er påkrevd.
Altså: Endringsproblem løst.

Når det gjelder prisen var ETC med på å godta en time for time pris i analyse & design delen av prosjektet. Vi tror derfor det er gode muligheter for at de ville godtatt en prismodell som først ble satt etter 1-2 iterasjoner (sprinter) i metoden. FIDO kunne motivert ETC ved nettopp å bruke metodens krav om å inkludere kunden i utviklingen, forklare hvordan metoden er tenkt brukt, og også brukt den korte tiden én iterasjon tar som inspirasjon til å prøve dette ut. Ivan og Mike kan tett på observere hvordan FIDO jobber og se hvor mye/lite arbeid som legges ned i hver iterasjon. Da blir det lettere å forhandle om en pris, hvis alle har noenlunde samme oppfattelse av hvor mye arbeid som kreves.
Det finnes også forskjellige teknikker for å kunne estimere en pris, for eksempel gi hver oppgave såkalte "story-points", som sier noe om arbeidsmengden som kreves for å fullføre "storyen" (eller funksjonen/caset), og som kan brukes til å regne ut en ca pris.
Ved å godta betalingsfastsettelse pr funksjon som fullføres er det lettere for kunden OG leverandøren å kunne forholde seg til gjenværende oppgaver kontra pris og nytteeffekt, og derved lettere kunne vurdere om det lønner seg å gå videre eller stoppe prosjektet.
Altså: Prisfastsettelsesproblem løst.

Under punktet "Andre" i tabellen, listet vi opp noen tilleggsutfordringer. De er naturlig løst ved hjelp av det som er forklart over: FIDO mangler ikke lenger en metode, Mike og Ivan jobber forhåpentligvis greit nok sammen i rollen som Product Owner, "prosjektleder"-begrepet er ikke lenger med i bildet, og Janis forholder seg til Scrum-teamet sitt.
Altså: Andre utfordringer løst.

6 Refleksjon

Vi har lest mye på internett, i bøker og artikler under vårt arbeid med oppgaven, og det har vært vanskelig å holde oversikten over de forskjellige metodene, hvordan de har utviklet seg og kanskje særlig hvordan status er akkurat nå. Overraskende mye er skrevet for mange år tilbake, og mange har gjort nye og andre erfaringer siden, enn det som er beskrevet.
De erfaringene vi har fått oss fortalt fra andre underbygger følelsen av at det finnes et konglomerat av metoder hvor beskrivelsen i varierende grad er vellykket.
Og skulle velge én metode, og attpå til helst den riktige, forekommer oss ikke som noen enkel oppgave.
Det har derfor vært lettere å lese om de store trekkene, dvs. de største forskjellene på for eksempel strukturerte og agile metoder. Det var de store trekkene som ga oss retningen, siden kunne vi egentlig konsentrere oss om å finne den metoden innenfor retningen som passet best. Men den behøver nødvendigvis ikke passe perfekt, den må bare passe godt nok. I det virkelige liv vil metoden etter hvert tilpasses prosjektene eller bedriften den brukes i, og bli det som kalles methods-in-action, den metoden prosjektet eller bedriften virkelig bruker.
Vi har snakket med mennesker som arbeider med systemutvikling i forskjellige bedrifter, og følelsen forsterker seg: det er litt tilfeldig akkurat hvilken metode de valgte gitt at noen få grunnleggende krav ble imøtekommet av metoden. Derfra og videre er det gjort mange spesifikke tilpasninger og de foregår kontinuerlig.
Man kan kanskje innvende at dette nettopp *er* en del av for eksempel agile metoder. Og agile metoder blir da også det naturlige valget for systemutviklingsprosjekter i 2007 forutsatt at særlig disse 2 kravene er oppfylt:
Systemet må ikke fordre store sikkerhetskrav.
Prosjektteamet må, så langt det er mulig, bestå av de samme personene så lenge som mulig.
Andre faktorer gjelder også, som for eksempel at det er en fordel å ha med prosjektdeltakere med høy kompetanse, det er en fordel at kontormiljøet rent fysisk er slik at man kan innrette seg slik metoden ønsker, det er en fordel at prosjektmedlemmene kjenner metoden best mulig osv. Men dette er faktorer som har noenlunde lik tyngde for alle metoder, de er ikke spesielle for de agile.
Videre har vi gjort oss den erfaringen at forskjellen mennesker imellom er av betydning for hvordan et team fungerer. Man kan ikke tro at det er likegyldig hvilke mennesker man gir de enkelte roller, selv om disse menneskene faglig sett har nøyaktig de samme forutsetninger og erfaringer. Enkeltmenneskers evner til å samarbeide, kjemien mellom individer og dagsformen har innvirkning på prosjektet. Bør man derfor ha med disse elementene når man utvikler en systemutviklingsmetode? Faktisk nei, mener vi. Dette er et eget fag innenfor organisasjonspsykologi og det er ikke systemutviklingsmetodene som sådan som skal ta høyde for disse forskjellene, de er jo godt beskrevet andre steder, nemlig nettopp under organisasjonspsykologien. Derimot er det prosjektleders ansvar og ha kunnskap om også disse emnene slik at han kan sette sammen teamet sitt både på grunnlag av faglige kvalifikasjoner innenfor systemutvikling, og på grunnlag av de momentene som finnes i organisasjonspsykologien.

Referanseliste:

Abrahamsson, P., Salo, O., Ronkainen, J. og Warsta, J. (2002) Agile Software Development Methods, *VTT Publications* 478, Espoo

Avison, D. og Fitzgerald, G. (1995), *Information systems development, Methodologies, Techniques & Tools.* McGraw-Hill, London.

Avison, D. og Fitzgerald, G. (2006, 4th edition), *Information systems development, Methodologies, Techniques & Tools.* McGraw-Hill, London.

Bass, B (2001), *fra transaksjonsledelse til transformasjonsledelse: Å lære å dele en visjon.* Vecchio, R. P (2001) *Makt, politikk og innflytelse* (begge i Martinsen (red), *Perspektiver på ledelse.* s. 29-54 og s. 187-201), Gyldendal Akademiske, Oslo.

Coad, P., Lefebvre, E. & De Luca, J. (1999). Java Modeling in Color With UML*: Enterprise Components and Process.* Prentice Hall International, Upper Saddle River, NJ.

Fitzgerald, B., Russo, N.L. and Stolterman, E. (2002), *Information Systems Development:* Methods in Action. McGraw-Hill, London.

Gane, C. and Sarson, T. (1979), *Structured analysis, design and implementation of information systems.* Structured Systems Analysis. Improved System Technologies Inc., New York

Kaufmann, G. og Kaufmann, A. (1996) *Psykologi i organisasjon og ledelse,* 2. utgave, s. 277-289 Problemløsning og beslutninger i grupper. Fagbokforlaget, Bergen.

Palmer, S.R., & Felsing, J.M. (2002). *A Practical Guide to Feature-Driven Development.* Prentice Hall. Upper Saddle River, NJ

Sørensen, A. og Egeland, E. S. (2007) *Agile Systemutviklingsmetoder i praksis.* Masteroppgave HiA.

Vedlegg

Oppsummeringsrapport til Calle

Beskrivelse av 3 metoder hvorav 1 er valgt i konklusjonen

METODE 1, STRADIS

STRADIS er en strukturert metode som har tatt i bruk flere forskjellige teknikker. Teknikkene finns i andre metoder, og mange oppfatter metoden som et sammendrag av de metoder som er basert på å skulle bryte ned problemstillinger i mindre deler. I tillegg gjør den seg bruk av dataflytdiagrammer.

Her følger en kort beskrivelse av STRADIS, hvor vi vektlegger å peke på de punkter som vil lette/vanskeliggjøre/mangle i forhold til FIDOs utvikling av ETC sine systemer:

Innledningsfasen

Innledningsvis ønsker man først og fremst å forsikre seg om at det systemet man velger å utvikle er det systemet som mest sannsynlig vil kunne gi bedriften en ekstra fordel i et konkurranseutsatt marked. Det viktigste kriteriet blir da kostnader og gevinster ved et nytt system, evt. ved hvert av alternativene.
Man starter med å samle inn informasjon fra ledere og brukere i bedriften. Videre går man gjennom eksisterende dokumentasjon og belyser alternativene i forhold til strategiske planer som måtte finnes i bedriften. Man lager et dataflytdiagram som viser en oversikt over det eksisterende systemet og dets brukergrensesnitt., og en oversikt over den tid og kostnad man vil bruke på å gjøre en mer detaljert undersøkelse. I tillegg kan man gjøre noen grovere overslag for det endelige resultatet. Denne fasen er beregnet til å ta mellom to dager og fire uker, avhengig av størrelsen og viktigheten av systemet.
Avslutningsvis ender denne fasen i en rapport som forelegges ledelsen, som så kan ta avgjørelsen om hvorvidt de vil gå videre til en mer detaljert studie-fase, eller om de vil avslutte der og da. Velger de å gå videre, aksepterer de samtidig kostnadene for den detaljerte fasen, men ikke nødvendigvis og fortsette med utviklingen etter det.

Fordeler og ulemper i forhold til FIDO:
Ved å bruke innledningsfasen til STRADIS vil man kunne kartlegge ETC's eksisterende system, og også vite litt mer om den tid og kostnad man vil bruke på en mer detaljert undersøkelse.
Den gir ledelsen i ETC en mulighet til å få en fast pris på detaljkartleggingsfasen, og det passer bra siden ETC's ledelse helst vil bruke fast-pris-prinsippet. På den annen side vet vi at detaljkartleggingsfasen i dette tilfelle ikke ville avdekket alle behovene ETC har, fordi behovene ikke kom fram før på et mye senere tidspunkt, derfor vil ikke STRADIS metoden kunne gi de svarene FIDO trenger for å kunne fullføre fasen. Det betyr at FIDO må gjøre mye mer arbeid enn de først hadde trodd hvis de virkelig skal komme til bunns av problemstillingen i denne fasen, og de vil dermed tape penger på det. Og mest sannsynlig vil de allikevel ikke kunne gi ETC de svarene som trengs.
Konklusjonen til denne fasen blir allikevel at den kan brukes til en første kartlegging, men at å gi en fast pris på detaljkartleggingsfasen vil være vanskelig, særlig fordi vi vet at en detaljkartleggingsfase kanskje ikke engang er en fase som vil la seg gjennomføre i dette prosjektet.

Detaljkartleggingsfase

Etter denne fasen konsentrerer STRADIS-metoden seg om det eksisterende systemet.
Man prøver her å kartlegge hvem som er brukere av systemet, kostnader ved gammelt og nytt system og fordeler ved et nytt system.
Det lages et dataflytdiagram for hver viktige prosess og til slutt et overslag over tids- og kostnadsbruk for neste fase.

STRADIS kartlegger brukerne på 3 nivåer:

- De som bestilte systemet, dvs. de lederne som innså at behovet var tilstede og deretter foreslo å bestille et nytt.
- Mellomledere i de berørte avdelingene
- Sluttbrukerne, dvs. de som direkte vil bruke systemet.

Kort summert skal denne fasen resultere i:

- En detaljert definisjon av de forskjellige brukerne av systemet, funksjonene, sammenhengene, hvilke arbeidsoppgaver som vil bli berørt, jobbeskrivelser osv
- En logisk modell av eksisterende system
- En oversikt over de fordelene som kan gis av et nytt system, både i form av effektivitet og besparelser, men også en oversikt over evt. økninger av utgifter i form av vedlikehold, større lagringsbehov osv
- Oversikt over lovbestemmelser man muligens må ta hensyn til, konkurransedyktigheten til systemet, systemkostnader og et budsjettforslag for neste fase, gjerne med alternativer.

Fordeler og ulemper i forhold til FIDO:
Kartlegging av brukerne, og oversikt over kostnadene ved det gamle systemet er oppgaver FIDO vil kunne nyttiggjøre seg fra denne fasen. De vil også kunne lage dataflytdiagrammer over de prosessene som de til nå har avdekket, men vi vet fra oppgaveteksten at alle behov ikke *kan* avdekkes her.
Det blir derfor også vanskelig å gi et overslag over tids- og kostnadsbruk for *hele* neste fase. Spørsmålet er om de i det hele tatt kan kjøre neste fase som én fase.

Definering og design av de alternative løsningene

Her følger en fase hvor man definerer alternative løsninger på problemene til det eksisterende systemet.

Først tar man for seg de organisatoriske målene som ble kartlagt i innledende fase og bearbeider disse målene fram til systemmål. Et systemmål vil si å gi en beskrivelse av hva et nytt system burde kunne gjøre for å hjelpe ledelsen å nå de organisatoriske målene. Videre må systemmålene bør være spesifikke og målbare, ikke generelle.

Analytikere vil deretter bruke disse målene til å produsere logiske dataflytdiagrammer for det nye systemet. Også dataflyten mellom det som vil gjenstå som manuelle deler av systemet, slik som å ta i mot bestillinger via telefon/e-post, kundeoppfølging før kontraktsinngåelse, hvordan tar man vare på dokumenter/e-post/samtaler som fører fram til en kontrakt, hvem legger inn kontrakten og dermed starter opp den nye automatiserte delen av systemet, osv.

Slike dataflytdiagrammer bør ende opp såpass detaljert at man ser at systemkravene er imøtekommet.

Analytikere og designere skal her jobbe sammen om å komme opp med 3 forskjellige designkategorier:

4. Lavbudsjett, rask implementering, møter ikke nødvendigvis alle krav
5. Middels stort budsjett, noe senere implementering, oppfyller de fleste kravene
6. Høyt budsjett, versjon som oppfyller alle krav.

Her gir man et røft estimat for kostnad, fordeler/gevinster, tidsforbruk, maskinvare, programvare.
Utfallet av denne fasen skal være en rapport som inneholder de tre forlagene over og som skal presenteres for ledelsen, og ledelsen må bestemme seg for ett av alternativene. Rapporten bør inneholde følgende:

- et dataflytdiagram over eksisterende system
- begrensninger i eksisterende system
- logisk dataflytdiagram over nytt system

Til hvert av de 3 alternative løsningene bør følgende være med i rapporten:

- den delen av dataflytdiagrammet som vil bli implementert
- UI (brukergrensesnitt, terminaler, rapporter osv)
- Estimert kostnad og gevinst
- Implementeringstimeplan
- Risiko

Fordeler og ulemper i forhold til FIDO:
Idéen med 3 alternative løsninger virker forlokkende, da det gir ledelsen muligheten til å senke kravene til funksjonalitet mot at produktet leveres i rett tid, evt. at man utvider tidshorisonten og krever full funksjonalitet, eller at man oppfyller alle krav innenfor tidsfristen men dermed må øke ressursene og få et dyrere system.
Problemet i dette tilfelle er at FIDO ikke vil kunne gi ETC en full oversikt over *hele* systemet, men idéen burde kunne brukes på de delene av systemet man har oversikt over. Det betyr at STRADIS igjen gir oss noen detaljer vi kan bruke, men kun det, vi kan ikke følge metoden helt og holdent her heller.

Fysisk design

Det valgte alternativet blir så utviklet til et spesifikt fysisk design av designerne.
Denne fasen består av et antall aktiviteter, bl.a.:

- Detaljerte dataflytdiagrammer, med beskrivelse av feilhåndtering og unntak av alle slag, og all prosesslogikk, rapport og skjermbildeformater. Rapporter og skjermbildeformater bør valideres av brukerne..
- Fysiske filer el. databaser blir designet. De baseres på dataflytdiagrammene, som viser hvilke data man har behov for å få lagret i den gitte prosessen som beskrives. Mange av dataflytdiagrammene vil etter hvert overlappe hverandre med hensyn til data som har behov for lagring.
- Datalagrene rasjonaliseres, teknikker for normalisering blir brukt for å slå sammen og forenkle datalagrene i logiske grupperinger.
- Modellere et hierarki av funksjonene, på bakgrunn av innholdet i dataflytdiagrammene. Her prøver man å finne ett av to mulige strukturer for

dataprosesseringen: "transform-centered", hvor alle transaksjonene følger veldig like prosesseringsveier(stier), eller "transaction-centered", hvor transaksjonene krever forskjellig prosessering.

Til slutt definerer man hvilke manuelle jobber det nye systemet vil kreve, dvs. hvor grensene for det automatiserte systemet går.

Alle oppgavene over utføres til et slikt detaljert nivå at man er stand til å gi et pålitelig anslag over kostnad for utvikling og betjening av det nye systemet.
De vesentligste sidene av disse kostnadene identifiseres ved:

- arbeidstimer som kreves for å utvikle de modulene man har bestemt
- Maskinvare og programvare som kreves
- Perifere enheter (skrivere og lignende) og datakommunikasjonskostnader
- Arbeidstimer til å utvikle brukerdokumentasjon og opplæring av brukere
- Tiden brukerne bruker på systemet
- Arbeidstimer som e skal til for å vedlikeholde og videreutvikle/rette systemet i dets levetid

Fordeler og ulemper i forhold til FIDO:
I forhold til hovedingrediensen i denne fasen, nemlig at alle rapporter og skjermbilder osv skal designes til bunns, er umulig, fordi man enda ikke har brakt alle kravene fram i dagens lys.
Det er muligens aktuelt å bruke prinsippet, som beskrevet over, til å utvikle mindre moduler av systemet, men i den sammenheng vil det ikke være aktuelt å for eksempel designe databaser for *hver enkelt* modul eller funksjon. Flere av de andre kravene vil også være ting man gjør for hele systemet på en gang, så alt i alt vil vi konkludere med at denne fasen av STRADIS ikke lar seg bruke for FIDO's prosjekt.

Implementeringsfasen

STRADIS er en metode som tar for seg hovedsakelig analysedelen, noe mindre opptatt av designdelen, og nesten ikke har noe skrevet om implementering, men det finnes noen punkter som gir en pekepinn om hva som skal til for å fullføre systemet:

- lage en implementeringsplan som inkluderer plan for testing og aksept av systemet
- utvikle applikasjonen og database/datakommunikasjons-funksjonene og samtidig
- "befolke"(populere) databasen
- Teste og forsikre seg om at man har aksept for hver del av systemet
- Forsikre seg om at systemet oppfyller de performance-kriterier som er definert i systemmålene, og utsette det for realistisk datahåndteringsbelastning for så å måle responstid og throughput (mengden av dataflyt).
- Sette systemet i gang i det virkelige liv, sjekke evt. flaskehalser og tune det.
- Sammenlikne systemet og performance-graden med det som opprinnelig var målet, og sørge for å rette alt som ikke svarer til forventingene.
- Analysere og se etter mulige forbedringsområder, prioritere disse, og så la systemet gå over i en vedlikeholdsfase.

Fordeler og ulemper i forhold til FIDO:
Etter hvert som STRADIS sammenliknes med behovet for systemutviklingsmetoder for FIDO ser man at implementeringsfasen til STRADIS ikke så lett lar vurdere i den sammenheng. Dette er fordi at vi har konkludert de tidligere fasene med at man kan ta elementer fra dem, og kun det, fordi prosjektet til FIDO vil måtte brytes ned i mye mindre moduler. Dermed vil hver modul måtte ha sin egen test- og implementeringsplan, som vil skille seg fra det som er å teste hele systemet på en gang. Implementeringsfasen i STRADIS kan derfor ikke brukes.

Forhold ved FIDO som ikke blir tatt hensyn til ved bruk av STRADIS metoden:
Det er mange forhold ved FIDO som ikke kan tas opp til vurdering hvis man bruker STRADIS. Det største problemet er at kompleksiteten ikke avdekkes med en gang, slik at det hele tiden kommer endringer underveis. STRADIS håndterer i liten grad endringer, i beste fall må man satse på å begynne kartleggingsfasen på nytt hver gang en endring oppstår.
Dette forholdet er av så stor betydning at andre forhold som ikke blir tatt hensyn til, slik som at brukerne sitter svært spredt, at man har brukere med store motsetninger internt i ETC osv ikke behøver å kommenteres punkt for punkt her, men kommer med i en endelig konklusjon, hvis nødvendig.

Metode 2, Feature Driven Development (FDD)

FDD ble utformet av Jeff De Luca i 1997 og først rapportert av Coad, P., Lefebvre, E. & De Luca, J. (1999). FDD er en metode som har fokus på å utvikle funksjoner som kan utvikles på under 2 uker. Metoden beskriver aktiviteter fra analyse til utvikling. FDD består av fem sekvensielle prosesser og inneholder roller, artefakter, mål og tidslinjer som trengts i et prosjekt.

Utvikle en overordnet modell

Man starter med å samle domain (domene/fagområde)- og utviklingsmedlemmene under veiledning av en erfaren objektmodellerer som innehar rollen "Chief Architect".

Det blir gjennomført en gjennomgang på toppnivå av omfanget til systemet og konteksten. Deretter blir det gjennomført en detaljert gjennomgang av hvert område, slik at det kan bli modellert. Etterpå bli små grupper bestående av fagpersoner og utviklere satt sammen. Gruppene tar så å utvikle sine egne modeller for å støtte opp om domenegjennomgangen. Gruppene presenterer så sine modeller for hver andre for gjennomgåelse og diskusjon. En av de foreslåtte modellene, eller en sammenslåing av modeller, blir så valgt av gruppen slik at modellen blir brukt på det enkelte domene/fagområdet. En sammenslåing av områdemodellen inn i en overordnet modell blir så utført.

Fordeler og ulemper i forhold til FIDO:
I analysefasen blir det nødvendig for FIDO å bruke flere av sine ansatte til å utføre selve analysen. Det positive er at man jobber side-ved-side med den som vet hva systemet skal brukes til. I denne fasen ville FIDO tidligere oppdaget behovet for et kontraktssystem. Ut i fra dette kunne man reforhandle fastprisen siden det ble avdekket et nytt behov. Ulempen blir at FIDO trenger en person med erfaring innen objektmodellering, og er avhengig av å ha folk fra ETC hos seg under denne fasen.

Lage en funksjonalitetsliste

En gruppe av hovedprogrammerer blir dannet for å bryte ned funksjonaliteten i domenene i mindre emneområder, og hver av dem skal igjen brytes ned i forretningsaktiviteter som igjen brytes ned i enda mindre deler, slik at man utformer en kategorisert funksjonalitetsliste. Kategoriseringen av funksjonalitetslisten blir utført i samarbeid med domeneekspertene.

Fordeler og ulemper i forhold til FIDO:
Ved at FIDO bruker sjefs programmererene til å gjøre om domenemodeller til funksjoner, får FIDO frem de ulike forretningsaktivitetene og hvilke funksjoner som trengts. På denne måte får man noe å jobbe ut i fra.
Ulempen blir at er avhengig av å ha folk fra ETC hos seg under denne fasen.

Planlegg etter funksjonalitet

Prosjektlederen, utviklingslederen og hovedprogrammererne planlegger rekkefølgen som funksjonaliteten skal implementeres, ut i fra funksjonalitetsavhengighet, arbeidskapasiteten til utviklingsgruppen og kompleksiteten til funksjonaliteten som skal implementeres. Planen som blir utformet kan oppdateres etter behov.

Fordeler og ulemper i forhold til FIDO:
FIDO vil få frem en plan for når ting skal bli ferdig og i hvilken rekkefølge. De får også frem ansvar for forretningsaktiviteter og liste over klasseeiere.
Ulempen er at man kan bomme på rekkefølgen man skal utvikle funksjonaliteten på.

Designe etter funksjonalitet

Hovedprogrammererne velger funksjonaliteter fra sin *innboks*. Han kan velge mer enn en funksjonalitet, som bruker de samme klassene. Hovedprogrammererne danner så en funksjonalitetsgruppe ved å identifisere eiere av klasser som mest sannsynlig vil bli brukt i utviklingen av funksjonaliteten(e) han har valgt ut til utvikling. Gruppen lager så "Sequence Diagram(s)" av funksjonaliteten. Hovedprogrammereren raffinerer objektmodellen basert på innholdet i "Sequence diagram(s)". Utviklerne skriver klasse og metode innledninger. Til slutt blir det holdt en design inspeksjon.

Fordeler og ulemper i forhold til FIDO:
I dette stadiet får FIDO kartlagt akkurat hva en funksjon skal gjøre med tanke på utviklingen. FIDO får også samlet de personene som er ansvarlige for de ulike klassene slik at man kan unngå implementeringsproblemer.

Utvikle etter funksjonalitet

Ved å se på designpakken, tar utviklingseieren av klassen å implementere de nødvendige tingene for at klassen deres kan støtte designet av funksjonaliteten. Koden som så er utviklet gjennomgår en enhetstest og kodeinspeksjon. Etter at koden har gjennomgått en suksessfull inspeksjon, blir koden sendt videre til hovedsystemet.

Fordeler og ulemper i forhold til FIDO:
For FIDO vil nok dette passe godt inn med tanke på at de i dag må utvikle det de trur er ferdig analyser og designet. Ved å bruke FDD vil FIDO mest sannsynlig kunne utvikle korrekte funksjoner, samtidig som kravene kan endres. En ekstra styrke er at FIDO etter hver iterasjon kan vise til ny funksjonalitet.
Et problem kan være at FIDO bommer på hvor lang tid det tar å utvikle funksjonen, og dermed kan ende opp med å bruke mer tid på prosjektet enn anslått.

Metode 3, SCRUM

Forløperen til Scrum er Sashimi som ble utviklet av japanerne som hadde de samme problemene med Fossefallsmodellen som alle ellers, nemlig at de ikke kunne gå tilbake til en tidligere fase, derfor tilpasset de Sashimi til å passe deres egen stil. Tidsbruk og fleksibilitet var like viktig som høy kvalitet og lav pris, derfor reduserte de antallet faser til fire - krav, design, prototyp og mottakelse uten å fjerne noen aktiviteter, dette ført til overlapping av de forskjellige fossefallsfasene. Andre utviklet så Sashimi videre, slo sammen de fire fasene til én, og kalte det Scrum.
Andre mener at Scrum fremdeles kan deles inn i flere faser og allikevel opprettholde fleksibiliteten, vi har derfor valgt å presentere Scrum i 3 faser.

Scrum prosess i tre faser:

Forspillsfasen (Pre-game Phase) som inkluderer to underfaser:

- *Planlegging (Planning)* inkluderer beskrivelsen av systemet som skal utvikles, prosjektgruppe, redskaper og andre ressurser. Et produkts backlog (gjenstående arbeid) inneholder alle kjente krav. Kravene er prioritert og backlog'en er bestandig oppdatert med nye krav og med mer nøyaktig vurderinger. For å kunne gå til neste iterasjon, må backlog'en sjekkes først.

  Planleggingen inkluderer bl.a. definisjon av prosjektgruppen, verktøy, risikovurdering osv.

- *Arkitektur/Høy nivå design (Architecture/High level design)* planlegger det høye jevne designet av systemet, samt arkitekturen.

Fordeler og ulemper i forhold til FIDO
Ved å bruke forspillfasen til Scrum, vil man vite nok av beskrivelsen til systemet som skal utvikles, til å kunne gjennomføre en første iterasjon, og også vite litt om den tid og kostnad man vil bruke for det.

Utviklingsfasen (Development Phase) er laget slik at den er smidig (agil), og denne smidigheten er litt av kjernen i Scrum. Her er uforutsette hendelser forventet, og man vet hvordan man skal takle det ved hjelp av såkalte Sprinter.

Sprinter er iterative sykluser. Hver Sprint inkluderer de tradisjonelle fasene av systemutvikling: krav, analyse, design, evolusjon og leveringsfaser, men har kun en kravspesifikasjon som gjelder den lille biten av systemet (for eksempel én funksjon) som skal utvikles akkurat nå.. En Sprint varer fra en uke til en måned. Det kan være et varierende antall personer som jobber på hver Sprint.

Scrums metodikk krever at hver person i gruppen forstår hele problemstillingen og alle trinnene ved utviklingen av systemet.

Under hver sprint holdes det daglige møter og man har et demonstrasjonsmøte på slutten av en sprint. Et demonstrasjonsmøte engasjerer kunden og tilfredsstiller utviklere ved at de har fått gjort noe på prosjektet.

Man løser altså en oppgave i hver Sprint, og når det ikke lenger er oppgaver igjen å løse, eller man mener at det ikke vil lønne seg å løse de siste, er utviklingen ferdig.

Fordeler og ulemper i forhold med FIDO
Kommunikasjon og interaksjon er et krav i Scrum prosessen. I FIDO har de personer som bruker hjemmekontor mye, kommunikasjonsmetodene bør derfor tilrettelegges nøye, og formålet og viktigheten av kommunikasjonen må forklares til alle som skal være en del av prosjektgruppen.

Etterspillfasen (Postgame Phase) inneholder oppgaver som testing, integrasjon av funksjonen i systemet og dokumentasjon.

Denne fasen går man inn i når et av kravene er fullført.

Fordeler og ulemper i forhold med FIDO
Denne fasen øker sannsynligheten for at resultatet blir godkjent av kunden, siden kunden var så mye involvert i utviklingen av prosjektet.

Konklusjon

Det største problemet for FIDO er at kompleksiteten i systemet hos ETC ikke kan avdekkes med en gang, slik at det hele tiden kommer endringer underveis.
Det finnes flere andre utfordringer også, men denne ene, det at det skjer stadige endringer underveis, er nok til at vi ikke bør velge en strukturert metode som Stradis, som er basert på fossefalls-tankegangen, og dermed gjør det til en forutsetning at man kan klare å avslutte f.eks design-fasen før man går videre til programmeringsfasen.
Både Scrum og FDD takler endringer underveis, så hvilken velger vi da?
Vi har sammenliknet alle 3 metodene i tabellen under, i forhold til om de kanskje kunne hjelpe FIDO med ETC-prosjektet, og vi ser at FDD og Scrum kommer nokså likt ut.
Erfaringene sier oss at det er heller usannsynlig at vi kommer til å følge én metode fullt ut. Vi må derfor prøve å velge den som gir oss det beste grunnlaget til å tilpasse en metode til FIDO på, og siden FIDO ikke har brukt noen metoder før, ser vi på det som viktig at metoden som velges i første omgang, er lett og sette seg inn i og få en viss grad av oversikt over.
Siden ingen i gruppen hos oss har førstehåndserfaring med noen av de to metodene, har vi prøvd å sette oss i FIDOS sted, for å se hvilken metode som virker mest tiltrekkende. Valget faller på Scrum. Vi har allerede i tabellen vist at den kanskje kan dekke de fleste utfordringene, og vi synes informasjon om Scrum har vært lettere tilgjengelig enn informasjon om FDD.
I tillegg mener vi at Scrum også egner seg som metodegrunnlag i en organisasjon som FIDO, som tar sikte på å arbeide med flere prosjekter av ETC-typen senere, og kanskje også større prosjekter.

Kategorier	FIDO problemer	Metoder		
		STRADIS	FDD	SCRUM
Endringer	Kompleksiteten avdekkes et stykke ut i løpet	Nei	Ja	Ja
	Avdekket etter hvert behov for enda et system	Nei	Ja	Ja
	Det ekstra system økter også i kompleksiteten etter hvert	Nei	Ja	Ja
	A&D-fasen strekker seg langt inn i programmeringsfasen	Nei	Ja	Ja
	ETC får ikke noe skikkelig estimat for tidsbruk og kostnadsbruk	Ja/Nei	Ja	Ja
	Programmererne konsentrerer seg om de deler av kravene som virker mest stabile, uavhengig av hva som er viktigst å få til først, eller viktigst for kunden	Nei	Ja	Ja
Kommunikasjon	Terminaldivisjonen, representert ved Mike, har sine tvil og føler at de ikke får dekket behovene sine	Nei	Ja	Ja
	Kommunikasjonen mellom A&D-teamet og programmeringsteamet er muligens ikke god nok	Ja/Nei	Ja	Ja
	FIDO bruker egne folk og i tillegg innleide som flytter til Norge, og innleide som bor i Norge allerede	Ja	Ja	Ja
	Stig er bare på kontoret et par ganger i uka, jobber helst hjemmefra	Ja	Nei	Nei
	Brukerne sitter svært spredt	Ja	Nei	Nei
Pris	Time-for-time pris i A&D-fasen	Ja	Ja	Ja
	Fastpris for programmeringsfasen	Ja	Tja	Nei
Andre	FIDO mangler helt en metode for systemutvikling	Ja	Ja	Ja
	Mike liker ikke Ivan og vi vet at det er merkbart i organisasjonen	Nei	Tja	Tja
	FIDO mangler en prosjektleder som har ansvaret for HELE prosjektet, roller/ansvar?	Ja/Nei	Ja	Ja
	Janis har roller i begge prosjektorganisasjonene, behov for klargjøring av roller/ansvar?	Nei	Ja	Ja

Tabell 2: Metode sammenligning