ESTIMERING AV WEBUTVIKLINGSPROSJEKTER

Use case poeng sammenliknet med eksperters estimater, funksjonspoeng, COCOMO II og WEBMO

MASTEROPPGAVE

skrevet av
Kristine Sæhlie

26. april 2005
Simula Research Laboratory
&
Institutt for Informatikk
Universitetet i Oslo

SAMMENDRAG

Anbudsprosessen for konsulentoppdrag er viktig for programvareleverandører. Det er her en kunde avgjør hvilken leverandør som velges til utvikling av planlagt programvare. Estimering av pris og utviklingstid vil være en viktig del av leverandørenes tilbud. Det er opp til leverandøren å lage så gode og så realistiske estimater som mulig, slik at den blir valgt. Ved valg av leverandør studerer kunden estimatene. Kundens ønske er å få best mulig system til lavest mulig pris og innen kortest mulig tid. Dette kan føre til at for å bli valgt må leverandøren presse ned priser og tid urimelig, vel vitende om at estimatet blir overskredet. Det vil være nyttig å studere om en estimeringsmodell til bruk for kunde kan være med å hjelpe kunden til å forstå programmets størrelse og tid som må brukes til utvikling, samt vurdere hvilken estimeringsmodell dette kan være. Estimater er også viktig for at utviklingsprosjekter i det hele tatt settes i gang. En kunde må vite omtrentlig utviklingstid og kostnader, for å se om den faktisk har råd til å få systemet utviklet. For leverandøren er estimater viktig slik at inntekter kan beregnes samtidig som den har oversikt over hvor lenge ansatte er opptatt ved ulike prosjekter. Ofte baseres estimater på utviklernes egne erfaringer. En utfordring ved estimater i anbudsprosessen er at de må lages tidlig, allerede før detaljert kunnskap om systemet som skal utvikles foreligger. Det er derfor ekstra viktig å kunne estimere riktig ut fra den funksjonalitet som kravspesifikasjonen fra kunden beskriver. Dette gjør det nyttig å studere hvordan krav-spesifikasjonen, spesielt de funksjonelle kravene, kan brukes til estimering.

I mai 2003 startet forskere i avdeling for Software Engineering ved Simula Research Laboratory et forskningsprosjekt på utvikling av webbasert programvare. Flere leverandører utviklet en forskningsdatabase i parallell samtidig som forskere ved avdeling for Software Engineering studerte utviklingen. Denne oppgaven studerer estimeringen av utviklingsprosjektet. Hovedfokus er å studere hvordan en estimeringsmodell basert på use case kan støtte både kunder og utviklere i anbudsprosessen. Use case poeng metoden tar utgangspunkt i at et systems funksjonelle krav beskrives i use case. Det er stor interesse for metoden i industrien, men det er nødvendig med flere studier for å tilpasse den til forskjellige typer prosjekter. Denne oppgaven studerer hvordan en metode for use case basert estimering kan brukes til å estimere webapplikasjoner ved å prøve den ut på prosjektene og sammenlikne den med mer etablerte metoder. De utvalgte metodene er funksjonspoeng, COCOMO II og WEBMO. De viktigste aspektene i studiet er; hvilken kunnskap om systemet som er nødvendig for å estimere med metodene, hvordan metodene estimerer i forhold til utviklernes egne estimater og faktisk brukt tid, hvilken kvalitet metodene planlegger samt når og hvem som kan bruke metodene.

Denne studien viser at målet på størrelse i use case poeng metoden er enklere å benytte for personer uten teknisk kompetanse, som for eksempel kunder i

software prosjekter, enn funksjonspoeng. Dette forutsetter godt strukturerte use case, og denne oppgaven gir en del retningslinjer for strukturering når use casene i utgangspunktet er meget detaljerte. Studien viser videre at kostnadsdrivere knyttet til kodekvalitet som vedlikeholdbarhet, gjenbruk, dokumentasjon, kompleksitet og erfaring påvirker innsats i stor grad. Studien ser på hvordan slike kostnadsdrivere kan settes i webprosjekter. I dette tilfellet estimerte metodene like godt eller bedre enn utviklerne selv, noe som viser at use case poeng metoden kan støtte både kunder og utviklere i en anbudsprosess der en use case modell som beskriver de funksjonelle kravene til systemet foreligger.

FORORD

Dette dokumentet utgjør masteroppgaven min ved Institutt for Informatikk, Universitetet i Oslo.

Først vil jeg takke min veileder Bente Anda for god veiledning og innspill ved gjennomføring av oppgaven. Hun har gitt meg tilgang til alle nødvendige dokumenter og informasjon. Videre vil jeg takke familie og venner for all støtte og interesse i arbeidet. Spesielt takk til Ida, Solvor, Toril og far som har lest igjennom og kommentert deler av oppgaven.

Oslo, 26. april 2005

Kristine Sæhlie

INNHOLD

TABELLER

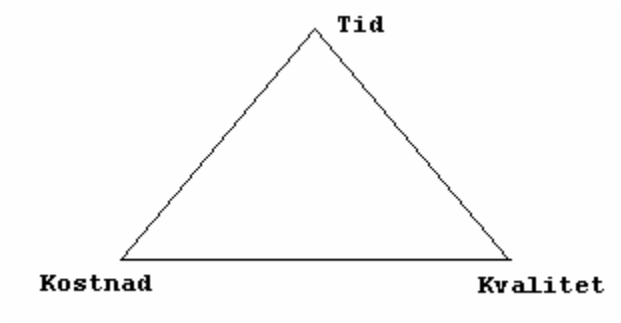
FIGURER

1 INTRODUKSJON

Å estimere betyr å vurdere fremtiden [Gundersen 1996]. Det er vanskelig å forutse hendelser. I mange tilfeller er det umulig. Programvareutviklere er ofte nødt til å estimere tid, kostnader og funksjonalitet uten detaljert kunnskap om prosjektet som skal estimeres. Dette fører til stor usikkerhet knyttet til estimatene. Programvareindustrien er kjent for store overskridelser av tids- og kostnadsbudsjetter [Boehm et. al 2000c].

1.1 Programvareutvikling og estimering

Programvareutvikling er arbeidet med å spesifisere og utvikle programvare som skal møte et kundebehov. Typiske kundekrav er at programvaren skal leveres i tide, til budsjettert kostnad og av god kvalitet. Hvorvidt disse kriteriene er oppfylt, avgjør om utviklingsarbeidet har vært en suksess [Hjertø 2002]. Kostnad, kvalitet og tid er tre faktorer som påvirker hverandre som ytterkantene i en trekant, se Figur 1-1. Jo høyere kvalitetskrav, desto høyere kostnader og lengre tid vil utviklingsarbeid ta.



Figur 1-1 Kostnad, kvalitet og tid

Prosjektleder for et utviklingsprosjekt er nødt til å vite omfang, kostnad og funksjonalitet allerede i startfasen og må estimere så riktig som mulig. Tid og kostnader i programvareutvikling avhenger av mange faktorer. Blant annet menneskelig kunnskap, antall nøkkelpersoners tid til utvikling, bruk av kjent/ukjent teknologi eller kunders stabilitet i krav er alle usikre faktorer. Allikevel må prosjektleder finne så gode og realistiske estimater som mulig for å kunne gi tilbud på kontrakter, eller vurdere gjennomførbarhet i henhold til kostnytteanalyser [Hjertø 2002]. Dette gjør estimering til noe av det vanskeligste innen programvareutvikling, men også det viktigste [Hansen 2002].

1.1.1 Måling

Måling er utgangspunktet for å skaffe seg kunnskap. Hansen (2002; s. 2) definerer måling som; *"… prosessen med å knytte symboler, vanligvis tall, til en egenskap vi tilegner en gjenstand"*. I programvareutvikling kan det være nødvendig med blant annet følgende målinger [Hansen 2002];

- Programvarens størrelse
- Programvarens vedlikeholdsvennlighet

- Antall feil i programvare
- Tid for utvikling
- Kostnader i utvikling
- Antall personer nødvendig til utvikling
- Tid for testing
- Kostnader for testing

For å forstå målinger knyttes de til forskjellige skalaer. Størrelsen på et program måles etter absolutt skala. Å telle antall kildekodelinjer skal kun resultere i et tall, når reglene for definisjon av en kildekodelinje er spesifisert [Hansen 2002].

Grunnlaget for estimering er ulik type måling. Målingen utgjør en fundamental størrelse som er basis ved bruk av estimeringsmodeller. Dette studiet tar for seg to typer målinger. Den ene typen er estimering av arbeidsinnsats målt i antall timer. Den andre typen er estimering av programvarens størrelse målt i antall kildekodelinjer og såkalt use case poeng, funksjonspoeng og webobjekt. Dette beskrives mer senere.

1.1.2 Estimeringsmodeller

Ved hjelp av gode historiske data, god faglig innsikt og erfaring skal det være mulig å lage gode estimater [Hansen 2002]. Eksperters[1] egne estimater er mest brukt. Dersom de er basert på mye kunnskap så er det antageligvis det beste man har. I noen tilfeller kan nødvendig kunnskap mangle. Da vil estimatene preges av ekspertens erfaring og gjetning. En ulempe med eksperters estimater er at de ikke kan testes før prosjektet er gjennomkjørt. Det er også vanskelig å etterprøve eksperters estimater. Mange års erfaring gir høy kompetanse. Allikevel finnes ingen garantier for at en ekspert alltid ”gjetter” riktig. I denne typen situasjon kan estimering med estimeringsmodeller være til hjelp. Dersom estimeringsmodellenes og ekspertens estimater avviker betraktelig, kan det være et tegn på at eksperten bør se igjennom estimatene på nytt [Boehm et al. 2000a]. En estimeringsmodell baseres på statistisk utregning av erfaringsdata. Typisk for mange estimeringsmodeller er problemer med verdisetting. Mange av modellene krever mye kunnskap og kan derfor være vanskelige å bruke [Boehm et al. 2000a]. Generelt er estimeringsmodeller et omstridt tema og ingen modell er kåret til alltid å være den beste [Hansen 2002].

Estimering kan gjøres nedenfra og opp eller ovenfra og ned. Nedenfra og opp estimering begynner med en liste av alle elementene på laveste nivå. Til dette kan både eksperters erfaring og flere metoder benyttes. Hvert element tilegnes tidsforbruk. Tidsforbruket for elementene summeres. Summen utgjør det totale estimatet. Ovenfra og ned estimering begynner enten med en forhåndsbestemt kostnad på produktet eller hele produktet i en helhet for så å dele det opp i mindre biter. Estimatene for hvert element kalkuleres ut fra den totale kostnaden. Ovenfra og ned benyttes oftest i veldig tidlige faser. Nedenfra og opp er mer tidkrevende, men gir ofte nøyaktigere estimater [Boehm er al. 2000a]

Metoder som baseres på matematisk utregning kalles modellbaserte metoder. Herunder hører metoder som COCOMO [Boehm et al. 2000c], Use Case Poeng

[1] Når det i dette studiet refereres til eksperter menes prosjektleder eller programvareutvikler med noen års erfaring innen programvareutvikling. En nyutdannet regnes ikke som en ekspert.

[Karner 1993] og WEBMO [Reifer 2002]. Metodene baserer seg på livssyklusmodeller og inneholder en algoritme som henter data fra tidligere prosjekter. Modellbaserte estimeringsmetoder er de mest utbredte metodene. Sammen med modellbaserte estimeringsmetoder brukes regresjonsbaserte estimeringsmetoder. Regresjonsbaserte teknikker er den mest populære måten å bygge modeller. Blant annet benytter COCOMO II regresjonsbaserte teknikker [Boehm et al. 2000a].

Estimeringsmodellene har både fordeler og ulemper. Å trekke ut de beste egenskapene fra to eller flere modeller, for så å kombinere disse, gir sammensatte estimeringsmetoder. Dette studiet vil dreie seg om modellbaserte estimeringsmetoder sammenliknet med eksperters estimering.

1.1.3 Webutvikling visa tradisjonell systemutvikling

Den raske forandringen innen programvareutvikling har gjort det vanskelig å utvikle gode estimeringsmodeller som gir nøyaktige estimater i alle domener. De siste årene har gitt en kraftig økning i utvikling av webbaserte systemer. Bedrifter satser mer på webbasert intranett og kommunikasjon over Internet [Boehm et al. 2000a].

Tradisjonell systemutvikling har følgende særtrekk [Reifer 2000];

- Oftest tre ulike roller i utvikling der utvikleren spesialiserte seg i rollen og samarbeidet lite med de andre rollene
- Klassisk faseinndelt utvikling basert på bruker- og funksjonskrav
- Inkrementell levering i tillegg til hyppig bruk av use case og en dokumentdrevet prosess
- Objektorienterte metoder med hjelp av genereringsverktøy og programmeringsspråk som C++
- Kvalitetsstyringssystem basert på tyngre systemer som CMM [Paul et al. 1995]
- Størrelsen på et typisk tradisjonelt utviklingsprosjekt er medium til stor med opp til flere hundre deltakere.
- Den typiske tidslinjen ligger mellom 12 og 18 måneder mens kostnadene er i millionklassen

Webutvikling skiller seg fra tradisjonell utvikling ved følgende særtrekk [Reifer 2000];

- Krav til et tettere samarbeid der rollene ofte veksler mellom prosjektdeltakerne [Roetzheim 2000]
- Agile metoder og prosesser som eXtreme Programming (XP) eller Rational Unified Process [Krutchen 2003]
- Ofte bruk av 4. og 5. generasjonsspråk som Java og html med komponentbaserte metoder
- Kvalitetsstyringen ofte mer ad hoc
- Gjennomføres ofte av mindre prosjektgrupper med 5 til 30 prosjektdeltakere, generelt er prosjektene blitt mindre
- Hovedmålet er å få produkter av god kvalitet ut på markedet så raskt som mulig
- Tidslinjen er typisk 3 til 6 måneder mens kostnadene ligger på tusener opp til noen få millioner

1.2 Mål med studiet

Det overordnede målet med denne oppgaven er å samle kunnskap om estimeringsmetoden use case poeng [Karner 1993]. Det viktigste perspektivet er om, for hvem og eventuelt når metoden er egnet for bruk i estimering av web-utviklingsprosjekter. Mer detaljert studeres størrelsesmål, kostnadsdrivere og estimater. De krav use case poeng metoden setter til kunnskap om system og omgivelser, sammenliknes med eksperters egne estimater, funksjons-poenganalyse [Garmus og Herron 2004], COCOMO II [Boehm et al. 2000c] og WEBMO [Reifer 2002]. Totalt skal dette gi en indikasjon på anvendeligheten for modellene i webutvikling.

Use case poeng er en estimeringsmodell basert på et programs funksjonelle krav, beskrevet i use case. Metoden ble utviklet av Karner (1993), og det er stor interesse for den. Den er beskrevet mange steder, det forskes på den, det finnes kurs i den og det finnes flere verktøy tilpasset den. Allikevel er forskning på bruk av metoden nødvendig, for å kunne tilpasse den alle typer av dagens systemer. Dette studiet bidrar med forskning på bruk av use case poeng metoden i enkle webapplikasjoner. Hovedsakelig ble use case poeng metoden utviklet som et tillegg til funksjonspoeng og COCOMO II. Dette studiet diskuterer muligheten ved å kombinere eller hente tips fra funksjonspoeng og COCOMO II, slik at alle de viktigste områdene innenfor programvareutvikling også dekkes i use case poeng metoden.

Ofte brukes use case som input til estimering uten bruk av use case poeng metoden. Dette studiet tar utgangspunkt i estimering av fire webutviklings-prosjekter. Disse ble estimert ut fra systemets funksjonelle krav formet i use case, men uten bruk av use case poeng metoden. Oppgaven undersøker om ekspertene ville hatt nytte av å bruke use case poeng metoden som støtte i estimeringsarbeidet.

Meningen med use case poeng metoden, COCOMO II og WEBMO er at det skal kunne estimeres tidlig i utviklingsarbeidet. Det er imidlertid funnet at COCOMO II og WEBMO krever noe mer detaljkunnskap enn use case poeng metoden. Dette gjør use case poeng metoden enklere å bruke ved tidlig estimering og også nyttig for bruk av kunder. Generelt antar modellene høy kvalitet i arbeidet, s

1.3 Metode

Dette studiet er en multippel case studie. Utgangspunktet for forskningen er et prosjekt der fire ulike leverandører har utviklet samme system i parallell. Prosjektet kalles *The Simula Database of Empirical Studies* (DES). Datainnsamlingen er allerede gjennomført. Dette gjør studiet til en tekstlig analyse av prosjektdokumentasjon. Studiet baseres på følgende kilder;

- Kravspesifikasjon
- Leverandørenes tilbud
- Intervjutranskripsjoner
- Regneark med antall timer brukt
- Relatert litteratur

Studiet er et bidrag til videre forskning på egnetheten til use case poeng metoden ved utvikling av webapplikasjoner. I tillegg kartlegges kunnskapskrav i funksjonspoeng, COCOMO II og WEBMO ved utvikling av webapplikasjoner.

1.4 Forskningsspørsmål

En studie av use case poeng metodens estimater i forhold til eksperters estimater ved webutvikling. Hvor egnet er use case poeng metoden ved bruk til estimering av webbaserte programmer?
Studerer fire leverandørers utvikling av samme system og sammenlikner metodene med faktisk brukt tid og metodenes krav til kunnskap om systemet.

1.4.1 Underpunkter

Sammenlikning av informasjonskrav for størrelsesmålene use case poeng, funksjonspoeng og webobjekter samt use case poeng metodens og COCOMO II's kostnadsdrivere.
Studere estimeringsmodellenes krav til kunnskap om systemet som er under utvikling når det gjelder størrelsesmål og kostnadsdrivere.

Studie av tekniske faktorer i use case poeng metoden med tanke på webbaserte programmer.
Studere leverandørenes kunnskap om use case metodens tekniske faktorer i systemet som er under utvikling samt de tekniske faktorenes relevans i webutvikling.

Sammenlikning av kvaliteten på leverandørenes leverte og godkjente webapplikasjon i forhold estimeringsmodellenes estimater.
Studere leverandørenes tanker og krav til viktige aspekter i utvikling, utviklingsmodell og kvalitet. Samt studie av hvor godt metodene gir rom for god kvalitet og stødig utviklingsmodell.

1.5 Avgrensninger

En grundig sammenlikning av de fire estimeringsmetodene use case poeng, funksjonspoeng, COCOMO II og WEBMO er mye arbeid. Siden dette er en masteroppgave der tidsperspektivet er en viktig faktor er følgende avgrensinger gjennomført;

- Studiet er avgrenset til i hovedsak å gjelde sammenlikning av størrelsesmål, kostnadsdrivere og estimater.
- For størrelsesfaktorer sammenliknes kun use case poeng og funksjonspoeng i detalj. Webobjekt diskuteres generelt til slutt. Funksjonspoeng er valgt siden det utgjør grunnlaget for et systems størrelse i COCOMO II.
- For kostnadsdrivere sammenliknes use case poeng metoden og COCOMO II.
- Mange aspekter påvirker utviklingstid. I dette studiet er det lagt vekt på kvalitet i utvikling, kvalitet i kodeoppbygging og struktur samt kvalitet i estimering og gjennomføring.

1.6 Oppgavens struktur

Kapittel 2 beskriver webutvikling i forhold til tradisjonell systemutvikling. I tillegg introduseres case'et i denne oppgaven.

Kapittel 3 introduserer estimeringsmodellene som brukes i dette i studiet. Først en liten historisk introduksjon, deretter estimering generelt, funksjonspoeng analyse, COCOMO II og WEBMO.

Kapittel 4 beskriver use case modellering. I tillegg introduseres use case poeng metoden. Til slutt presenteres relatert arbeid.

Kapittel 5 beskriver metoden som brukes til gjennomføring av dette case studiet.

Kapittel 6 presenterer resultatene ved estimering av DES med de ulike estimeringsmodellene samt ekspertenes egne estimater.

Kapittel 7 diskuterer resultatene fra kapittel 6 i følgende rekkefølge; størrelses-faktorer, kostnadsdrivere, estimeringsmodellene og kvalitetsaspekter.

Kapittel 8 evaluerer forskningsmetoden og resultatene funnet i dette studiet. Til slutt gis konklusjon og forslag til videre arbeid.

Kapittel 9 forklarer ord og forkortelser i form av en ordliste.

Vedlegg presenterer tabeller og utvidet informasjon om estimeringsmodellene.

2 PRESENTASJON AV CASE

Våren 2003 satte avdeling for Software Engineering (SE) ved Simula Research Laboratory AS (SRL) i gang et forskningsprosjekt der forskningsområdet var utviklingen av en webbasert forskningsdatabase. Tidligere ble all forskningsdata ved SE lagret i form av en statisk html tabell. Ulempen med denne var at administrator måtte kjenne html samtidig som tabellen ikke var særlig brukervennlig. SE ønsket å utvide funksjonaliteten i html tabellen. Dette skulle gjøres ved å implementere et nytt, brukervennlig og dynamisk databasesystem døpt "The Simula Database of Empirical Studies" (DES). SE utviklet en kravspesifikasjon for DES med beskrivelse av funksjonelle krav. De viktigste funksjonene er lagring, endring, sletting, gjenfinning og rapportering av forskningsdata. DES støtter tre brukergrupper; databaseadministrator, studieadministrator og gjest. En komplett oversikt over funksjonelle krav er lagt i Vedlegg A. DES følger de grafiske prinsippene som ligger til grunn i SE's eksisterende websider (Simulaweb). Hovedmålet med DES er å utvikle en effektiv støtte for vedlikehold og rapportering av studier og publikasjoner ved SE.

Utviklingen av DES ga forskerne ved SE mulighet til å studere alt fra anbudsprosess til oppstart, gjennomføring og avslutning av et utviklingsprosjekt. Norske og internasjonale programvareutviklingsbedrifter ble invitert til å gi tilbud på DES. I juni 2003 var tilbudene klare. De inneholdt bedriftenes forståelse av problemstillingen med løsningsskisser, tids- og kostnadsestimater. 35 bedrifter lokalisert i Norge, ga hvert sitt tilbud på samme kravspesifikasjon. Jørgensen og Carelius (2004) beskriver og studerer anbudsprosessen. De har funnet at typen anbudsprosess har stor innflytelse på bud. Lav kunnskap om systemet som skal lages fører til generelt høyere bud enn ved høy kunnskap om systemet samt stabilitet i krav.

SE valgte fire ulike leverandører til å utvikle hvert sitt eksemplar av DES. Leverandørene refereres til som A, B, C og D. Tabell 2-1 viser de fire leverandørenes pris- og timeestimater. Leverandørene ble valgt med en bevist variasjon i type leverandør, pris- og timeestimat. Utviklingen av DES foregikk parallelt høsten 2003.

Leverandør	Pris i NOK	Innsats i timer
A	160 000	250
B	306 900	341
C	70 000	100
D	552 500	650

Tabell 2-1 Leverandørenes pris- og timeestimat

Gjennomgang av tilbudene til de 35 bedriftene ga store forskjeller i pris og estimat. Dette motiverte en studie av hvorvidt en estimeringsmodell kunne gi støtte til valg av leverandør med realistisk pris. Use case poeng metoden ble valgt fordi den krever liten teknisk innsikt. COCOMO II og funksjonspoeng ble valgt på grunn av at de er godt kjente modeller.

3 MODELLER FOR TID OG KOSTNADSESTIMERING

Dette kapittelet beskriver estimeringsmodeller. Kapittelet innledes kort med en presentasjon av første forskning på estimeringsmodeller. Videre forklares en generell metode for estimering. De siste avsnittene beskriver tre estimeringsmodeller. Disse modellene blir brukt til å estimere DES; funksjons-poenganalyse, COCOMO II og WEBMO. Modellene er valgt fordi de egnet seg til å gi tidlige estimater samtidig som funksjonspoenganalyse og COCOMO II er kjente modeller. Dermed vil det være interessant å studere disse mot use case poeng som er noe mindre kjent. Use case poeng metoden er basert på funksjonspoenganalyse. Det er viktig å kjenne til denne metoden for å forstå use case poeng [Ribu 2001]. Use case og use case poeng er detaljert beskrevet i kapittel 4. WEBMO ble valgt fordi den er eneste modell direkte rettet mot estimering av webapplikasjoner.

3.1 Estimeringsmodellenes grunnstein

Forskning på kostnadsestimering startet i 1965 med en studie av attributter ved 169 programvareutviklingsporsjekter i USA. Forskningen førte til en delvis utvikling av kostnadsmodeller. Disse ble benyttet mot slutten av 1960 og begynnelsen av 1970-tallet. Forskerne som fortsatte studiene utover 1970-tallet støtte alle på samme problem; ettersom programmene vokste i størrelse, vokste også kompleksiteten. Dette gjorde det vanskelig å forutse nøyaktig kostnad for programvareutviklingen. Forskerne identifiserte etter hvert grunnsteinene for systemutviklingens kostnadsmodeller. Mot slutten av 1970 årene ble flere robuste modeller produsert, blant annet COCOMO og funksjonspoenganalyse [Boehm et al. 2000a].

3.2 Generell estimeringsmetode

Valacich et al. (2001) beskriver, blant annet, en generell fremgangsmåte for estimering. Noen av leverandørene i dette studiet bruker en tilpasset versjon av denne framgangsmåten. Den baserer seg hovedsakelig på eksperters estimater.

$$\text{Forventet timeantall} = \frac{\text{Optimistisk timeantall} + 4 * \text{Realistisk timeantall} + \text{Pessimistisk timeantall}}{6}$$

Tabell 3-1 Formel for forventet timeantall [Valacich et al. 2001]

Først identifiseres delaktiviteter i utviklingsprosjektet som skal gjennomføres. Totalt benyttes tre estimater for å finne antall timer; pessimistisk, realistisk og optimistisk timeantall. Optimistisk og pessimistisk timeantall reflekterer henholdsvis minimum og maksimum antall timer nødvendig for å fullføre en aktivitet. Realistisk timeantall reflekterer ekspertens beste gjetting. Til slutt beregnes forventet timeantall for hver delaktivitet med formelen i Tabell 3-1. Ved

å summere timeantallet for hver delaktivitet beregnes totalt forventet timeantall [Valacich et al. 2001].

Når forventet timeantall er beregnet for samtlige delaktiviteter bestemmes aktivitetenes sekvens. Disse føres inn i en prosjektplan. Til slutt gjennomføres en risikovurdering der kritiske stier gjennom prosjektplanen identifiseres. Den lengste tid forskjellige aktiviteter kan bli forsinket uten at det går ut over prosjektet beregnes også [Valacich et al. 2001]. Generelt forholder kunder seg helst til et konkret estimat. Allikevel kan best og verst tenkelig timeantall være et sikrere estimat enn ett enkelt tall ved tidlig estimering [Boehm et al. 2000b].

3.3 Funksjons Poeng Analyse

I løpet av 1970-årene tok IBM's Alan Albrecht tak i problemet med å måle et systems størrelse. Han utviklet teknikken funksjonspoenganalyse (FPA). Utover 1980–tallet ble FPA tatt i bruk av mange bedrifter. FPA er første metode for å måle størrelse av programmer, uavhengig av teknologi og programmeringsspråk. Med FPA kan programvareutvikling måles på tvers av prosjekter og bedrifter. FPA er basert på teorien om at funksjonene i et programvaresystem er det beste målet på størrelsen av et system [Longstreet 2003, Garmus og Herron 2004].

3.3.1 Beregning av funksjonspoeng

Et av målene med FPA er at tid og kostnader skal kunne estimeres tidlig i programvareutvikling. Med utgangspunkt i brukerkrav brytes et system opp i mindre funksjoner. Hver funksjon kategoriseres til en av fem mulige kategorier. De to første kategoriene grupperer data som logisk informasjon. Disse er [Longstreet 2004a];

Intern logisk fil (ILF) er en brukeridentifisert gruppe av logisk relaterte data. Dette kan være en del av en database eller et filsystem. ILF vedlikeholdes gjennom ekstern inndata.

Ekstern grensesnitt fil (EIF – External Interface File) tilsvarer en annen applikasjons interne logiske fil. Dataene brukes til lesing eller referering og lagres utenfor systemets grense. En EIF vedlikeholdes av en annen applikasjons eksterne inndata.

De tre siste kategoriene tilsvarer funksjoner som opererer mot interne logiske filer eller eksterne grensesnitt filer. Disse er [Longstreet 2004a];

Ekstern inndata (EI) er en prosess der data krysser grensen fra utsiden til innsiden av programmet. Data kommer fra en aktør. En aktør er en utenforstående som kommuniserer med systemet og har mulighet til å legge til, endre, slette eller hente informasjon fra en intern logisk fil. Dette gjelder både kontrollinformasjon og forretningsinformasjon.

Ekstern utdata (EU) er en prosess der data krysser grensen fra innsiden til utsiden av programmet. Dataene blir sent fra systemet og til en aktør. Utdata refererer blant annet til rapporter, skjermbilder og feilmeldinger. Utdata lages av informasjon som kommer fra en eller flere interne logiske filer eller eksterne grensesnittfiler.

Ekstern spørring (ES) er en prosess med både inndata- og utdatakomponenter. En aktør spør etter data og får et umiddelbart svar fra programmet i form av utdata. Inndataprosessen opptaterer ikke interne logiske filer.

Funksjonene i hver kategori vurderes i henhold til kompleksitet; lav, gjennomsnittelig eller høy. Kompleksiteten avgjøres ved hjelp av forhåndsdefinerte kriterier [Garmus og Herron 2004]. Antall funksjoner vurdert for hver kompleksitet i hver kategori multipliseres med en fastsatt faktor, se Tabell 3-2. Ved å summere produktene beregnes programvarens størrelse i ujusterte funksjonspoeng (UFP). Alle formler for FPA er vist i Tabell 3-3.

Kategori	Lav	Gj.snitt	Høy
# Ekstern input	3	4	6
# Ekstern output	4	5	7
# Ekstern spørring	3	4	6
# Intern logisk fil	7	10	15
# Ekstern grensesnittfil	5	7	10

Tabell 3-2 Vektlegging av komponenter [Garmus og Herron 2004]

UFP justeres ved å multiplisere inn en verdijusteringsfaktor (VAF). Den viktigste delen av formelen for å finne VAF består av vurderingen av 14 kostnadsdrivere. En *kostnadsdriver* er en konstant som påvirker antall funksjonspoeng både i positiv og negativ retning. Vurderingen av kostnadsdriverne varierer fra prosjekt til prosjekt. Tabell 3-4 forklarer kostnadsdriverne i FPA. Hver kostnadsdriver vurderes ut fra den betydning de har for utviklingen. Skalaen går fra 0 – 5 [Garmus og Herron 2004];

- 0 betyr ingen innflytelse,
- 1 betyr tilfeldig innflytelse,
- 2 betyr moderat innflytelse,
- 3 betyr gjennomsnittlig innflytelse,
- 4 betyr betydelig innflytelse og
- 5 betyr essensiell innflytelse.

Vurderingene av kostnadsdriverne summeres. I formelen for VAF, se Tabell 3-3, utgjør F_j kostnadsdriverne. Ved estimering av DES benyttes UFP som utgangspunkt for COCOMO II, se avsnitt 3.6.

$$UFP = \sum_{i=1}^{5} (\text{Antall komponenter}_i * \text{Vurdering}_i)$$

$$VAF = (0{,}065 + 0{,}01 * \sum_{j=1}^{14} F_j)$$

$$FP = UFP * FP$$

Tabell 3-3 Formler i FPA [Garmus og Herron 2004]

Kostnadsdrivere		Beskrivelse - Spørsmål
1	Operasjonell letthet	I hvilken grad krever applikasjonen effektive og stabile oppstarts-, sikkerhetskopierings- og gjenopprettelsesprosedyrer?
2	Datakommunikasjon	Hvor mange kommunikasjonsmuligheter finnes til hjelp i overføring/utveksling av informasjon med applikasjonen?
3	Distribuert prosessering	I hvilken grad påvirker håndteringen av distribuerte data og dataprosessering utviklingen?
4	Ytelse	I hvilken grad krever bruker spesifikk responstid eller gjennomstrømmingshastighet?
5	Konfigurasjon	I hvilken grad skal applikasjonen kjøre i et eksisterende tungt utnyttet miljø/plattform?
6	Adgang til online data	Hvor mye av informasjonen hentes online?
7	Sluttbrukereffektivitet	Er applikasjonen designet for sluttbruker-effektivitet der online datainntasting krever at interaksjonen bygges over skjermer eller operasjoner?
8	Online oppdatering	Blir alle masterfiler/logiske filer oppdatert online?
9	Transaksjonshyppighet	Hvor hyppig utføres transaksjoner? Daglig, ukentlig, månedlig etc.
10	Kompleks prosessering	Krever applikasjonen en utpreget kompleks prosessering?
11	Gjenbrukbarhet	Er koden designet for å være gjenbrukbar?
12	Installasjon	Er konvertering og installasjon inkludert i design?
13	Multiple sider	Er applikasjonen spesielt designet, utviklet og støttet for å bli installert for multiple sider i multiple organisasjoner?
14	Brukerendring	Er applikasjonen designet slik at den lett kan endres av brukeren?

Tabell 3-4 14 kostnadsdrivere i FPA [Garmus og Herron 2004, Longstreet 2004a]

3.3.2 FPA og objektorientering

FPA er i første omgang utviklet for dataprosesseringsprogrammer. Dermed er den opprinnelige beregningen av FP ikke kompatibel med objekt orientering, use case og grafisk brukergrensesnitt. Med tiden er de opprinnelige formlene justert og forbedret mange ganger, og kan nå brukes i objektorientert utvikling. Longstreet (2003) presenterer en teknikk for beregning av funksjonspoeng ved bruk av use case. Denne teknikken beskrives mer utfyllende av Garmus og Herron (2004) og er brukt ved estimering av DES.

3.3.3 Timefaktor

For å estimere det timeantall et utviklingsprosjekt krever kan en faktor for antall timer per FP benyttes. Det er anbefalt at hver bedrift beregner dette ut fra bedriftens tidligere erfaringer. Antall timer per funksjonspoeng beregnes dermed ved hjelp av følgende formel [Garmus og Herron 2004];

$$\text{Antall timer per FP} = \frac{\text{Antall prosjekttimer}}{\text{Antall FP levert}}$$

Tabell 3-5 Antall timer per funksjonspoeng [Garmus og Herron 2004]

For å utvikle en verdi som er generell for alle prosjektene i bedriften må timefaktoren beregnes på nytt hver gang et prosjekt er fullført. Slik kan bedriften etter hvert finne sitt generelle timeantall per funksjonspoeng, for så å bruke denne i senere estimering. Antall timer per funksjonspoeng kan variere kraftig fra bedrift til bedrift. Verdien vil blant annet avhenge av hvilken type utvikling bedriftene gjennomfører, utviklingsprosess, programmeringsspråk og erfaring hos utviklerne [Garmus og Herron 2004].

3.3.4 Utfordringer

FPA fører med seg noen vanskeligheter. Generelt har FPA hatt en tendens til å villede i programmer med høy algoritmisk kompleksitet. Det er vanskelig å måle delsystemer for så å summere og finne et estimat for systemet som helhet. I tillegg bruker FPA fortsatt en terminologi som reflekterer tidlige systemutviklings-metoder [Longstreet 2004a].

FPA har også flere fordeler i bruk. Det at funksjonspoeng kan brukes uavhengig av språk og teknologi forenkler sammenlikning av programvare på tvers av prosjekter og bedrifter. Funksjonspoeng skal være lettere å forstå for mennesker som ikke har særlig erfaring innenfor datateknolog ved at det blir enklere for kunden å forstå størrelsen på programmet de ønsker. Funksjonspoeng kan også være til hjelp med kommunikasjon mot ledelsen samt beregnes av forskjellige mennesker til forskjellig tid. Allikevel skal det oppnås omtrent samme resultat med en rimelig feilmargin [Longstreet 2003].

3.4 MKII funksjons poeng analyse

MARKII funksjons poeng (MKII FP) ble utviklet av Charles Symons i 1988. Dette er en variant av FPA, stort sett brukt i England. Det ble oppdaget flere problemer med den opprinnelige FPA. Blant annet var vektleggingen lav, gjennomsnittlig og høy overforenklet. Dette førte til at veldig komplekse deler ikke ble riktig vektlagt. I den forbindelse ble MKII FP utviklet for å bedre vurderingen av intern prosesseringskompleksitet [Treble og Douglas 1995]. I dette studiet brukes ikke MKII FP. Det er allikevel tatt med en beskrivelse av metoden for å få en helhetlig forståelse av funksjonspoeng og use case poeng.

I MKII FP består informasjonssystemer av to store deler; informasjonshåndtering og teknisk implementering. I stedet for de fem kategoriene i FPA ser MKII FP systemet som en samling logiske transaksjoner. En logisk transaksjon defineres som en unik inndata/prosess/utdata kombinasjon. Denne utløses av interesse fra bruker [Treble og Douglas 1995].

I MKII FP er antall kostnadsdrivere utvidet til 20. De 14 første er de samme som i FPA. De seks neste er beskrevet i Tabell 3-6.

Kostnadsdriver		Beskrivelse - Spørsmål
15	Andre applikasjoner	I hvilken grad preger krav til andre applikasjoner utviklingen?
16	Sikkerhet	I hvilken grad preger sikkerhet og personvern utviklingen?
17	Brukeropplæring	I hvilken grad preger senere brukeropplæring utviklingen?
18	Tredje parter	I hvilken grad preger direkte bruk av tredje parter utviklingen?
19	Dokumentasjon	I hvilken grad preger krav til dokumentasjon utviklingen?
20	Klientdefinerte karakteristikker	I hvilken grad preger andre karakteristikker definert av kundene utviklingen?

Tabell 3-6 Utvidede kostnadsdrivere i MKII FP [Hansen 2002]

I MKII FP utgjør funksjonspoeng indeks (FPI) den totale størrelsen på et datasystem. I motsetning til den opprinnelige FPA ble MKII FPA utviklet til bruk med objektorienterte metoder [Treble og Douglas 1995].

3.5 COCOMO 81

I 1981 lanserte Barry Boehm den algoritmiske estimeringsmodellen COnstructive COst MOdel (COCOMO). Modellen omtales som COCOMO 81 på grunn av oppfølgeren COCOMO II som ble lansert første gang i 1997 [Boehm et al. 2000c]. COCOMO 81 blir ikke brukt til estimering i dette studiet. Modellen er beskrevet for å få en helhetlig forståelse av utviklingen til COCOMO II, se avsnitt 3.6.

COCOMO 81 baseres på et formelverk og et sett erfaringsdata. Modellen tar utgangspunkt i antall tusen kildekodelinjer (KLOC). Ved estimering varierer konstantene a, b, c og d i Tabell 3-8 etter programvarens størrelse. Størrelsen er delt inn i tre modus; organic, moderate eller embedded, se Tabell 3-7 og Vedlegg B.1.

Organic	Moderate	Embedded
Enkelt, kjent og stabilt < 50 KLOC	Begrenset erfaring < 300 KLOC	Komplekst og/eller ukjent trange rammer > 300 KLOC

Tabell 3-7 De tre modi i COCOMO 81 [Boehm 1981]

COCOMO 81 deles inn i tre presisjonsnivåer. *Basic* COCOMO gir grove estimater. *Intermediate* COCOMO innfører 15 kostnadsdrivere som modifiserer estimatene. *Detaljert* COCOMO trekker inn kostnadsdrivere for hver av de enkelte fasene i vannfallsmodellen [Boehm 1981].

En forutsetning for COCOMO 81 er sterk prosjektstyring og stabil kravspesifikasjon. Modellen er primært utviklet for en streng fossefallsmetode. Siden COCOMO 81 ble lansert i 1981 har store forandringer skjedd i programvare

og i utviklingsprosesser. Noe av grunnlaget for utviklingen av COCOMO II er tilbakemeldinger fra brukere av COCOMO 81 samt at måten å utvikle programvare er forandret [Boehm et al. 2000c].

For store prosjekter:

$$Innsats = a * størrelse^{b}$$

For prosjekter med 1 - 3 deltakere:

$$Innsats = a * størrelse + b$$

Varighet:

$$Varighet = c * Innsats^{d}$$

Tabell 3-8 Formler i COCOMO 81 [Boehm 1981]

3.6 COCOMO II

I 1994 startet prosjektet som resulterte i COCOMO II. Intensjonen var å håndtere prosjekter for neste generasjons programvareutvikling. Strategien bak COCOMO II er at modellen skal utvikles i inkrementer. Den skal testes slik at erfaringer utvinnes, og utvikles i samarbeid med ledende bedrifter innen programvare. Målet med modellen er at den skal kunne tilpasses enhver utviklingsprosess. Definisjoner i COCOMO II skal være enkle å forstå slik at også ferske brukere av modellen skal kunne lage riktige estimater [Boehm et al. 2000c].

COCOMO II kan skreddersys til alle typer organisasjoner og forskjellig type programvare. Ulik informasjon programvare imellom har resultert i tre metoder internt i COCOMO II; applikasjonssammensetning, tidlig design og post-arkitektonisk metode [Boehm et al. 2000c]. Applikasjonssammensetning benyttes ikke i dette studiet. En beskrivelse av denne metoden er vedlagt i Vedlegg B.1.

Tidlig design og postarkitektonisk modell benytter samme formler i estimeringen, se Tabell 3-9. Allikevel har metodene et ulikt detaljeringsnivå. I tidlig design kreves mindre detaljkunnskap enn for postarkitektonisk metode. COCOMO II baseres på programvarens størrelse målt i antall tusen kildekodelinjer (KLOC). UFP, objektpoeng og applikasjonspoeng kan gi grunnlaget for størrelse. Siden Boehm et al. (2000c) ikke konkret beviser at objektpoeng eller applikasjons-poeng gir nøyaktigere estimater enn UFP, benyttes UFP som grunnlag ved estimering av DES. Avhengig av programmeringsspråk konverteres UFP til KLOC med utgangspunkt i antall LOC per UFP [Boehm et al. 2000c, s. 20].

Innsats:

$$MV_{ns} = A * Størrelse^{E} * \prod_{i=1}^{n} EM_i$$

der

$$E = B + 0{,}01 * \sum_{j=1}^{5} SF_j$$

Varighet:

$$VAR_{ns} = C * (MV_{ns})F$$

der

$$F = D + 0{,}2 * 0{,}1 * \sum_{j=1}^{5} SF_j = D + 0{,}2 * (E - B)$$

Tabell 3-9 Formler i COCOMO II [Boehm et al. 2000c]

Eksponentiell skaleringsfaktorer		Beskrivelse
PREC	Forståelse	Fanger organisasjonens forståelse av applikasjonen og krav til teknologi. Vurderer i hvilken grad programmet likner på tidligere utviklede programmer.
FLEX	Utviklingsfleksibilitet	Vurderer i hvilken grad krav til programvaren, som kodekrav, karv til utvikling og eksterne grensesnitt standarder, følges.
RESL	Risikovurdering	Grad av integritet og forståelse av program arkitekturen samt risikoer ved utvikling.
TEAM	Team kohesjon	Grad av kohesjon i teamet. Her vurderes konsistens i kulturer, visjoner, erfaring i å operere som et team, teambygging og lignende.
PMAT	Prosessmodenhet	Mål på utviklingsprosessens modenhet. Basert på CMM, [Paul et al. 1995].

Tabell 3-10 Skaleringsfaktorer i COCOMO II [Boehm et al. 2000c]

Kostnadsdriverne i COCOMO II er delt i to grupper. *Eksponentiell skaleringsfaktor* (SF), se Tabell 3-10 og Vedlegg 0, er lik for både tidlig design og postarkitektonisk modell. Hver SF vurderes mot en skala fra veldig lav, lav, nominell og høy til veldig høy og ekstra høy [Boehm et al. 2000c, s. 30 - 36]. *Multiplikatorene* (EM) skiller tidlig design og postarkitektonisk modell. Tidlig design vurderer 7 multiplikatorer, se Tabell 3-11, mens postarkitektonisk metode vurderer 17 multiplikatorer, se Vedlegg B.2.2. Hver EM vurderes etter tilsvarende skala som SF [Boehm et. al 2000c, s. 36 - 57].

Beregningen i COCOMO II gir innsats målt i antall månedsverk (MV). Ved estimering av DES benyttes 152 timer per månedsverk [Boehm et al. 2000c].

COCOMO II tilbyr flere muligheter. Blant annet;

- å trekke gjenbruk av kode inn i utregningen
- å estimere månedsverk for veldikehold av kode etter utviklingsstopp
- å redusere estimert innsats ved å ta gammel kode og la et verktøy automatisk restrukturere den
- å fordele innsats på utviklingsfaser i for eksempel Unified Process (UP) [Larman 2002].

Når det i resten av studiet refereres til COCOMO II menes COCOMO II tidlig design. Det er tidlig design som er benyttet i estimeringen av DES.

Multiplikatorer		Tidlig Design
RCPX	Stabilitet og kompleksitet	Tilsvarer punktene stabilitet (RELY), databasestørrelse (DATA), kompleksitet (CPLX) og dokumentasjonskrav (DOCU) i post-arkitektonisk metode. Her vurderes graden av stabilitet, dokumentasjon, kompleksitet og databasestørrelse.
RUSE	Gjenbruk	Lik punktet gjenbruk (RUSE) i postarkitektonisk metode. Vurderer grad av tilleggsinnsats for å lage komponenter til gjenbruk i fremtidige applikasjoner.
PDIF	Plattformvansker	Tilsvarer punktene eksekveringstid (TIME), krav til minneplass (STOR) og plattformvansker (PVOL) i postarkitektonisk metode. Her vurderes grad av plattformstabilitet samt tid og lagringsfaktor prosentvis.
PERS	Personellets kompetanse	Tilsvarer punktene analysekompetanse (ACAP), programmeringskompetanse (PCAP) og personellets stabilitet (PCON) i post-arkitektonisk metode. Her vurderes kombinert analyse- og programmeringskompetanse i prosentsats. I tillegg vurderes også personellets stabilitet i prosentvis årlig fravær.
PREX	Personellets erfaring	Tilsvarer punktene applikasjonserfaring (APEX), plattformerfaring (PLEX) og språk- og verktøy-erfaring (LTEX) i postarkitektonisk metode. Her vurderes personellets gjennomsnittlige antall års erfaring med applikasjon, plattform, språk og verktøy.
FCIL	Fasiliteter	Tilsvarer punktene grad av verktøybruk (TOOL) og flersteds utvikling (SITE) i postarkitektonisk metode. Her vurderes graden av verktøystøtte.
SCED	Tidspress	Lik punktet tidspress (SCED) i postarkitektonisk metode. Vurderer grad av arbeid under tidspress i forhold til normalt.

Tabell 3-11 Multiplikatorer i tidlig design [Boehm et al. 2000c]

3.7 WEBMO

Slutten av 1990-årene førte med seg en eksplosjon i Internett- og intranettbaserte programmer. I den forbindelse ble Web Modell (WEBMO) utviklet for å tilpasses estimering av webutviklingsprosjekter. WEBMO ble utviklet på grunnlag av eksperters vurderinger og realistiske data fra 64 prosjekter. Den matematiske definisjonen i WEBMO bygger på COCOMO II tidlig design metodens dataanalyse av mer enn 161 prosjekter rettet mot webutvikling. WEBMO legger hovedvekten på webutviklingsprosjekter som raskt må være ute hos kunde [Reifer 2002].

Reifer (2002) mener at konverteringen fra FP til LOC ikke kan brukes for å måle riktig størrelse på webapplikasjoner. FP tar blant annet ikke hensyn til alle elementer en webapplikasjon består av. Dette gjelder elementer som knapper, bilder, maler, skript eller liknende. Andre metoder for webestimering har blitt foreslått, blant annet objekt poeng og applikasjonspoeng [Reifer 2002]. Ingen metode har vist seg som generelt bedre enn de andre. En metode kan imidlertid være best innenfor et avgrenset område. På bakgrunn av dette utviklet Reifer (2002) metrikken webobjekter som tar hensyn til alle elementer en webapplikasjon består av. Disse elementene er delt i fem grupperinger [Reifer 2002];

Ujusterte funksjonspoeng utgjør en del av ujusterte webobjekt. Beregningen av UFP gjøres på samme måte som beskrevet i avsnitt 3.3.

Antall linker er en størrelsesfaktor utviklet for å finne den innsats som kreves for blant annet å linke html/xml data, generere automatiske rapporter, integrere og animere forhåndsdefinert logikk, linke applikasjoner, integrere applikasjoner sammen dynamisk og binde dem til databassen eller andre applikasjoner.

Antall multimedia filer er en størrelsesfaktor utviklet for å finne den innsats som kreves for blant annet å sette lyd, video og bilde inn i applikasjoner.

Antall skript er en størrelsesfaktor utviklet for å finne den innsats som krevers for blant annet å generere rapporter, linke html/xml data med applikasjoner og filer, integrere dem dynamisk og binde dem mot databaser eller andre applikasjoner. Et use case vurderes som et skript.

Antall webbyggesteiner er en størrelsesfaktor utviklet for å finne den innsats som kreves for å utvikle finkornete komponenter eller biblioteker for web. Herunder gjelder ikke standard biblioteker som ligger inkludert i blant annet Java. I stedet gjelder byggeklosser som enten utvikles eller hentes inn til webapplikasjonen.

For grupperingene ovenfor telles komponenter tilhørende hver gruppering. Komponentene i hver gruppering kategoriseres etter kompleksitet; lav, gjennomsnittlig eller høy [Reifer 2002]. Antallet lave, gjennomsnittlige og høye komponenter for hver kategori summeres for så å multipliseres med en vektleggingsfaktor. Disse er vist i Tabell 3-12. Produktene summeres og gir antall webobjekter.

Webobjekt faktor	Lav	Gjennom-snittlig	Høy	Antall webobjekter
# UFP				Antall UFP
# multimediefiler	4	5	7	
# webbyggesteiner	3	4	6	
# skript	2	3	4	
# linker	3	4	6	
SUM	Sum Lav	Sum Gj.snitt	Sum Høy	Sum Lav + Gj.snitt + Høy + UFP

Tabell 3-12 Beregningsskjema for webobjekter [Reifer 2002].

Antall webobjekter brukes for å bestemme programvarens størrelse. Webobjektene konverteres til KLOC ved hjelp av Language Expanision Factors (LEF), se Vedlegg D. Formlene i WEBMO vises i Tabell 3-13. Konstantene A og B samt kraftlovene P1 og P2 settes ut fra det applikasjonsdomene utviklingen foregår i [Reifer 2002].

$$\text{Innsats} = A \prod_{i=1}^{8} cd_i (\text{Størrelse})^{P1} \qquad \text{Varighet} = B(\text{Innsats})^{P2}$$

Der

A og B er konstanter — cd_i er kostnadsdrivere

P1 og P2 er kraftlover — Størrelse er antall kilo kildekodelinjer

Tabell 3-13 Formeler i WEBMO [Reifer 2002]

WEBMO vurderer 9 kostnadsdrivere i estimeringen. Disse er beskrevet i Vedlegg C. Kostnadsdriverne vurderes og multipliseres på samme måte som multiplikatorene i COCOMO II. Estimatet i WEBMO er innsats målt i antall månedsverk. Som for COCOMO II defineres ett månedsverk til 152 timer.

4 USE CASE OG USE CASE POENG

Use case modellering er en populær og ofte brukt teknikk for å beskrive funksjonelle krav i ett programvaresystem. Funksjonelle krav forteller hvilke funksjoner et system skal inneholde. Use case poeng er en estimeringsmodell som tar utgangspunkt i funksjonelle krav. Dette kapitelet forklarer først hva use case er, for deretter å beskrive use case poeng. Til slutt refereres relatert arbeid.

4.1 Use Case

4.1.1 Historikk

De første skissene av det som senere er blitt til use case ble introdusert av Ivar Jacobson, da han jobbet hos Ericson i 1967. Jacobson arbeidet videre med utviklingen og etablerte Objectory AB i Stockholm i 1987. Sammen med flere ansatte formaliserte han grunnstegene for en utviklingsprosess kalt Objectory. Slik begynte use case analysen å ta form [Schneider og Winters 2001]. I 1992 introduserte Jacobson en programutviklingsmetodologi med objektorientering. Dette er den første use case drevne prosess der modellering av use case inkluderes i alle skritt av programvareutviklingen [Jacobson et al. 1992].

Det neste steget i anvendelse av use case var Gustav Karners estimeringsmodell; use case poeng [Karner 1993]. Modellen er en videreutvikling av Albrechts funksjonspoeng fra 1970 årene og baserer seg på Jacobsons arbeid med use case. Use case poeng tar utgangspunkt i funksjonelle krav, altså use case, for å estimere størrelse og innsats målt i arbeidstimer ved utvikling av objektorienterte systemer. Hensikten med use case poeng var at metoden skulle brukes i tillegg til COCOMO 81 eller funksjons poeng [Schneider og Winters 2001].

Neste skritt i utviklingen av use case oppsummeres av Cockburn (2001), der han oppsummerer sitt arbeid fra 1992 og fremover. Boken definerer use case og forklarer retningslinjer for hvordan use case skrives riktig [Cockburn 2001].

4.1.2 Systemavgrensning og terminologi

Ved utvikling av et systems funksjonelle krav må systemets avgrensninger defineres. Det må være klart hva systemet skal bestå av, hva som kommuniserer direkte med systemet og hva som indirekte avhenger av systemet. Først defineres aktører. En aktør berører systemet på et vis og har en egen oppførsel, men er ikke en del av systemet. En aktør er en rolle spilt av et menneske, en organisasjon, annen programvare, hardware komponenter, datalagre eller nettverk. Hver av disse komponentene kan være representert av en eller flere aktører [Schneider og Winters 2001].

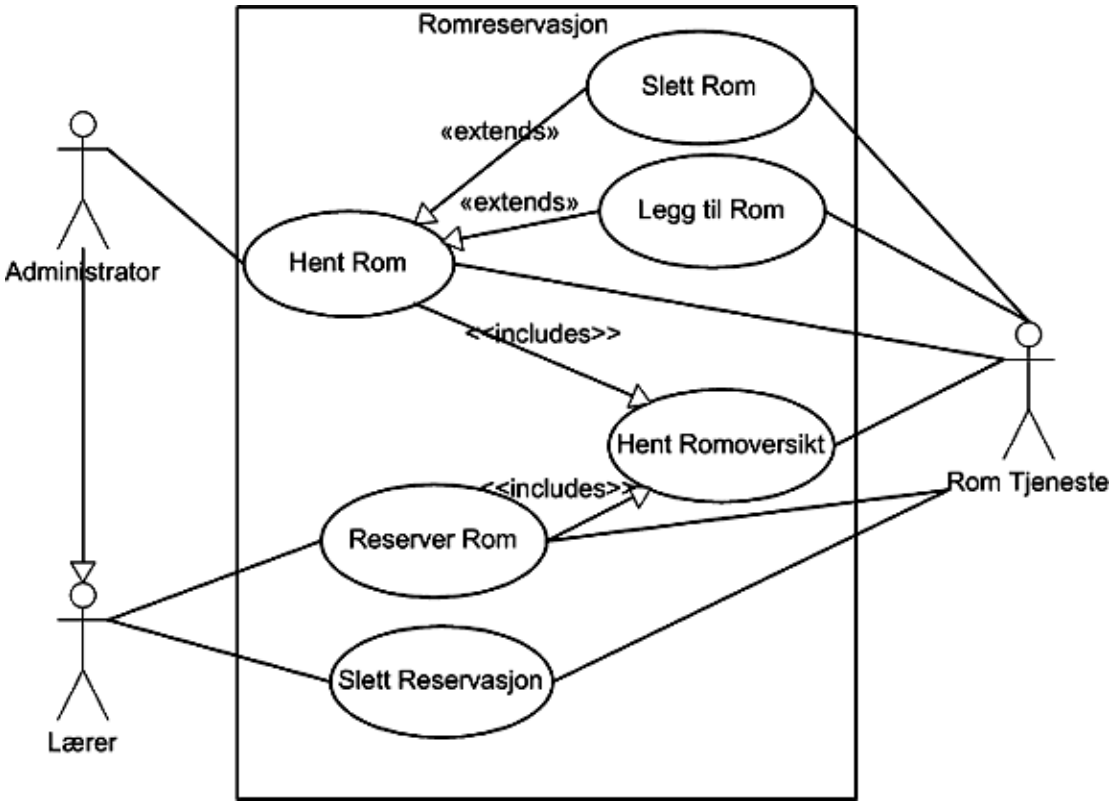
Aktører er delt inn i tre kategorier. En *primæraktør* er i dirkete kontakt med systemet for å oppfylle et eller flere ønsker. Det kan for eksempel være et menneske som skal bruke det fremtidige systemet. En *støtteaktør* supplerer systemet med assistanse for at primæraktøren skal få oppfylt sitt ønske. En primæraktør utløser en handling som kaller opp støtteaktørens ansvar. Den primære aktørens ønske oppfylles dersom støtteaktøren svarer. Svaret kan for

eksempel være data om ulike rom som hentes ut fra en database, der databasen utgjør støtteaktøren, i et system for romreservering ved en skole. En *indirekte aktør* har interesse for systemets oppførsel, men berører ikke systemet selv. En indirekte aktør for et reservasjonssystem ved en skole kan for eksempel være elevene ved skolen. De er ikke direkte brukere av reservasjonssystemet, men påvirkes allikevel av det [Larman 2002].

Et *scenario* er en sti gjennom systemet for en spesifikk hendelse, fra primær aktør gjennom et use case og ut til en støtte aktør. Et scenario kan for eksempel være en suksessfull registrering av en reservasjon i reservasjonssystemet for rom. Et use case er en samling suksessfulle og ikke suksessfulle scenarioer som utgjør en aktørs ønske. Et spesifikt ønske er et delmål ved systemet, som for eksempel "Reserver Rom" i et reservasjonssystem. Hver aktør har sitt ansvar relatert til systemet og kan kobles med andre aktører gjennom delmålene. [Larman 2002]

4.1.3 Grafisk use case

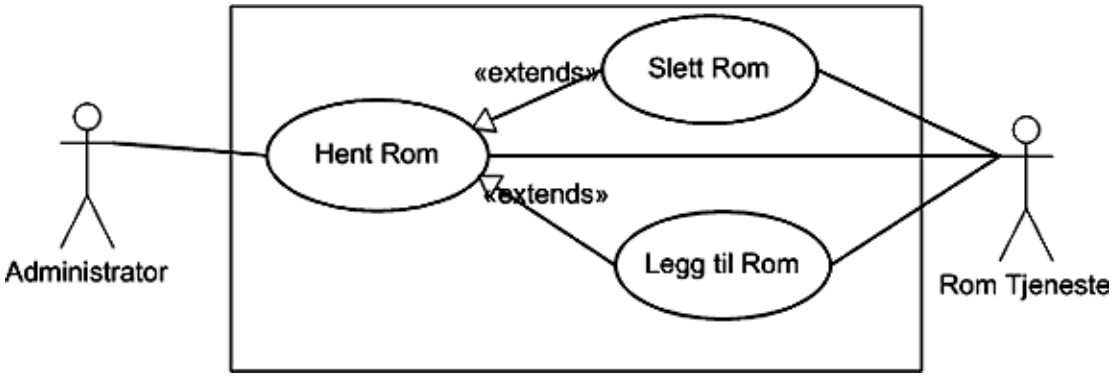
Det grafiske use case diagrammet er en statisk oversikt over funksjonelle krav i et system under utvikling. Aktørene tegnes som strekmennesker med tilknytning til en eller flere ellipser. Ellipsene representerer use case'ene i systemet som skal lages. Use casene omgis av et rektangel. Dette rektangelet representerer systemets grense, se Figur 4-1.



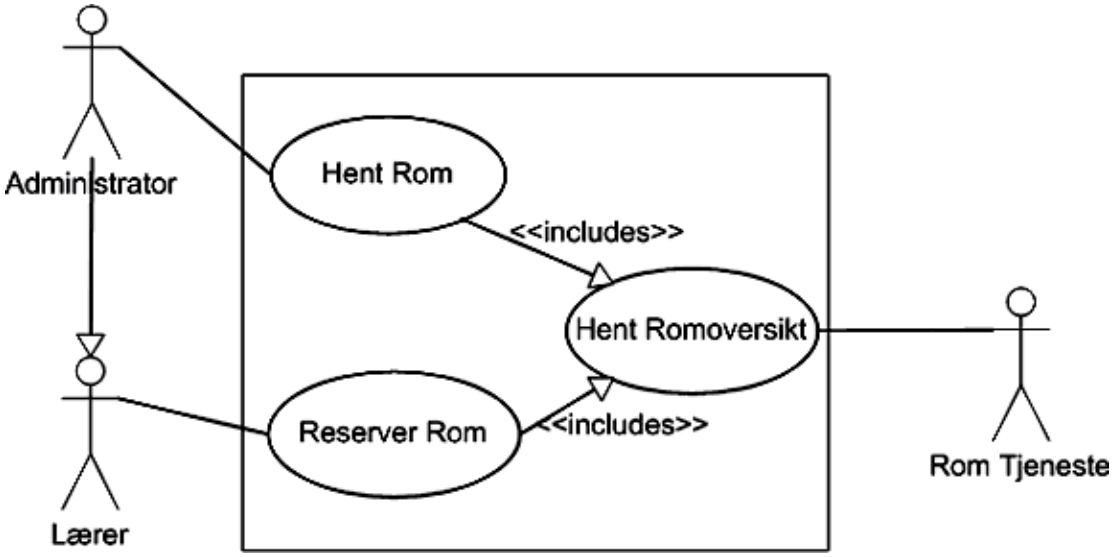
Figur 4-1 Utdrag av en grafisk use case modell for et reservasjonssystem

Use case diagrammet viser hvilke primæraktører og støtteaktører som finnes og hvordan de samhandler gjennom use case. Indirekte aktører vises ikke i et

grafisk use case diagram. Linjene mellom use case'ene, mellom aktørene, og mellom aktør og use case forklarer hvordan de ulike elementene samhandler. En linje kalles en assosiasjon. Figur 4-1 viser use case diagrammet for et reservasjonssystem for rom. I alle eksempler og forklaringer benyttes dette reservasjonssystemet. Det er et utdrag av et planleggerprogram laget som en hovedoppgave ved Høgskolen i Sør-Trøndelag. Utdraget har følgende funksjoner; hent rom, slett rom, legg til rom, reserver rom og slett reservasjon.



Figur 4-2 Utvidet use case



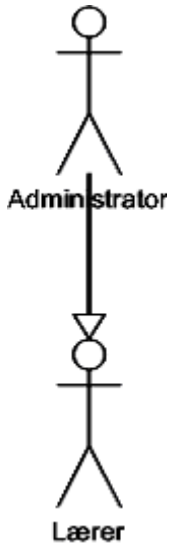
Figur 4-3 Inkludert use case

Et utvidet use case, se Figur 4-2, indikerer en assosiasjon mellom to use case der de to use case'ene, "Legg til Rom" og "Slett Rom", er en utvidelse av det første use case'et, "Hent Rom". I enkelte situasjoner kan det være nødvendig med utvidet funksjonalitet til "Hent Rom" og et av de utvidede use case'ene gjennomføres. Et utvidet use case er et spesielt tilfelle av hoved use case'et. "Slett Rom" og "Legg til Rom" er avhengig av "Hent Rom", og kan ikke eksistere uten [Kulak et al. 2004]. Utvedede use case merkes i diagrammet ved assosiasjonen «extends».

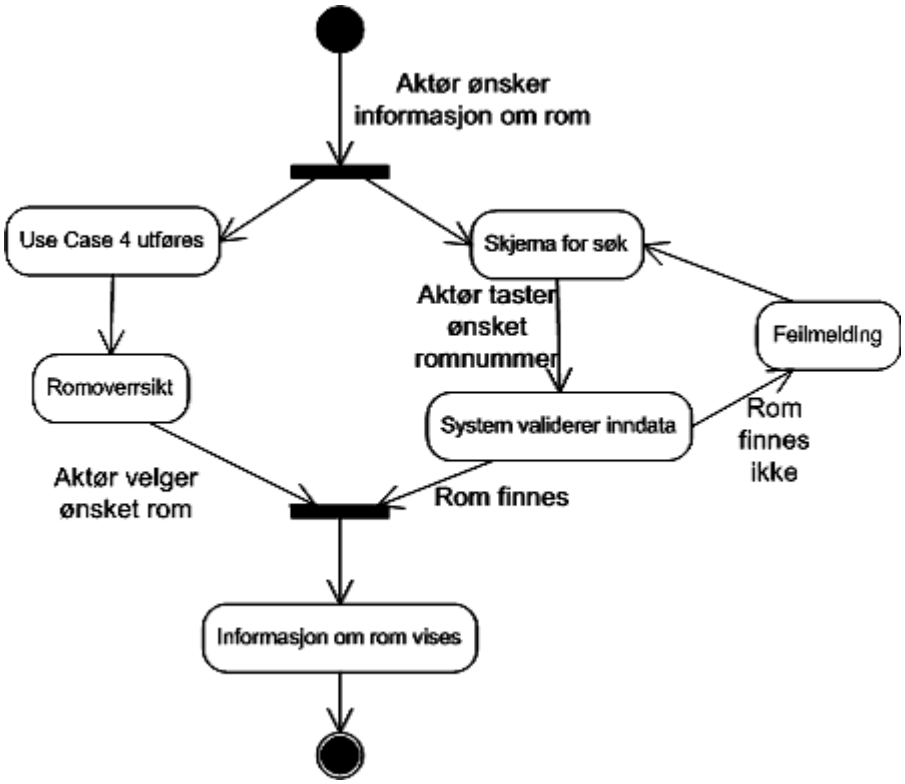
Inkluderte use case indikerer fellestrekk som går igjen i flere use case. Fellestrekkene kan trekkes ut og danne egne use case. I Figur 4-3 inneholder både ”Hent Rom” og ”Reserver Rom” de samme stegene for å få ut en oversikt over rom. I den forbindelse er ”Hent Romoversikt” trukket ut i et eget use case. [Kulak et al. 2004]. Et inkludert use case merkes i diagrammet ved assosiasjonen «includes».

Konseptet med generalisering er hentet fra objektorientering. Generalisering kan brukes når et element er en type av et annet element. Konseptet brukes både for aktører og for use case. Ved generalisering mellom use case er et use case en spesifikasjon av et annet use case. Det spesifiserte use case’et arver egenskapene fra det generelle use case’et. Ved generalisering mellom aktører fyller en subaktør samme rolle som en superaktør, i tillegg til andre roller subaktøren måtte ha [Schneider og Winters 2001]. I Figur 4-4 arver ”Administrator” som er subaktør alle assosiasjoner til use case fra ”Lærer” som er superaktør. I tillegg besitter ”Administrator” en assosiasjon som ”Lærer” ikke har. Ved å bruke generalisering mellom aktører blir det grafiske use case diagrammet mer oversiktlig. Færre assosiasjoner tegnes inn. I tillegg beskrives felles egenskaper kun en gang [Kulak et al. 2004].

Use case’ene kan spesifiseres i tilstands- og/eller aktivitetsdiagrammer [Mathiassen et al. 2000]. Diagrammene beskriver overgangen fra en tilstand til en annen. Aktivitetsdiagrammet til use case 1 i Tabell 4-1 vises i Figur 4-5. I et aktivitetsdiagram vises både hovedflyt og alternativ flyt. Et aktivitetsdiagram gir et godt overblikk over use case’et. Allikevel utelates flere detaljer. Som for eksempel aktører, pre- og postbetingelser og spesifisering av hva som er hovedflyt, alternativ flyt og utvidelser. En annen type spesifikasjon kan være systemsekvensdiagram. Dette beskrives ikke her.



Figur 4-4 Generalisering mellom aktører



Figur 4-5 Eksempel på aktivitetsdiagram for use case "Hent Rom".

4.1.4 Tekstlig use case

Et komplett use case er egentlig noen sider tekst der hvert delmål i systemet detaljeres stegvis. Denne spesifiseringen beskriver interaksjonen mellom aktør og system under utvikling. I dette studiet benyttes malen fra [Kulak et al. 2004]. To av de tekstlige use case'ene til reservasjonssystemet er vist i Tabell 4-1 og Tabell 4-2. Samtlige use case vises i Vedlegg E.

En use case beskrivelse bør først inneholde [Kulak et al. 2004];

- Unik use case identifikator, gjerne et navn og et enkelt nummer
- I hvilken iterasjon use case'et er skrevet
- Sammendrag med kort beskrivelse av use case'ets hovedmål
- Prebetingelse som beskriver hvilke betingelser som må være oppfylt før use case'et kan gjennomføres
- Postbetingelse som beskriver tilstandsendringen etter at hovedflyten i use case'et er gjennomført
- Beskrivelse av hvilke aktører som samhandler med use case'et

Use Case navn:	UC1: Hent Rom
Iterasjon:	1
Sammendrag:	Henter oversikt over et spesifikt rom
Prebetingelse:	Innlogget som administrator
Postbetingelse:	Hente informasjon om et rom
Aktør:	Administrator
Hovedflyt:	1. Aktør ønsker å hente informasjon om et rom 2. Inkluder UC4: Hent Romoversikt 3. System viser oversikt over alle rom 4. Aktør velger rom 5. System viser data om valgt rom 6. System klart for nye kommandoer
Alternativ flyt:	A-1: *. Aktør kan når som helt avslutte A-2: 1a. 1. Aktør søker opp rom ved å taste rom nummer 2. System henter rom 3. Fortsett fra hovedflyt punkt 5 A-3: 1. Aktør taster ugyldig rom 2. System fortsetter hovedflyt punkt 1
Utvidet flyt:	U-1: 3a. 1. UC3: Legg til Rom 2. Fortsett punkt 6 i hovedflyt U-2: 5a. 1. UC2: Slett Rom 2. Fortsett punkt 6 i hovedflyt
Forfatter:	
Dato:	

Tabell 4-1 Use Case 1: Hent Rom

Hovedflyt beskriver stegvis det vanligste suksesscenarioet gjennom use case'et. Her beskrives ingen feilmeldinger eller alternative scenarioer. En aktør tar alltid det første steget, for deretter å få respons fra systemet. Denne samhandlingen gjennomføres helt til aktørens mål er oppnådd [Kulak et al. 2004].

Kulak et al. (2004) anbefaler et punkt for alternativ flyt i tillegg til et punkt for feilflyt. I dette studiet brukes alternativ flyt og feilflyt under samme punkt. Grunnen til dette er at use case'ene i DES er små og lite komplekse. Tabell 4-1 viser et eksempel av alternativ flyt. A-2 beskriver søk etter rom ved inntasting av rom nummer i stedet for å liste opp alle rom. Punktnummereringen under A-2 viser at denne alternative flyten kan startes allerede fra punkt 1 i hovedflyten.

Utvidet flyt viser tekstlig beskrivelse av utvidede use case. Utvidet flyt U-1 i Tabell 4-1, kan gjennomføres fra punkt 3 i hovedflyten. Tabell 4-2 beskriver det utvidede use case'et som U-2 i Tabell 4-1 viser til.

Til slutt registreres navn på personen som har skrevet use case'et, samt den dato use case'et ble skrevet. Tekstlige use case er nødvendig for å få en viss oversikt over hva som skal skje for hvert av aktørenes ønsker. Dette er en oversikt både

utvikler og kunde skal forstå. På den måten kan use case utgjøre kontrakten og kvalitetskontrollen mellom leverandør og kunde.

Use Case navn:	UC2: Slett Rom
Iterasjon:	1
Sammendrag:	Aktør ønsker å slette et registrert rom
Prebetingelse:	Innlogget som administrator
Postbetingelse:	Rom slettet
Aktør:	Administrator
Hovedflyt:	1. Aktør velger slett rom 2. System ber om bekreftelse på slett rom 3. Aktør bekrefter sletting av rom 4. System sletter rom
Alternativ flyt:	A-1: *. Aktør kan når som helt avslutte A-2: 4a. 1. Aktør avbryter sletting av rom 2. System returner til hovedflyt punkt 1 3. System klart for nye kommandoer
Forfatter:	
Dato:	

Tabell 4-2 Use Case 2: Slett Rom

4.2 Use Case Poeng

Med god forståelse av funksjonelle krav kan utviklere tidlig estimere tids- og resursforbruk i programvareutviklingsprosjekter. For å lage dette estimatet tar use case poeng utgangspunkt i use case.

4.2.1 Kategorisering av Aktører

Først kategoriseres aktørene til følgende grupper: enkel, gjennomsnittlig og kompleks. Enkel representerer et annet system med definert API og blir vektlagt med 1. Gjennomsnittlig representerer et system som påvirkes med protokoller som TCP/IP, denne vektlegges til 2. Kompleks representerer en person som opererer gjennom et grafisk brukergrensesnitt eller en web side, denne vektlegges til 3. Ved hjelp av disse kriteriene beregnes den første konstanten, ujustert aktørvekt (UAW - unadjusted actor weights). UAW kalkuleres ved å telle antall aktører i hver kategori, multiplisere antallet med vektfaktoren og til slutt summere produktene fra hver kategori.

For å vise utregningen av use case poeng benyttes eksempler fra reservasjonssystemet for rom i Figur 4-1. I dette eksempelet finnes det tre aktører. Lærer og Administrator er roller spilt av mennesker som kommuniserer gjennom et grafisk brukergrensesnitt. Disse kategoriseres til komplekse aktører. RomTjeneste er et lagringsmedium, for eksempel en database. Dette blir en enkel aktør.

```
UAW = ( 1 enkel * 1 ) + ( 2 komplekse * 3 ) = 7
```

4.2.2 Kategorisering av use case

Hvert use case kategoriseres til følgende grupper; enkel, gjennomsnittlig og kompleks. Hvilke use case som hører til hvilken kategori bestemmes ut fra antall

transaksjoner use case'ene består av. Transaksjoner telles i hovedflyt og alternativ flyt. Inkluderte og utvidede use case telles ikke [Karner 1993]. En transaksjon defineres som en handling mellom aktør og system. En handling er utført når aktør spør og system svarer. Tabell 4-3 viser forskjellen på enkel, gjennomsnittlig og komplekst use case med vektleggingsfaktor.

Use Case type	Beskrivelse	Faktor
Enkel	3 eller færre transaksjoner	5
Gjennomsnittlig	4 til 7 transaksjoner	10
Kompleks	8 eller flere transaksjoner	15

Tabell 4-3 Kompleksitet i use case

Ujustert use case vekt (UUCW - unadjusted use case weights) beregnes ved å telle antall use case i hver kategori, multiplisere antallet med vektleggings-faktoren og summere produktene for hver kategori.

Siden inkluderte og utvidede use case ikke er med i beregningen, telles antall transaksjoner i følgende use case; Hent rom, Reserver Rom og Slett Reservasjon. For fullstendig tekstlig use case, se Vedlegg E.

```
Hent Rom – tre transaksjoner gir enkelt use case.
Reserver Rom – fire transaksjoner gir gjennomsnittlig use case.
Slett Reservasjon – tre transaksjoner gir enkelt use case.

UUCW = ( 2 enkle * 5 ) + ( 1 gjennomsnittlig * 10 ) = 20
```

Til slutt beregnes ujusterte use case poeng (UUPC - unadjusted use case points) ved å summere UAW og UUCW. I eksempelet finnes følgende:

```
UUPC = UAW + UUCW = 7 + 20 = 27
```

4.2.3 Tekniske faktorer og omgivelsesfaktorer

UUPC justeres ved å ta hensyn til noen tekniske faktorer og faktorer med hensyn på omgivelsene rundt utviklingen. Tekniske faktorer tar for seg ikke funksjonelle krav, som responstid, flyttbarhet, effektivitet og liknende. Tabell 4-5 viser komplett liste over tekniske faktorer med hver av faktorenes forhåndsdefinerte vekt. Hver teknisk faktor vurderes i forhold til faktorens relevans for utviklingen. Skalaen går fra 0 til 5, der 0 betyr irrelevant for utviklingen mens 5 betyr essensielt for utviklingen. Kolonnen vurdering i Tabell 4-5 viser vurderingen av de tekniske faktorene i reservasjonssystemet. Beregnet vurdering for hver faktor er vekten multiplisert med vurderingen.

TFactor beregnes ved å summere beregnet vurdering hos faktorene T1 til T13. Til slutt kan den tekniske kompleksitetsfaktoren utregnes (TCF - Technical Complexity Factor). Følgende formel beregner TCF;

TCF = 0,6 + (0,01 * TFactor)

Tabell 4-4 Formel for beregning av TCF

Nummer	Beskrivelse	Vekt	Vurdering	Beregnet vurdering
T1	Distribuert system	2	1	2
T2	Responstid	1	2	2
T3	Sluttbrukereffektivitet	1	4	4
T4	Kompleks prosessering	1	1	1
T5	Gjenbrukbar kode	1	0	0
T6	Enkelt å installere	0,5	2	1
T7	Brukbarhet	0,5	5	2,5
T8	Flyttbarhet	2	0	0
T9	Enkelt å forandre	1	1	1
T10	Flerbruk	1	5	5
T11	Sikkerhetsregler	1	2	2
T12	Tilby tilgang til tredje parter	1	3	3
T13	Brukeropplæring	1	0	0

Tabell 4-5 Tekniske faktorer i UCP-metoden

Dersom kolonnene Vurdering og Beregnet vurdering i Tabell 4-5 beskriver vurderingene av de tekniske faktorene i romreservasjons-systemet blir utregningen av TCF følgende;

TFactor = 2 + 2 + 4 + 1 + 0 + 1 + 2,5 + 0 + 1 + 5 + 2 + 3 + 0 = 23,5
TCF = 0,6 + (0,01 * TFactor) = 0,6 + (0,01 * 23,5) = 0,835

Omgivelsesfaktorene beskriver ulike aspekter ved utviklingsmiljøet som kan påvirke programvareutviklingen. Tabell 4-7 viser en komplett oversikt over omgivelsesfaktorene med hver av faktorenes forhåndsdefinerte vekt. Omgivelsesfaktorene beregnes på omtrent samme måte som de tekniske faktorene. Hver av omgivelsesfaktor vurderes individuelt i forhold til en skala mellom 0 og 5. Ribu (2001) har identifisert hvert nivå i skalaen for hver omgivelsesfaktor. For faktorene E1 til E4 betyr 0 ingen erfaring eller nybegynner på området. 2 – 3 betyr 1 – 1 ½ års erfaring for noen av utviklerne innenfor området. 5 betyr ekspert. For E5 betyr 0 ingen motivasjon, 3 – 4 betyr at utviklerne er motivert til å gjøre en god jobb og 5 betyr at de er ekstra motivert og inspirert. For E6 betyr 0 ekstremt ustabile krav, 3 – 4 betyr generelt stabilt mens 5 betyr uforanderlige krav. For E7 betyr 0 ingen deltidsarbeidende, 3 – betyr at halve utviklingsgruppen er deltidsarbeidende mens 5 betyr at samtlige arbeider deltid. For E8 betyr 0 at alle utviklerne er erfarne programmerere. 3 betyr at de fleste utviklerne har mer enn ett års erfaring. 5 betyr at alle utviklerne er nybegynnere [Ribu 2001].

Hver faktor vurderes for så å multiplisere vurderingen med den fastsatte vektoren. EFactor beregnes ved å summere beregnet vurdering for E1 til E8. Tabell 4-7 viser vekt, vurdering og beregnet vurdering for reservasjonssystemet

beskrevet tidligere. Omgivelsesfaktoren EF (the enviromental factor), beregnes ved følgende formel;

EF = 1,4 + (-0,03 * EFactor)

Tabell 4-6 Formel for beregning av EF

Dersom kolonnene Vurdering og Beregnet vurdering i Tabell 4-7 beskriver vurderingene av omgivelsesfaktorene i romreservasjons-systemet blir utregningen av EF følgende;

EFactor = 4,5 + 1,5 + 4 + 1,5 + 4 + 6 + (- 2) + (- 3) = 16,5
EF = 1,4 + (- 0,03 * EFactor) = 1,4 + (- 0,03 * 16,5) = 0,905

Nummer	Beskrivelse	Vekt	Vurdering	Beregnet vurdering
E1	Kjennskap til utviklingsprosess	1,5	3	4,5
E2	Applikasjonserfaring	0,5	3	1,5
E3	Objektorientert erfaring	1	4	4
E4	Analysekompetanse	0,5	3	1,5
E5	Motivasjon	1	4	4
E6	Stabile krav	2	3	6
E7	Deltids arbeid	-1	2	-2
E8	Vansker med programmeringsspråk	-1	3	-3

Tabell 4-7 Omgivelsesfaktorer i UCP-metoden

EF og TCF er to faktorer som justerer UUCP til use case poeng (UCP). Dette beregnes på følgende måte;

UCP = EF * TCF * UUCP

Tabell 4-8 Formel for beregning av UCP

For reservasjonssystemet gjelder;
UCP = EF * TCF * UUCP = 0,905 * 0,835 * 27 = 20,40

4.2.4 Estimering av arbeidstimer

UCP brukes til å estimere antall arbeidstimer nødvendig for utvikling av systemet beskrevet i funksjonelle krav. Da Karner (1993) første gang presenterte metoden foreslo han å bruke 20 arbeidstimer per UCP. Anda et al. (2001) viser til en artikkel skrevet av Sparks i 1999 som sier at antall arbeidstimer per UCP kan variere fra alt mellom 15 og 30. Det å overføre UCP direkte til arbeidstimer kan gi stor usikkerhet. Ved å gjennomgå omgivelsesfaktorene på nytt forklarer Schneider og Winters (2001) at antall arbeidstimer per UCP kan bestemmes ut fra utviklernes erfaringsnivå. Dersom erfaringsnivået er lavt er det naturlig å bruke litt lenger tid på læring og utvikling, og nødvendig timeantall øker. For å

finne riktig antall arbeidstimer per UCP brukes følgende prosedyre; Summer antall vurderinger lavere enn 3 for omgivelsesfaktorene E1 til E6 med antall vurderinger høyere enn 3 for omgivelsesfaktorene E7 og E8. Benytt 20 arbeidstimer per UCP dersom totalsummen er 2 eller mindre. Er totalsummen 3 eller 4 benytt 28 arbeidstimer per UCP. Dersom totalsummen er 5 eller mer bør prosjektet revideres slik at verdiene justeres. Et alternativ er å bruke 36 arbeidstimer per UCP. Resultatet av UCP-metoden er et estimat på totalt antall arbeidstimer nødvendig for å gjennomføre et programutviklingsprosjekt.

I reserveringssystemet benyttes den siste metoden for å bestemme timefaktor. I E1 til E6 har ingen faktorer lavere vurdering enn 3. For E7 og E8 har heller ingen faktor høyere vurdering enn 3. Det betyr at totalsummen er 0 og 20 arbeidstimer per UCP benyttes.

20,40 UCP * 20 arbeidstimer per UCP gir estimatet 408 arbeidstimer.

4.3 Utfordringer med Use Case Poeng

UML er kun et notasjonsspråk for å illustrere diagrammer i programvareutvikling. UML forteller ingen ting om hvordan disse diagrammene skal spesifiseres. Per i dag finnes ingen standard på hvordan et use case lages. Det er opp til hver enkelt bedrift å drøfte dette og finne den måten som passer best for deres prosjekter [Kulak et al. 2004]. Utviklerne må selv bestemme hvilket use case format som passer best. Dette gjør det vanskelig for bedrifter seg i mellom å sammenlikne use case og use case poeng.

I noen tilfeller er det vanskelig å få use case'ene så korrekte at de danner et godt grunnlag for estimeringen. Å bedømme størrelse og kompleksitet når use case mangler tekst og/eller figur er vanskelig. For gode estimater i disse tilfellene kreves eksperters vurdering i tillegg. Ribu (2001) viser hvordan use case med manglende tekst men gode sekvensdiagrammer kan konverteres til logiske transaksjoner i MKII FPA på bakgrunn av Symons (2001).

Studier viser at nøyaktighet når en skriver use case diagrammer er avgjørende for utfallet av estimatene. Det er stor forskjell i hvordan utviklere i samme prosjekt kan representere et use case. En kan bruke et høyt detaljeringsnivå mens en annen generaliserer mer. I UCP-metoden gir denne forskjellen i detaljeringsnivå store forskjeller i estimeringen [Merrick 2005b]. Cockburn (2001) konkluderer med at use case som har over 10 steg i hovedflyt ofte inneholder komponenter for brukergrensesnitt eller er skrevet for detaljert. På bakgrunn av dette presenterer Ribu (2001) huskereglen at ved mer enn 10 steg i hovedflyt bør use case'et redigeres. Enten ved forkorting eller ved å skille ut funksjonalitet.

Inkluderte og utvidede use case telles ikke med ved kategorisering av use case [Karner 1993, Schneider og Winters 2001]. Allikevel må denne funksjonaliteten implementeres. Dersom inkluderende og utvidede use case inneholder stor del av essensiell funksjonalitet vil det være nødvendig å inkludere disse i tellingen. Dette fører til at estimatene øker. Kulak et al. (2004) fraråder å bruke inkluderende og utvidede use case. Det er alltid muligheter for å unngå disse use case'ene. Å la være å bruke inkluderende og utvidede vil generelt gi høyere estimater. Uansett vil det være bedre å overestimere enn å underestimere for mye. En annen måte å justere use case estimatene på er å generalisere aktørene til en superaktør. Ofte har flere aktører like use case. Dette kan føre till høyere

estimater enn nødvendig ved bruk av UCP. En generalisering mellom aktører fører dermed til færre aktører å telle som igjen gir et lavere estimat [Anda et al. 2001]. Disse justeringene testes ut i estimeringen av DES.

UCP-metoden baseres på et systems funksjonelle krav. Det betyr at kun funksjonaliteten blir vurdert i estimeringen. I programvareutvikling utføres flere aktiviteter enn kun arbeid med funksjonaliteten. Opplæring og trening for å lage nytt utviklingsmiljø eller prosjektadministrasjon er faktorer som ikke tas direkte hensyn til i UCP-metoden. Schneider og Winters (2001) foreslår å legge til tre uker i estimatene, for å gjennomarbeide eventuell opplæring eller annen nødvendig administrasjon. Fortsatt er det usikkert om kravutarbeidelse, testing og dokumentasjon er med i beregningen. UCP-metoden er fortsatt lite utprøvd. Mer forskning er nødvendig for å finne nøyaktigheten i estimatene.

4.4 Relatert arbeid

4.4.1 Estimering med hjelp av use case

Anda et al. (2001), Ribu (2001), Anda (2002) og Anda et al. (2002) beskriver UCP-metodens estimater i forhold til eksperters estimater. I Ribu (2001) ble studien av ekspertenes estimater i forhold til UCP-metodens estimater gjennomført over to prosjekter i en større bank. Det ene systemet var en webapplikasjon. Det andre prosjektet var en del av et større kommersielt system. Generelt for begge prosjektene lå ekspertenes estimater henholdsvis 2300 og 1300 timer under det faktisk brukte timeantallet. Med UCP-metoden estimerte Ribu (2001) i snitt 850 timer over brukt timeantall.

Ribu (2001) gjør også en studie på UCP-metoden i forhold til studenters estimater. Her var alle deltakerne uerfarne, både med use case, UML og estimering. Resultatet ble mer en bekreftelse på en av UCP-metodens utfordringer diskutert i avsnitt 4.3; nøyaktighet og detaljnivå i tekstlige use case påvirker estimatene. På grunn av lite detaljerte use case ble en del av prosjektene kraftig underestimert. Andre ble kraftig overestimert på grunn av for detaljerte use case. Ribu (2001) restrukturerte samtlige use case noe som generelt reduserte estimert timeantall.

Generelt kommer Ribu (2001) frem til at UCP-metoden er en god støtte for eksperters estimater. I tillegg foreslår Ribu (2001) at antall UCP som ligger til grunn for beregning av antall timer skal utelate teknisk faktor. Ribu presenterer også hva som bør ligge til grunn i skalaen for vurdering av omgivelsesfaktorer sammen med et forslag til hvordan beregne UCP av lite detaljerte use case.

4.4.2 UCP og eksperters estimater

Anda et al. (2001) gjennomfører en studie i industrien med den hensikt å bedre et firmas estimater. Her ble det samlet data fra tre utviklingsprosjekter på omtrent samme størrelse og med samme omfang. Meningen var å sammenlikne ekspertenes estimater med UCP-metodens estimater. UCP-metodens estimater var i dette tilfellet veldig nær ekspertenes egne estimater. Ekspertene hadde alle erfaring fra liknende prosjekter. Anda et al. (2001) ønsker å studere UCP-metodens brukbarhet i andre omgivelser, der ekspertene har mindre erfaring eller i andre typer prosjekter. Ribu (2001) har delvis fortsatt med dette i studiene beskrevet i avsnitt 4.4.1.

Anda (2002) gjennomførte et studie ved å dele ut et sett use case til en gruppe eksperter. Ut fra de funksjonelle krav beskrevet i use case'ene skulle ekspertene lage estimater. Ingen av ekspertene kjente til UCP-metoden. Dette førte til at ulike estimeringsmetoder ble brukt. Anda skriver at ekspertene estimerte relativt bra i forhold til den kunnskap de hadde om systemet. I forhold til faktisk brukt tid på utviklingen var estimatene fra UCP-metoden generelt bedre. Dette støtter tidligere resultater med at UCP-metoden er god for å støtte eksperter. Anda fortsetter med å konkludere at UCP-metoden kombinert med ekspertenes kunnskap kan være fordelaktig når ekspertene vet lite om applikasjonsdomenet og teknologien brukt. Her anbefales å fortsette arbeidet med å teste UCP-metoden i andre typer utviklingsprosjekter. Denne studien tar tak i dette ved å teste UCP-metoden på området for utvikling av enkle webapplikasjoner.

4.4.3 Forenkling av UCP-metoden

Merrick (2005c) beskriver et forslag for hvordan UCP-metoden kan forenkles. Hensikten var å gjøre UCP-metoden egnet til raskere å kunne kalkulere viktige estimater for en kunde til bruk tidlig i programvareutviklingens syklus. Merrick (2005c) foreslår å redusere antall aktørkategorier til 2. Han mener at det holder å skille mellom primæraktør og støtte aktør der primæraktøren regnes som kompleks og støtteaktøren som enkel. Et liknende forslag om enkel og kompleks foreslås i forbindelse med kategorisering av use case. Et enkelt use case tilsvarer et use case som lager en ny instans, søker igjennom et sett data og returnerer en eller flere instanser, eller modifiserer en instans. Enkle use case kalles NSM (New, Search, Modify), og forbindes med CRUD-funksjoner (Create, Read, Update, Delete). Komplekse use case defineres til å være alt som ikke er enkle use case. Til dette foreslås følgende vekter;

Komponent	Vekt
Primær aktør	3
Sekundær aktør	1,5
Enkelt use case	5
Komplekst use case	10

Tabell 4-9 Forslag til vektlegging i forenklet UCP-metoden [Merrick 2005c]

Merrick (2005c) ser ut til å utelate både omgivelsesfaktorer og tekniske faktorer i forenklingen. I stedet foreslås ulike måter å beregne antall arbeidstimer på. En foretrukket metode er for hver bedrift å reversere et system som allerede har blitt laget med bruk av samme regler, teknologi og programmeringsspråk som det systemet som skal lages. Det er viktig at totalt brukt tid på dette prosjektet er kjent, samt utgangspunktet med de dokumenter som var tilgjengelig i begynnelsen. Slik telles UCP for det allerede ferdige prosjektet og det totale antall arbeidstimer divideres med antall UCP. Produktet blir da antall timer organisasjonen trenger for å levere funksjonalitet for ett UCP. Metrikken brukes så til estimering i det nye prosjektet. Merrick (2005c) avslutter med å forklare betydningen med det å justere beregningene kontinuerlig. Han nevner at både faktoren programmeringstid og tid ikke brukt til programmering må medregnes i kalkuleringen. Antall timer brukt i hver av faktorene må kontinuerlig noteres for hele tiden å kunne forbedre beregningen som utgjør grunnlaget for estimeringen.

4.4.4 Use Case og Funksjonspoeng

Longstreet (2003) beskriver en metode for å beregne funksjonspoeng ut fra use case. Da use case generelt brukes i objektorienterte programmer, er det blant annet denne overgangen som gjør funksjonspoenganalyse brukbar i objektorientert utvikling. Longstreet (2003) skriver at et av problemene med funksjonspoeng har vært et mangelfullt og ukonsistent sett med krav. Ved å bruke use case sammen med funksjonspoeng økes kvaliteten på kravdokumentene. Med dette øker også nøyaktigheten i å bestemme programmets størrelse og nøyaktigheten i estimeringen. Videre forklarer Longstreet (2003) at fordelen ved å bruke use case og funksjonspoeng sammen er muligheten med å estimere prosjektene tidligere i livssyklusen. I tillegg er det enklere å oppdatere estimatene ettersom use case'ene forandres.

5 FORSKNINGSMETODE

Dette kapittelet beskriver forskningsmetoden brukt i dette studiet. Først introduseres forskning generelt. Deretter diskuteres datainnsamling og til slutt analysen.

5.1 Introduksjon

Begrepet forskning omfatter all faglig innsats for tilegning av ny kunnskap. For å oppnå denne kunnskapen planlegges, kartlegges og analyseres over lengre perioder. Bestemte strategier og fremgangsmåter benyttes slik at resultatene blir pålitelige. Forskning kan deles inn i kvantitativ forskning og kvalitativ forskning. Mange ulike metoder benyttes for å gjøre forskning. Blant annet kartlegging, eksperiment og case studier. Kartlegging og eksperiment er oftest kvantitativ mens case studie oftest er kvalitativ.

5.1.1 Kvalitativ forskning

Silverman (2001, s. 25) introduserer kvalitativ forskning slik; *"Å kalle seg selv kvalitativ forsker fastsetter overraskende lite"*. Dette er første indikasjon på at uttrykket "kvalitativ forskning" inneholder mye. Creswell (2003) karakteriserer kvalitativ forskning med noen hovedpunkter;

- Kvalitativ forskning foregår i en naturlig kontekst. Forskeren tar i større eller mindre grad del i miljøet det forskes i
- Kvalitativ forskning blander metoder for å få frem resultater
- Kvalitativ forskning er fundamentalt tolkende. På bakgrunn av analyser og resultater gjør forsker en tolkning av data. Dette gjelder personlig tolkning, teoretisk tolkning samt tolkning som bakgrunn for nye forskningsspørsmål
- Kvalitativ forskning er som en læringsprosess. Nye aspekter dukker opp ettersom forskningen går fremover. Forskningsspørsmål kan endres etter som forsker lærer hva og hvem som bør spørres. Metoder for data-innsamling kan forandres ettersom nye veier pekes ut

5.1.1.1 Case studie

En case studie av informasjonssystemer foregår ved dybdeundersøkelse på bestemte områder. Dette kan være et bestemt program, en hendelse, et spesifikt prosjekt, en prosess eller undersøkelse av en eller flere individer. Forskerne samler detaljert informasjon ved å bruke en variasjon av datainnsamlings-metoder innenfor en bestemt tidsramme [Creswell 2003]. Studiet av DES er en multipell case studie fordi fire leverandører utvikler systemet.

I enkle case studier gjelder ofte resultatet kun for det enkelte tilfellet. Dette kan gjøre resultater fra case studier vanskelig å generalisere. Med generalisering menes å sammenfatte resultater til en generell teori [Yin 1994]. Det at DES er en multippel case studie styrker en fremtidig generalisering.

5.1.1.2 Metoder i kvalitativ forskning

Tre hovedmetoder ligger til grunn i kvalitativ forskning. Den første er *observasjon*. Observasjon er den eldste og mest grunnleggende metoden i

kvalitativ forskning [Adler og Adler 1994]. Generelt brukes observasjon for å observere oppførsel og aktivitet hos individer i et forskningsfelt. Målet med observasjon er å identifisere og eventuelt konkludere med nye trender eller mønstre [Silverman 2001]. Den neste metoden er *intervju*. I kvalitativ forskning er intervju viktig for å hente informasjon som ikke kan sees ved direkte observasjon. Dette gjelder informasjon som en persons følelser, tanker, intensjoner eller hendelser i fortiden. Intervju kan også benyttes for å få klarhet i eventuelle uklare aspekter. Målet med intervjuer er å entre andre menneskers perspektiver [Quinn 1987]. Den siste metoden innenfor kvalitativ forskning er *tekst- og dokumentanalyse*. Tekst- og dokumentanalyse kan brukes både som bakgrunnsmateriale for forskning og til å samle kvalitative data. Dette gjelder blant annet studie av filer, offisielle dokumenter, e-post, litteratur og mange andre typer tekster [Silverman 2001]. Transkripsjon, foto, lyd og video er ”*ypperlige opptak av naturlig forekommende interaksjon*” [Silverman 2001, s. 13]. Med denne type dokumentasjon kan kvalitative forskere lett gå tilbake til viktige hendelser i forskningen [Silverman 2001].

Data relevant for denne studien er samlet inn ved hjelp av intervjuer, timelister og kodeanalyse. Jeg har hatt tilgang til dokumenter fra dette og har derfor analysert disse. I den forbindelse skriver jeg mer om dokumentanalyse i avsnitt 5.3.

5.1.2 Kvantitativ forskning

Kvantitativ forsking har den hensikt å forme informasjon om til målbare enheter. Slik blir det mulig å gjennomføre regneoperasjoner på resultatene. Dette gjelder blant annet å finne gjennomsnitt ellet prosenter. Creswell (2003) beskriver to metoder for å gjøre kvalitativ forskning. Det ene er kartlegging. Kartlegging er en kvantitativ eller numerisk beskrivelse av trender, holdninger eller meninger. Disse tilhører en forespurt del av en populasjon. Fra resultatene kan en forsker gjøre generaliseringer. Den andre metoden er *eksperiment*. Her er også intensjonen å gjøre generaliseringer for en populasjon. Et eksempel på et eksperiment er testing av hvordan en type medisin virker på en forhåndsdefinert folkegruppe [Creswell 2003, s. 153 - 154].

5.2 Datainnsamling

5.2.1 Dokumenter

Som kapittel 2 forklarer, startet SE ved SRL våren 2003 et forskningsprosjekt. Forskningen skulle studere utvikling av en webbasert forskningsdatabase kalt DES. I den forbindelse utviklet SE kravspesifikasjonen for DES. Den utgjør grunnlaget for forståelsen samt grunnlaget for estimering av størrelsen til DES.

Under utviklingen av DES gjennomførte forskerne i SE ved SRL intervjuer med utviklerne. Transkripsjonen av intervjuene danner grunnlaget for vurderingen av omgivelsesfaktorene i UCP-metoden, COCOMO II og WEBMO.

Hver leverandør registrerte timebruk per use case i utviklingen. Dette er data samlet i et Excel regneark etter instruksjon fra SE. Regnearket danner grunnlaget for faktisk brukt tid i sammenlikningen av leverandørenes estimater og estimeringsmodellenes estimater. Denne sammenlikningen utgjør den kvantitative delen av studiet.

En forsker ved SE og en ekstern konsulent har gjennomført hver sin generelle evaluering av leverandørenes kodekvalitet. Disse evalueringene ligger til grunn i dette studiets diskusjon om estimeringsmodellenes antagelser angående kvalitet.

5.2.2 Litteratur

I tillegg til ovennevnte dokumenter benyttes teori for å danne grunnlegende kunnskap om forskningsområdet samt gi basis for sammenlikning av resultater [Walsham 2002]. Jeg begynte denne case studien ved å studere tidligere forskning på estimering ved hjelp av use case. Anda et al. (2001, 2002) og Anda (2002) introduserte UCP-metoden som en alternativ estimeringsmetode for meg. Gjennom disse studiene ble jeg også introdusert til tips om problemstillinger og fremtidig arbeid. I tillegg til Anda's forskning kjente jeg til noe relevant litteratur fra tidligere kurs innen datateknologi. Dette gjelder teori fra blant annet Mathiassen et al. (2000), Valacich et al. (2001), Hansen (2002), Hjertø (2002), Larman (2002) og Krutchen (2003). Bibliografiene i nevnte referanser inneholder noe relevant litteratur som jeg også benyttet meg av. For å få tak i denne litteraturen brukte jeg det webbaserte biblioteksystemet BIBSYS. Her fant jeg blant annet Treble og Douglas (1995), Boehm et al. (2000c) og Garmus og Herron (2004).

Internet med utgangspunkt i det webbaserte informatikkbiblioteket ved Universitet i Oslo ble også brukt til å søke opp flere artikler. Ved søk i ACM[2] brukte jeg blant annet følgende søkeord; "cocomo", "internet estimation", "web estimation", "software estimation" og "function point analysis". Dette hjalp meg å finne flere artikler, blant annet; Boehm et al. (1998), Boehm et al. (2000b) og Reifer (2000). I tillegg brukte jeg søkemaskiner som Google[3]. Et søk på uttrykket "use case poeng" resulterte i omtrent 1000 svar. Et søk med uttrykket "function point analysis" ga også stor respons. Dette hjalp meg å finne websider som http://www.omg.org og http://www.ifpug.com/. Disse inneholder flere artikler om estimeringsmodeller, UML og use case, blant annet; Longstreet (2003), OMG (2003) og Longstreet (2004a) og Longstreet (2004b).

Når en bruker Internet som kilde for forskning må visse forhåndsregler tas. På Internet kan hva som helst publiseres uten restriksjoner. For meg er det derfor viktig at websiden jeg henter kilder fra er oppriktig. Dette kan jeg vurdere ved å stille meg selv fire ulike spørsmål om websiden:

- Når ble websiden sist oppdatert?
- Er websiden sponset og hvem er eventuelle sponsorer?
- Hvordan ender URL'en? .com, .org, .gov etc.
- Hvem snakker gjennom websiden, en kjent organisasjon eller en privat-person?

Ved å vurdere disse elementene kan jeg finne ut om websiden er en sikker kilde som er verdt å bruke som referanse.

5.3 Dokumentanalyse

Forskningsmetoden i dette case studiet er litteratur- og dokumentanalyse. Min oppgave er å studere dokumenter som inneholder data innsamlet av SE ved SRL. Alle dokumenter inneholder mange kvalitative data, mer enn det som er

[2] ACM er et digitalt bibliotek med en fulltekst samling av all litteratur publisert siden 1952, http://www.acm.com
[3] http://www.google.com

nødvendig for denne studien. Det er en utfordring å sortere dataene riktig for å finne den informasjonen som er relevant i dette tilfellet.

5.3.1 Kravspesifikasjon og tilbud

Etter å ha studert tidligere forskning, lest bøker og tilegnet meg nok kunnskap om forskningsfeltet kunne jeg sette i gang dokumentanalysen. Første punkt for alle estimeringsmodellene er å finne systemets størrelse. Til dette gjorde jeg en grundig analyse av kravspesifikasjonen. Dette hjalp meg å forstå ønskene SE som fremtidig bruker, hadde for DES. Leverandørenes tilbud hjalp meg å forstå hvordan hver enkelt leverandør ønsket å lage DES. For å skille ut nyttig informasjon, ble hvert dokument gjennomgått flere ganger for hver estimeringsmodell. Ved å lage notater sorterte jeg ut informasjon som tilhørte de ulike estimeringsmodellene. Slik kunne denne informasjonen overføres til et selvlaget regneark der størrelsesmålene ble beregnet.

5.3.2 Timeføring

Regnearket i Excel der leverandørene hadde ført timer sorterte jeg på følgende måter:

- Tidsbruk totalt per leverandør
- Tidsbruk per fase per leverandør
- Tidsbruk per use case per leverandør

Slik lå timene til grunn for sammenlikning med estimeringsmetodenes og ekspertenes estimater.

5.3.3 Evaluering av kodekvalitet

Evalueringen av leverandørenes kodekvalitet ble gjort av to uavhengige personer. Dette gjør resultatene ev evalueringen tydelig subjektive. Allikevel brukes evalueringene som grunnlag i vurderingen av estimeringsmodellenes antagelser om kvalitet.

5.3.4 Intervjuer

Intervjutranskripsjonene ble brukt som bakgrunn for å gjøre riktigst mulig vurderinger av estimeringsmodellenes kostnadsdrivere. De innehold mye mer informasjon en nødvendig. For å sile ut den informasjonen estimeringsmodellene krever, gikk jeg igjennom hvert intervju flere ganger. Jeg laget et dokument der jeg noterte sitater og beskrivelser som passet til hver kostnadsdriver. Ut fra helheten i notatene vurderte jeg kostnadsdrivernes nivå.

En intervjutranskripsjon gir intervjuerens oppfattelse av intervjuobjektets mening. En rask feil å gjøre er å vurdere det intervjuobjektet sier som sannheten. En annen ting å være klar over er tvetydigheten i svarene et intervjuobjekt kan gi [Silverman 2001]. Dette er viktig å være klar over idet vurderingen av kostnadsdriverne gjøres.

5.4 Oppsummering

Studiet denne rapporten beskriver er en multippel case studie. For uten antall timer og timeestimater er alle data er kvalitative data. Fire svært ulike leverandører har utviklet samme system i parallell. Siden utviklingen var gjennomført da jeg startet på min masteroppgave, baseres dette studiet på dokumentanalyse av prosjektdokumentasjon. Følgende dokumentasjon ligger til grunn for studiet;

- Kravspesifikasjon
- Tilbudsdokumenter
- Excel regneark med timeføring
- Intervjuertranskripsjonen
- Evaluering av kodekvalitet

6 ANALYSE OG RESULTATER

Dette kapittelet begynner med en beskrivelse leverandørene. Her presenteres ekspertenes estimater sammen med faktisk brukt tid. Videre presenteres estimatene for DES laget med UCP-metoden. Deretter følger UFP, COCOMO II og til slutt WEBMO. Det nest siste avsnittet tar for seg avvik mellom estimater og faktisk brukt tid. Til slutt gis en kort oppsummering av estimatene laget ved de ulike metodene i forhold til ekspertenes egne estimater og faktisk brukt tid.

6.1 Kontekst

6.1.1 Leverandør A

Leverandør A er en relativt stor bedrift med nesten 100 ansatte lokalisert i Norge. De leverer et bredt spekter av tjenester i fem hovedområder; strategi og organisasjon, kundehåndteringssystemer, portalløsninger, applikasjonsforvaltning og plattformutvikling og integrasjon. Kundeporteføljen består blant annet av en rekke offentlige virksomheter samt en stor del av Norges største bedrifter.

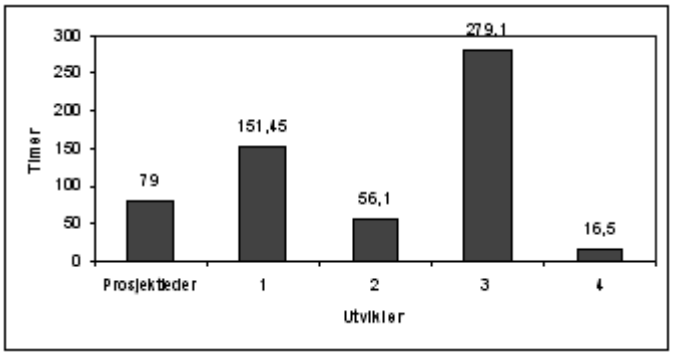
I sitt tilbud har leverandør A laget en grundig fremstilling av brukerprosessene i form av storyboards. Der beskrives hvilke elementer som skal være med i hvilke brukerprosesser. De mer komplekse prosessene forklares ved hjelp av illustrasjoner. Leverandør A valgte å bygge den tekniske arkitekturen på den allerede eksisterende plattformen hos SE.

I første omgang ble to utviklere og en prosjektleder stilt til disposisjon i utviklingen. I løpet av prosjektperioden har til sammen fem utviklere vært innblandet. Utviklerne delte på fire roller; prosjektleder, grensesnittutvikler, databaseutvikler og interaksjonsdesigner. Leverandør A benyttet en egenutviklet utviklingsmodell som er inspirert av Rational Unified Process (RUP) [Krutchen 2003] og eXtreme Programming (XP). Som eneste leverandør har ikke leverandør A oppgitt estimert timeantall. Med en fastpris på 160 000 NOK og en timepris på 800 NOK, antas her at leverandør A estimerer med omtrent 250 timer.

Estimert timeantall	Faktisk brukt timeantall
250	582,15
Estimert antall uker	Faktisk brukt antall uker
4,4	13

Tabell 6-1 Ekspertenes estimater i forhold til faktisk brukt tid hos leverandør A

Forholdet mellom faktisk brukt timeantall og faktisk brukt antall uker i Tabell 6-1, viser at utviklerne ikke jobbet full tid med DES. Leverandør A brukte 332 timer mer enn planlagt. Totalt gikk 13 uker med på fullføringen av DES. For utviklerne fordelte timeantallet seg som vist i Figur 6-1.



Figur 6-1 Antall timer totalt per utvikler hos leverandør A

6.1.2 Leverandør B

Leverandør B er en mindre skandinavisk bedrift med 16 ansatte Norge og totalt 35 ansatte. Leverandør B utfører tjenester med hovedvekt på strategirådgivning kombinert med utvikling av IT/Internett-løsninger. Kundeporteføljen består av offentlige virksomheter og noen større norske selskaper.

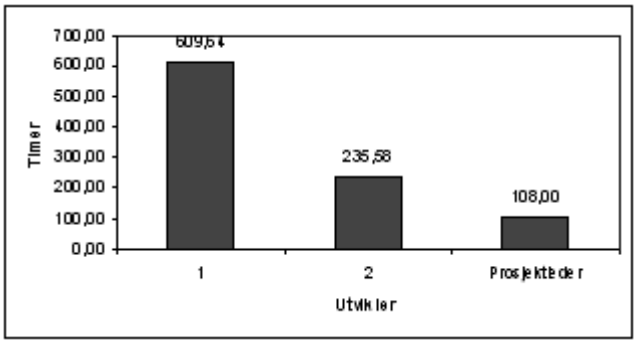
Leverandør B ga i sitt tilbud en kort, skriftlig beskrivelse av løsningen på DES. Dette inkluderer elementer til brukergrensesnitt. Vedlagt er også en oversikt over mulige databaseelementer. Ønsket teknologi er ikke beskrevet.

I første omgang ble to utviklere og en prosjektleder stilt til disposisjon. Noe reorganisering ble gjort før arbeidet satte i gang. Totalt har tre utviklere vært med på utviklingen. Følgende roller ble fordelt; informasjonsarkitekt, senior systemingeniør, og prosjektleder og kundeansvarlig. Som utviklingsmodell fulgte leverandør B en evolusjonær inkrementell utviklingsprosess. Hovedideen med modellen er gradvis å utvikle del leveranser av systemet. Leveransene skal evalueres av kunden. Dersom kunden ikke er fornøyd utvikles en ny leveranse med utvidet funksjonalitet eller andre endringer. Hovedformålet med denne utviklingsmodellen er enkelt å kunne håndtere risikoer som; uklarheter og endringer i kravspesifikasjon og implementering av ny teknologi.

Estimert timeantall	Faktisk brukt timeantall
341	953,22
Estimert antall uker	Faktisk brukt antall uker
9	14

Tabell 6-2 Ekspertenes estimater i forhold til faktisk brukt tid hos leverandør B

Leverandør B planla en utviklingstid på 9 uker. Det totale timeantallet ble estimert til 341, med et innregnet avvik på 13 timer. Dette estimatet ble funnet ved hjelp av en tilpasset versjon av den generelle estimeringsmetoden beskrevet i avsnitt 3.2. To eksperter estimerte beste og verste tilfelle for så å finne det reelle timeantallet. Timeantallet ble brutt ned i faser og aktiviteter tilpasset utviklingsmodellen. Tabell 6-2 viser ekspertenes estimater i forhold til faktisk brukt tid. Tabellen viser en kraftig overskridelse på omtrent 600 timer. For de tre utviklerne har timeantallet fordelt seg som vist i Figur 6-2.



Figur 6-2 Antall timer totalt per utviker hos leverandør B

6.1.3 Leverandør C

Leverandør C er en relativt ny og liten bedrift med 6 ansatte. Ved behov leies freelancere, tekniske og grafiske fagfolk. Leverandør C er ikke lokalisert samme by som SRL. Arbeidsområdene er utvikling av løsninger med hovedvekt på rapportering, prosjektstyring og webpublisering. Kundeporteføljen til leverandør C består av en offentlig virksomhet og flere private bedrifter.

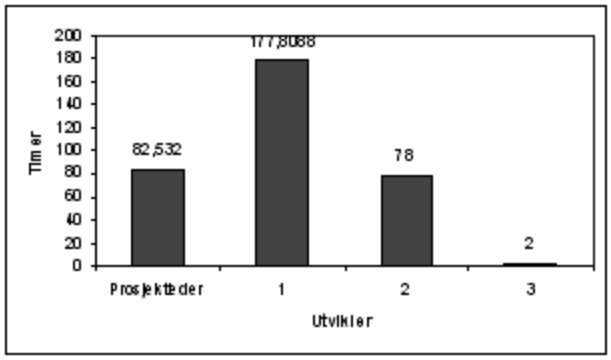
Leverandør C ga en detaljert og punktvis oversikt over use case'ene i DES. For hvert use case forklares eventuelle elementer i brukergrensesnittet og andre praktiske opplysninger. Noen nye use case er også lagt til. Leverandør C laget også en datamodell i et tabellarisk oppsett og valgte å benytte samme teknologi som Simulaweb. I løpet av utviklingen endret leverandør C programmeringsspråk til Java og JSP.

I utgangspunktet ønsket leverandør C å stille en utvikler og en prosjektleder til disposisjon. SE ønsket imidlertid at to utviklere skulle delta i tillegg til prosjektleder. Mot slutten av utviklingen hjalp også en fjerde utvikler til. Planen var at arbeidet med DES skulle foregå innimellom andre prosjekter. Det ble ikke fulgt noen spesifikk utviklingsprosess i gjennomføringen. Leverandør C jobbet som vanlig, tilsvarende XP og agile prosesser. Målet var å raskt lage kjørende versjoner av DES. Slik kan tilbakemeldinger mottas raskt.

Estimert timeantall	Faktisk brukt timeantall
100	345,39
Estimert antall uker	Faktisk brukt antall uker
3,6	12

Tabell 6-3 Ekspertenes estimater i forhold til faktisk brukt tid hos leverandør C

Leverandør C planla en utviklingstid på 3,6 uker. 100 timer ble estimert til bruk på utviklingen. Prosjektlederen for leverandør C har selv skrevet tilbudet og laget estimatene. Estimatene ble laget ved å bryte systemet ned i brukerprosesser. Basert på erfaring er tid per brukerprosess estimert. Andre generelle faktorer som prosjektledelse ble også vurdert. Dette ble beregnet som en tilleggsfaktor på 12 %. Tabell 6-3 viser estimert tid sammen med faktisk brukt tid. Figur 6-3 viser hvordan timeantallet har fordelt seg på de utviklerne.



Figur 6-3 Antall timer totalt per utviker hos leverandør C

6.1.4 Leverandør D

Leverandør D er et større internasjonalt selskap med 13 000 ansatte i mer enn 20 land. De har eksperter innenfor flere fagområder, blant annet bank, økonomi, helse, offentlig sektor, skog, industri, energi, handel og logistikk. Innenfor disse bransjene utfører leverandør D konsulenttjenester i utvikling, drift og forvaltning av systemer.

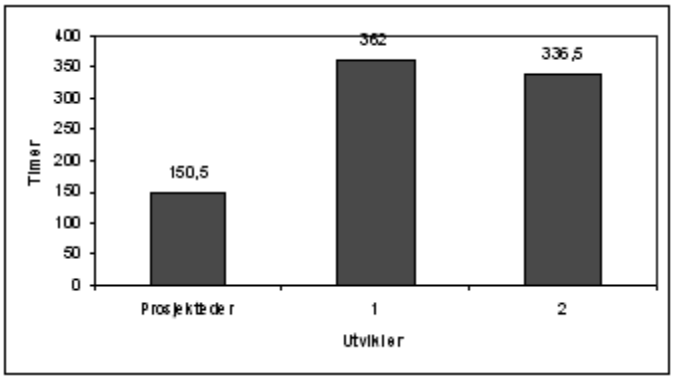
Leverandør D laget en tekstlig løsning på DES. Løsningen var oppdelt i tre hovedområder; liste opp og se detaljer angående studier, administrasjon av studier og administrasjon av brukere. Løsningen inkluderte elementer som tilhørte brukergrensesnittet. Det ble ikke forklart hvordan funksjonene skulle løses programmeringsmessig. Leverandør D beskrev også grundig databasens utseende. Den tekniske løsningen baserte seg på JSP og JavaBeans, samt MVC for å skille mellom brukergrensesnitt og forretningslogikk.

Leverandør D stilte to utviklere og en prosjektleder til disposisjon som deltok gjennom hele utviklingen. Leverandør D brukte et egenutviklet rammeverk som delvis ble fulgt. Dette rammeverket omhandler formell bruk av ulike UML tekniker og er en dialekt av RUP, bare ikke fult så omfattende.

Estimert timeantall	Faktisk brukt timeantall
650	849
Estimert antall uker	Faktisk brukt antall uker
9,8	10

Tabell 6-4 Ekspertenes estimater i forhold til faktisk brukt tid hos leverandør D

Leverandør D planla en utviklingstid på 9,8 uker. Det estimerte timeantallet var 650 timer. For å komme frem til estimatene benyttet leverandør D en egenutviklet estimeringsmodell for objektorientert analyse og Java. Antall enkle, middels og vanskelige skjermbilder ble identifisert sammen men flere vurderinger. Vurderingene ble notert i et regneark som beregnet estimatene. Utviklerne hos leverandør D syntes estimatene fra regnearket var noe høye. På bakgrunn av personlige erfaringer justerte de ned estimatet. Tabell 6-4 viser estimert tid og faktisk brukt tid. Figur 6-3 viser hvordan timeantallet har fordelt seg på de forskjellige utviklerne.



Figur 6-4 Antall timer totalt per utviker hos leverandør D

6.2 Estimering med UCP

6.2.1 Kategorisering av Aktører

Kravspesifikasjonen identifiserer en enkel aktør og tre komplekse aktører. Den enkle aktøren er Simulaweb. Dette er det eksisterende systemet som DES integreres i. De tre komplekse aktørene er alle menneskelige aktører som opererer gjennom et grafisk brukergrensesnitt. Disse er:

- Databaseadministrator
- Studieadministrator
- Gjest

I den forbindelse beregnes ujustert aktørvekt (UAW) følgende:

UAW = (1 enkel aktør * 1) + (3 komplekse aktører * 3) = 10

6.2.2 Kategorisering av use case

To typer estimater for DES beregnes per leverandør. Det ene estimatet beregnes ut fra de opprinnelige use case'ene i kravspesifikasjonen. Det andre estimatet inkluderer alle use case leverandørene har laget i tillegg til use case'ene i kravspesifikasjonen.

DES består av relativt enkle funksjoner. Ingen use case er inkluderte eller utvidede use case. Kravspesifikasjonen inneholder 2 gjennomsnittlige use case og 7 enkle use case. Følgende gjelder for alle fire leverandører:

$UUCW_1$ = (2 gjennomsnittlige use case * 10) + (7 enkle use case * 5) = 55

For å finne estimatet der leverandørenes egne use case inkluderes, studeres leverandørene hver for seg.

6.2.2.1 Leverandør A

Leverandør A har laget et use case i tillegg til use case'ene i kravspesifikasjonen. Siden de i tilbudet ikke har detaljert dette use case'et, vurderes det å være et enkelt use case. Dette på grunn av DES sin generelle lave kompleksitet samt at tilleggs use case'et ikke utgjør hovedfunksjonalitet.

$UUCW_2$ = (2 gjennomsnittlige use case * 10) + (8 enkle use case * 5) = 60

6.2.2.2 Leverandør B

Leverandør B har laget 7 use case i tillegg til use case'ene i kravspesifikasjonen. Ett av disse er logg inn. Innlogging er en forutsetning for å bruke deler av DES og vurderes ikke som et eget use case. De andre use case'ene er ikke spesifisert ut over navn. De 6 tilleggs use case'ene vurderes å være enkle.

$UUCW_2$ = (2 gjennomsnittlige use case * 10) + (13 enkle use case * 5) = 85

6.2.2.3 Leverandør C

Leverandør C har laget 4 use case i tillegg til use case'ene i kravspesifikasjonen. Som eneste leverandør har de laget en detaljert spesifikasjon av alle trinnene i use case'ene. Et use case er logg inn. Innlogging er en forutsetning for å bruke deler av DES og vurderes ikke som et eget use case. De resterende 3 use case er alle enkle.

$UUCW_2$ = (2 gjennomsnittlige use case * 10) + (10 enkle use case * 5) = 70

6.2.2.4 Leverandør D

Leverandør D har laget et use case i tillegg til use case'ene i kravspesifikasjonen. Use case'et et ikke spesifisert og vurderes å være et enkelt use case.

$UUCW_2$ = (2 gjennomsnittlige use case * 10) + (8 enkle use case * 5) = 60

6.2.3 Tekniske Faktorer i UCP

Leverandørene utviklet DES med grunnlag i samme kravspesifikasjon. Dette gjør de fleste tekniske faktorene er like. Tabell 6-5 viser vurderingen av de tekniske faktorene. Hver vurdering begrunnes nedenfor. Tabell 6-6 har satt opp leverandørenes TFactor og TCF.

Nummer	Beskrivelse	Vekt	Vurdering	Beregnet vurdering
T1	Distribuert system	2	1	2
T2	Responstid	1	1	1
T3	Sluttbrukereffektivitet	1	4	4
T4	Kompleks prosessering	1	2	2
T5	Gjenbrukbar kode	1	A = 3 C = 1 B, D = 4	3 1 4
T6	Enkelt å installere	0,5	A, B, D = 2 C = 5	1 2,5
T7	Brukbarhet	0,5	4	2
T8	Flyttbarhet	2	1	2
T9	Enkelt å forandre	1	A, C = 3 B, D = 4	3 4
T10	Flerbruk	1	3	3
T11	Sikkerhetsregler	1	3	3
T12	Tilby tilgang til tredje parter	1	0	0
T13	Brukeropplæring	1	1	1

Tabell 6-5 Vurdering av tekniske faktorer for DES

Leverandør	TFactor	TCF
A	25	0,85
B	29	0,89
C	26,5	0,865
D	29	0,89

Tabell 6-6 Leverandørenes TFactor og TCF

6.2.3.1 T1: Distribuert system

DES er et webbasert system. Tjener prosesserer all informasjon som ligger lagret i en database. Verken tjener eller database er distribuert. Klientene er tynne klienter i form av nettleser. Distribuert system utgjør lav påvirkning på utviklingen.

6.2.3.2 T2: Responstid

Responstiden har lite å si for DES, men bør være lav på tjeneren. Kommunikasjon mot nettleserne går over intranett og Internet, http, TCP/IP. Hastigheten der kan ikke økes spesifikt ved gode programmeringsløsninger på tjener. Responstid vurderes til å ha lav påvirkning på utviklingen.

6.2.3.3 T3: Sluttbrukereffektivitet

Effektiviteten for sluttbrukere avhenger av koblingen mellom sidene i brukergrensesnittet. Det er viktig at det er få klikk med musa for å komme til ønsket side. I et enkelt søkeprogram som DES er, bør sluttbrukereffektiviteten være god for at sidene blir brukt. Sluttbrukereffektivitet vurderes til å ha litt over gjennomsnittlig betydning for utviklingen av DES.

6.2.3.4 T4: Kompleks prosessering

DES består hovedsakelig av enkle opprett, slett, finn og endre funksjoner. To funksjoner krever litt annen type prosessering. Dette gjelder; rapporter aggregert studier og eksporter studier til en kommaseparert fil. Kompleks prosessering vurderes til å ha litt under gjennomsnittlig påvirkning på utviklingen.

6.2.3.5 T5: Gjenbrukbar kode

Kravspesifikasjonen presiserer at koden skal være lett forståelig slik at en systemutvikler enkelt kan vedlikehold den. Leverandørene virker å ha ulik innstilling på dette punktet. Leverandør C gir utrykk for at det er bedre å bli raskt ferdig og utvikle DES på en enkel måte. Leverandør B og D gir uttrykk for at de gjør et virkelig prosjekt ut av DES, og at det er viktig å bruke tid på riktig bruk av mønstre og komponenter. Leverandør A gir inntrykk av å gjøre litt mindre ut av DES enn B og D, men mer enn leverandør C. For leverandør A vurderes gjenbrukbarhet å ha gjennomsnittlig påvirkning. For leverandør C vurderes gjenbrukbarhet til å ha lav påvirkning. For leverandør B og D vurderes gjenbrukbarhet til å ha litt over gjennomsnittlig påvirkning.

6.2.3.6 T6: Enkelt å installere

DES består av ett sett .php, .jsp filer og/eller Javakode med en kobling mot en database. Leverandørene vil også selv hjelpe til med installasjonen. DES skal

integreres i Simulaweb. Allikevel utgjør installasjon litt under gjennomsnittlig påvirkning på utviklingen.

Leverandør C er lokalisert i en annen by enn SRL. Derfor må DES leveres slik at innstalleringen er enkel. Leverandør C installerer vanligvis selv. Dette har større betydning for utviklingen hos leverandør C enn hos de andre leverandørene. I referat fra intervjuene med leverandør C uttales følgende; *"…på de prosjektene han har vært på har de installert selv. Det er greiere å ordne feil selv. Spesielt på en "one of a kind" web-server applikasjon så vil det alltid kreve mye av de som installerer…"*, *"…I det øyeblikket andre skal installere må alt være perfekt..."*

6.2.3.7 T7: Brukbarhet

Brukbarhet vektlegges høyt. Målet med DES er å lage en database som tar vare på studier og publikasjoner, og gjøre det enklere å søke i og vedlikeholde denne informasjonen. Det skal ikke lages noen hjelp funksjon og brukergrensesnittet må være selvforklarende. Brukbarhet vurderes til å ha litt over gjennomsnittlig påvirkning på utviklingen.

6.2.3.8 T8: Flyttbarhet

Det er et krav at DES skal kjøre på Simulaweb sin eksisterende plattform. Flytting mellom plattformer er ikke planlagt. Flyttbarhet vurderes til å ha litt under gjennomsnittlig påvirkning på utviklingen.

6.2.3.9 T9: Enkelt å forandre

Forandring er også et punkt der leverandørene gir uttrykk for forskjellig innstilling. Som nevnt under 6.2.3.5 presiserer kravspesifikasjonen betydningen i god kodedokumentasjon. Leverandør A og C ønsker å dokumentere koden bra og vurderes til gjennomsnittlig. For leverandør B og D vurderes denne faktoren til litt over gjennomsnittlig.

6.2.3.10 T10: Flerbruk

Et webbasert system er et system der det er meningen at flere bruker systemet samtidig. Når det gjelder Database- og Studieadministrators rettigheter ved endringer og oppdateringer er det viktig at det skjer synkronisert. Flerbruk vurderes til å ha gjennomsnittlig påvirkning på utviklingen.

6.2.3.11 T11: Sikkerhetsregler

Administratorer må logge inn via Simulaweb sin sikre innlogging (HTTPS). Denne tjenesten er allerede fungerende i Simulaweb. Siden DES integreres i Simulaweb antas sikkerhetsaspektene å være håndtert. Dette gjelder oppstartsrutiner, sikkerhetskopiering og gjenoppretting. Håndtering av spesielle sikkerhetsregler vurderes til å ha gjennomsnittlig påvirkning på utviklingen.

6.2.3.12 T12: Tilby tilgang til tredje part

DES skal ikke ha tilgang eller kommunikasjon til andre parter. DES skal heller ikke tilby biblioteker eller tjenester til eksterne systemer. DES skal kun integreres med Simulaweb. Tilby direkte tilgang til tredje part vurderes å ha ingen påvirkning på utviklingen.

6.2.3.13 T13: Brukeropplæring

Ingen spesielle fasiliteter for brukeropplæring er nødvendig. Kravspesifikasjonen spesifiserer et selvforklarende system. Utredende brukeropplæring skal ikke være nødvendig. For systemadministrator planlegges en rask gjennomgang av systemet. Brukeropplæring vurderes å ha lav påvirkning på utviklingen.

6.2.4 Omgivelsesfaktorer

Leverandørene er alle ulike og selvstendige. Tabell 6-7 viser omgivelsesfaktorene med vekt, vurdering og beregnet vurdering fra leverandørenes utviklingsmiljø. Faktorene er vurdert med bakgrunn i intervjuene gjennomført. Vurderingene begrunnes nedenfor. Tabell 6-8 viser leverandørenes EFactor og EF.

Nummer	Beskrivelse	Vekt	Vurdering	Beregnet Vurdering
M1	Kjennskap til utviklingsprosess	1,5	A = 4 B = 3 C = 2 D = 3	A = 6 B = 7,5 C = 3 D = 4,5
M2	Applikasjonserfaring	0,5	A = 4 B = 3 C = 4 D = 4	A = 2 B = 2 C = 2 D = 2
M3	Objektorientert erfaring	1	A = 3 B = 4 C = 3 D = 2	A = 3 B = 4 C = 3 D = 2
M4	Analysekompetanse	0,5	A = 3 B = 4 C = 3 D = 3	A = 1,5 B = 2 C = 1,5 D = 1,5
M5	Motivasjon	1	A = 2 B = 3 C = 4 D = 4	A = 2 B = 3 C = 4 D = 4
M6	Stabile krav	2	A = 5 B = 5 C = 5 D = 5	A = 10 B = 10 C = 10 D = 10
M7	Deltids arbeid	-1	A = 4 B = 4 C = 5 D = 4	A = -4 B = -3 C = -5 D = -4
M8	Vansker med programmeringsspråk	-1	A = 3 B = 3 C = 3 D = 3	A = -3 B = -2 C = -3 D = -3

Tabell 6-7 Vurdering av omgivelsesfaktorer for DES

Leverandør	EFactor	EF
A	17,5	0,875
B	18	0,86
C	15,5	0,935
D	17	0,89

Tabell 6-8 Leverandørenes EFactor og EF

6.2.4.1 M1: Kjennskap til utviklingsprosess

Leverandør A bruker en egenutviklet utviklingsprosess. Utviklerne har hatt introduksjonskurs i bruk av utviklingsmodellen, og bruker den på ulike måter i arbeidet. En av utviklerne mener at han ikke har så mye erfaring med prosessen. Han følger vanligvis den prosessen kundene ønsker. Generelt vurderes utviklingsteamet å ha litt over gjennomsnittlig erfaring med utviklingsprosessen.

Leverandør B bruker en egenutviklet, evolusjonær og inkrementell utviklingsprosess. De har valgt å legge mer vekt på metodikken enn vanlig. *"Metodikken blir mer vektlagt i dette prosjektet enn det som er vanlig. Siden den er en fremtredende del av prosjektet blir den også prioritert mer og det blir satt av mer tid til forarbeidet…"*. Siden det i dette prosjektet planlegges å bruke mer tid på utviklingsmodellen vurderes teamet å ha gjennomsnittlig erfaring med utviklingsmodellen.

Leverandør C nevner ingen utviklingsprosess i tilbudet. Gjennom intervjuene gis det inntrykk av at de ikke følger noen spesifikk utviklingsprosess, noe som gir litt rot i utviklingen. *"…Det som best beskriver deres prosess er mangel på prosess…"*. Denne faktoren vurderes til under gjennomsnittlig kjennskap til utviklingsprosess.

Leverandør D nevner ikke deres utviklingsprosess i fastpristilbudet. Derimot refererer de til et rammeverk i intervjuene. Dette er et rammeverk utviklerne er kurset i. Selv mener utviklerne at de ikke kjenner rammeverket særlig godt. Prosjektleder mener det sitter implisitt i deres tankegang. Kjennskap til utviklingsprosessen vurderes til gjennomsnittlig.

6.2.4.2 M2: Applikasjonserfaring

Utviklerne hos leverandør A har alle vært med i utvikling av tilsvarende applikasjoner tidligere. Erfaring med applikasjon vurderes til litt over gjennomsnittlig.

Utviklerne hos leverandør B mener generelt at den svenske delen av firmaet har bedre erfaring med flere prosjekter liknende DES. Selve løsningen på DES er vanlig for utviklerne. Applikasjonserfaring vurderes til gjennomsnittlig.

Leverandør C gir uttrykk for å ha god erfaring fra utvikling av denne type applikasjoner. Utviklerne mener at DES er et typisk prosjekt som de er kjent med. Applikasjonserfaring vurderes til litt over gjennomsnittlig.

For utviklerne hos leverandør D er DES et typisk utviklingsprosjekt. To av utviklerne har jobbet mye med utvikling av liknende applikasjoner. Eneste forskjell er størrelsen fordi DES betegnes som en liten applikasjon. Den tredje

utvikleren har kun vært med på utvikling av en liknende applikasjon tidligere. Generelt vurderes applikasjonserfaring til gjennomsnittlig.

6.2.4.3 M3: Objektorientert erfaring

En av utviklerne hos leverandør A dokumenterer arbeid med objektorientering i forbindelse med tidligere utvikling. To av utviklerne dokumenterer ikke arbeid med objektorientering. Den fjerde utvikleren har kun erfaring med objektorientering i forbindelse med programmering i Java. Utviklerne vurderes generelt å ha gjennomsnittlig erfaring med objektorientering.

Utviklerne hos leverandør B har generelt mye erfaring med objektorientering. *"All design han har vært med på har vært objektorientert… …Han har en del erfaring med det, i og med at det er en naturlig del av jobben å begynne med analysedelen."* Utviklerne hos leverandør B vurderes å ha over gjennomsnittlig erfaring med objektorientering.

Utviklerne hos leverandør C gir generelt inntrykk av å ha jobbet mye med objektorientering gjennom objektorienterte programmeringsspråk. Kun en av utviklerne programmerer uten å bruke spesifikke objektorienterte metoder. Erfaring med objektorientering vurderes til gjennomsnittlig.

En av utviklerne hos leverandør D har en del erfaring med objektorientering. De to andre har ikke noe særlig erfaring ut over universitetet. Erfaring med objektorientering vurderes til litt under gjennomsnittlig.

6.2.4.4 M4: Analysekompetanse

Utviklerne hos leverandør A gir inntrykk av å ha en del erfaring i programvareutvikling, både når det gjelder analyse og implementering. Noen har drevet mer med analyse enn andre. Analysekompetanse vurderes til gjennomsnittlig.

Utviklerne hos leverandør B har jobbet mye med objektorientert analyse og design. Analysekompetanse vurderes til litt over gjennomsnittlig.

Intervjuene forteller ikke spesielt om analysekompetansen til leverandør C. Utviklerne har også jobbet med utvikling og objektorientering i flere år. Analysekompetanse vurderes til gjennomsnittlig.

Hos leverandør D har utviklerne gjennomsnittlig erfaring med denne type applikasjon, og under gjennomsnittlig erfaring med objektorientering. Derimot har alle jobbet med utviklingsprosjekter tidligere. Utviklingsmetodikken beskriver også konkret analyse og design fasen i utviklingsprosjekter. I utviklingen av DES følges utviklingsmodellen litt mer enn vanlig fra begynnelsen av. Analysekapabiliteten vurderes dermed til gjennomsnittlig.

6.2.4.5 M5: Motivasjon

En av utviklerne hos leverandør A gir inntrykk av å ha lav motivasjon for prosjektet. Dette fordi DES er et lite prosjekt. Utvikleren mener alle andre oppgaver er viktigere enn arbeidet med DES. De andre utviklerne mener at de må være mer obs i dette prosjektet siden det er et forskningsprosjekt. Motivasjon vurderes til litt under gjennomsnittlig.

To av utviklerne hos leverandør B mener DES blir et helt greit prosjekt å jobbe med. Den tredje utvikleren mener det blir litt vanskelig å frigjøre seg fra tanken på DES som forskningsprosjekt. Dette kan gi et litt annet fokus. Motivasjon vurderes til gjennomsnittlig.

Hos leverandør C gir utviklerne et inntrykk av at DES er et enkelt prosjekt, som er planlagt i lang tid og som de vil gjøre ordentlig. Det at DES er et forskningsprosjekt virker å øke interessen for gjennomføring. Motivasjonen vurderes til litt over gjennomsnittlig.

Alle tre utviklerne hos leverandør D virker entusiastiske til å utvikle DES. De uttaler blant annet; *"…Hun tror dette prosjektet kan bli bra og spennende… …For han er dette en avveksling, noen nytt. Han opplever at dagene har flydd unna… …Spennende…".* Motivasjonen hos leverandør D vurderes til litt over gjennomsnittlig.

6.2.4.6 M6: Stabile krav

Utgangspunktet for DES var behovet for en ny forskningsdatabase. Selve utviklingen ble gjennomført med forskning. Det unike ved DES er at kravene er utviklet av profesjonelle datamennesker, og at de er stabile. Her gjelder det samme for Leverandørene. Flere uttaler at DES er et lite risikofylt prosjekt, det er atypisk på grunn av få avhengigheter til andre systemer og liten funksjonalitet. Dette punktet vurderes til uforanderlige krav for alle leverandørene.

6.2.4.7 M7: Deltidsarbeid

Hos leverandør A har ingen av utviklerne jobbet full tid med DES. En av utviklerne har jobbet betydelig mer enn de andre, se Figur 6-1, men fortsatt ikke full tid. Denne faktoren vurderes til litt over gjennomsnittlig deltidsarbeid.

Tilbudet til leverandør B viser at tre mennesker settes fulltid til utviklingen. Gjennom prosjektperioden har kun en utvikler arbeidet fulltid. De to andre har også jobbet med andre prosjekter. Antall deltidsarbeidende vurderes til litt over gjennomsnittlig.

Utviklerne hos leverandør C uttrykker konsekvent at de har hatt for mye å gjøre i tillegg til utviklingen av DES. De har ikke brukt så mye tid på DES som ønsket. Denne faktoren vurderes til samtlige utfører mye deltidsarbeid.

Utviklerne hos leverandør D gir inntrykk av å bruke det meste av sin tid på utviklingen av DES. De har også noen andre prosjekter pågående. Deltidsarbeid vurderes til litt over gjennomsnittlig.

6.2.4.8 M8: Vansker med programmeringsspråk

Hos leverandør A brukes PHP og HTML som programmeringsspråk. Det virker som om utviklerne ikke har vanskeligheter med dette. De roser hverandre for å være gode programmerere samtidig som de har flere års erfaring med denne typen programmering. Generelt er PHP og HTML forholdsvis enkelt. Dette punktet vurderes til gjennomsnittlig vanskelig programmeringsspråk.

Tilbudet til leverandør B forteller at; *"Prosjektdeltakerne er blitt valgt på bakgrunn av sine forutsetninger for å forstå og løse de spesifikke utfordringene som Simula står overfor"*. Utviklerne har en god del erfaring med

programmering, og har drevet med dette i flere år. Generelt vurderes vanskeligheter med programmeringsspråket til gjennomsnittlig.

Alle utviklerne hos leverandør C har erfaring med programmering fra tidligere. En av utviklerne er litt skeptisk til programmeringen på grunn av lite erfaring fra tidligere. En annen utvikler mener det er uvant med PHP. Utviklerne har generelt bra Java kunnskap. Denne faktoren vurderes til gjennomsnittlig.

En av utviklerne hos leverandør D anslår å ha programmert 80 – 90 % av tiden de siste tre år. De to andre har ikke mye Java erfaring. Denne faktoren vurderes til gjennomsnittlig.

6.2.5 Estimater

For å beregne ujusterte use case poeng, UUCP, summeres ujustert aktør vekt, UAW, og ujustert use case vekt, UUCW. Først beregnes $UUCP_1$ ut fra use case'ene beskrevet i kravspesifikasjonen. Denne beregningen er lik for alle fire leverandører.

$UUCP_1 = UAW + UUCW_1 = 10 + 55 = 65$

Se Tabell 6-9 for fullstendig oversikt over estimatene ved bruk av use case poeng.

6.2.5.1 Leverandør A

Ujusterte use case poeng justeres ved å multiplisere inn teknisk faktor og omgivelsesfaktor. For leverandør A beregnes justerte use case poeng, UCP_1, for use case'ene beskrevet i kravspesifikasjonen følgende;

$UCP_1 = UUCP_1 * TCF * EF = 65 * 0,85 * 0,875 = 48,34$

Ujusterte use case poeng, $UUCP_2$, der samtlige use case laget av leverandør A inkluderes, beregnes følgende;

$UUCP_2 = UAW + UUCW_2 = 10 + 60 = 70$

Dette fører til følgende justerte use case poeng, UCP_2;

$UCP_2 = UUCP_2 * TCF * EF = 70 * 0,85 * 0,875 = 52,06$

Metoden fra Schneider og Winters (2001) for å finne antall arbeidstimer resulterer i 20 arbeidstimer per UCP. Estimerte arbeidstimer for use case'ene i kravspesifikasjonen beregnes til 966 timer. Estimert antall arbeidstimer der alle use case laget av leverandør A er inkludert beregnes til 1041. Differansen mellom timeantallet for UCP_1 og UCP_2 hos leverandør A er 75 arbeidstimer.

6.2.5.2 Leverandør B

Justerte use case poeng, UCP_1, for use case'ene beskrevet i kravspesifikasjonen beregnes følgende;

$UCP_1 = UUCP_1 * TCF * EF = 65 * 0,89 * 0,86 = 49,751$

Ujusterte use case poeng, $UUCP_2$, der samtlige use case laget av leverandør B inkluderes, beregnes følgende;

$UUCP_2 = UAW + UUCW_2 = 10 + 85 = 95$

Dette fører til følgende justerte use case poeng, UCP_2;

$UCP_2 = UUCP_2 * TCF * EF = 95 * 0,89 * 0,86 = 72,713$

Metoden fra Schneider og Winters (2001) for å finne antall arbeidstimer resulterer i 20 arbeidstimer per use case poeng. Estimerte arbeidstimer for use case'ene i kravspesifikasjonen beregnes til 995 timer. Estimert antall arbeidstimer der alle use case laget av leverandør B er inkludert, beregnes til 1454. Differansen mellom timeantallet for UCP_1 og UCP_2 hos leverandør B er 459 arbeidstimer.

6.2.5.3 Leverandør C

Justerte use case poeng, UCP_1, for use case'ene beskrevet i kravspesifikasjonen beregnes følgende;

$UCP_1 = UUCP_1 * TCF * EF = 65 * 0,865 *0,935 = 52,57$

Ujusterte use case poeng, $UUCP_2$, der samtlige use case laget av leverandør B inkluderes beregnes følgende;

$UUCP_2 = UAW + UUCW_2 = 10 + 70 = 80$

Dette fører til følgende justerte use case poeng, UCP_2;

$UCP_2 = UUCP_2 * TCF * EF = 80 * 0,865 * 0,935 = 64,702$

Metoden fra Schneider og Winters (2001) for å finne antall arbeidstimer resulterer i 20 arbeidstimer per UCP. Estimerte arbeidstimer for use case'ene i kravspesifikasjonen, UCP_1, beregnes til 1051 timer. Estimert antall arbeidstimer der alle use case laget av leverandør C er inkludert beregnes til 1294. Differansen mellom timeantallet for UCP_1 og UCP_2 hos leverandør C er 243 arbeidstimer.

6.2.5.4 Leverandør D

Justerte use case poeng, UCP_1, for use case'ene beskrevet i kravspesifikasjonen beregnes følgende;

$UCP_1 = UUCP_1 * TCF * EF = 65 * 0,89 *0,89 = 51,487$

Ujusterte use case poeng, $UUCP_2$, der samtlige use case laget av leverandør B inkluderes, beregnes følgende;

$UUCP_2 = UAW + UUCW_2 = 10 + 60 = 70$

Dette fører til følgende justerte use case poeng, UCP_2;

$UCP_2 = UUCP_2 * TCF * EF = 70 * 0,89 * 0,89 = 55,447$

Metoden fra Schneider og Winters (2001) for å beregne antall arbeidstimer resulterer i 20 arbeidstimer per UCP. Estimerte arbeidstimer for use case'ene i kravspesifikasjonen, UCP_1, beregnes til 1030 timer. Estimert antall arbeidstimer der alle use case laget av leverandør D er inkludert beregnes til 1109. Differansen mellom timeantallet for UCP_1 og UCP_2 hos leverandør D er 79 arbeidstimer.

6.2.6 Restrukturering Use Case og Aktører

I kravspesifikasjonen er ingen spesielle hensyn tatt med tanke på use case'enes struktur. Heller ble ingen krav satt til struktur på leverandørenes tilbud. Dette førte til ulikt detaljnivå og ulike tilbud. I den forbindelse testes UCP-metoden på nytt ved å restrukturere use case og tilbud etter visse bestemmelser.

For CRUD operasjoner hersker generelt uenighet i use case'enes struktur [Cockburn 2001]. Spørsmålet er om use case som inneholder CRUD operasjoner skal defineres hver for seg eller som alternativ flyt i et større administrasjons use case. Prinsipielt definerer Cockburn (2001) CRUD operasjonene som separate ønsker, og dermed separate use case. Imidlertid identifiseres use case'ene, der CRUD operasjonene ikke er eksepsjonelt komplekse, som et administrasjons use case med flere muligheter i alternativ flyt. Adolph og Bramble (2003) støtter dette ved å si at CRUD use case enkelt beskriver individuelle rutinetransaksjoner som et system skal tilby. Ved å heve CRUD operasjonene et nivå, til et generelt use case, gjenspeiles bedre hva en bruker ønsker å oppnå med et system [Adolph og Bramble 2003]. I DES restruktureres dermed use case'ene ved å ta i bruk inkluderte og utvidede use case for trinn som gjentas og CRUD use case.

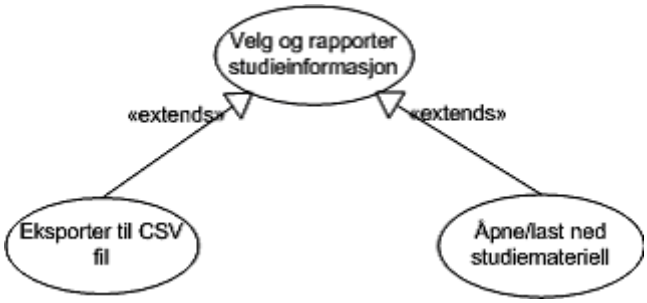
Fra kravspesifikasjonen definerer første linje for hvert use case handlingen logg inn. Dette gjelder der use case'et krever brukeridentifisering. Logg inn kunne blitt regnet som en egen transaksjon. Dersom logg inn hadde blitt vurdert som en egen transaksjon hadde flere av use case'ene blitt mer komplekse. UCP_1 og UCP_2 ville med dette blitt noe høyere. I tillegg ville samme transaksjon blitt telt flere ganger. En annen mulighet ville vært å lage et eget use case av logg inn. Dette har flere av leverandørene gjort. Allikevel vurderer jeg logg inn i UCP_1 og UCP_2 som en forutsetning for å bruke noe av funksjonaliteten og ikke som et eget use case. Ved restrukturering kan logg inn legges til som et inkludert use case. Siden dette ikke vil utgjøre noen forskjell i antall UCP vurderes logg inn også i UCP_3 kun som en forutsetning for å bruke deler av DES.

6.2.6.1 Restrukturering av use case fra kravspesifikasjon

Use case'ene rediger studie og slett studie er typiske CRUD use case. I den forbindelse heves disse opp ett nivå til ett felles use case; administrer studier. Slett studie og rediger studie blir to utvidede use case, se Figur 6-5.

Figur 6-5 Restrukturering av rediger studie og slett studie

En annen restrukturering av use case'ene i kravspesifikasjonen gjelder velg og rapporter studieinformasjon. Valg av en spesifikk studie går igjen som felles start i tre av de andre use case'ene. Disse er; skriv ut rapport, eksporter til CSV fil og åpne/last ned studiemateriell fil.

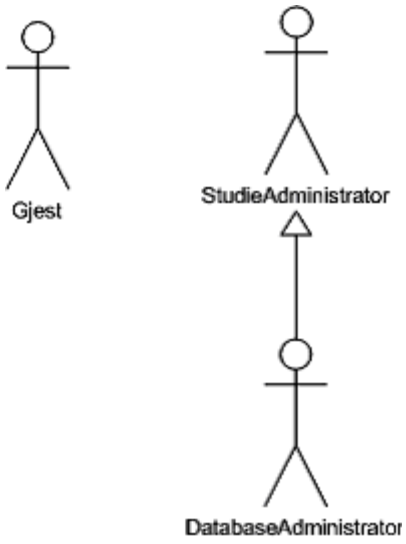


Figur 6-6 Restrukturering av Velg og rapporter studieinformasjon, Eksporter til CSV fil og Åpne/last ned studiemateriell.

Antall use case reduseres fra 2 komplekse og 7 enkle til 6 enkle use case. For leverandørene beregnes den nye use case vekten til 30. $UUCP_3$ blir 37.

6.2.6.2 Generalisering av aktører

De tre menneskelige aktørene generaliseres på følgende måte, se Figur 6-7; databaseadministrator arver ansvarsområdene til studieadministrator i tillegg til sine egne ansvarsområder. Når dette gjøres reduseres antallet komplekse aktører fra tre til to.



Figur 6-7 Restrukturering av aktører

Aktøren Simulaweb blir som den alltid har vært. Den nye ujusterte aktør vekt, UAW_3, beregnes til 7.

6.2.6.3 Leverandør A – Restrukturering av use case

Leverandør A legger til et use case. I dette gjøres ingen restrukturering. $UUCP_3$ for leverandør A blir 42. UCP_3 beregnes som følger;

$UCP_3 = UUCP_3 * TCF * EF = 42 * 0,85 * 0,875 = 31,24$

Med 20 arbeidstimer per use case poeng lyder estimatet for leverandør A på 625 arbeidstimer.

6.2.6.4 Leverandør B – Restrukturering av use case

Leverandør B har lagt til 6 use case. To av disse er use case'ene er vedlikehold av studiemateriale og vedlikehold av studiematerialenes relasjoner. Vedlikehold av relasjonene kan omgjøres til et utvidet use case av vedlikehold av studiemateriale. Disse to restruktureringene gjør at leverandør B reduserer sine use case med to enkle. $UUCP_3$ for leverandør B blir 62. UCP_3 beregnes som følger;

$UCP_3 = UUCP_3 * TCF * EF = 62 * 0,89 * 0,86 = 47,455$

Med 20 arbeidstimer per use case poeng lyder estimatet for leverandør B på 949 arbeidstimer.

6.2.6.5 Leverandør C – Restrukturering av use case

Leverandør C har lagt til 3 use case. Ett av disse er skriv ut rapport. Dette use case'et krever at velg og rapporter studieinformasjon er gjennomført først. Skriv ut rapport kan omgjøres til et utvidet use case. Dette gjør at leverandør C reduserer sine use case med ett enkelt. $UUCP_3$ for leverandør C blir 47. Dermed beregnes UCP_3 som følger;

$UCP_3 = UUCP_3 * TCF * EF = 47 *0,865 * 0,935 = 38,01$

Med 20 arbeidstimer per use case poeng lyder estimatet for leverandør C på 760 arbeidstimer.

6.2.6.6 Leverandør D – Restrukturering av use case

Leverandør D har kun lagt til ett use case. Ingen restrukturering kan gjennomføres. $UUCP_3$ for leverandør D blir 42. UCP_3 beregnes som følger;

$UCP_3 = UUCP_3 * TCF * EF = 42 * 0,89 * 0,89 = 33,27$

Med 20 arbeidstimer per use case poeng lyder estimatet for leverandør D på 665 arbeidstimer.

6.2.7 Sammendrag

Lev	Faktisk brukt tid	Ekspert estimat	UCP_1	UCP_2	UCP_3
A	582	250	966	1041	625
B	953	341	995	1454	949
C	345	100	1051	1294	760
D	849	650	1030	1109	665

Tabell 6-9 Sammenstilling av estimatene ved bruk av UCP-metoden

Tabell 6-9 viser en sammenstilling over estimatene hos leverandørene ved bruk av UCP-metoden. Schneider og Winters (2001) forslag for beregning av antall arbeidstimer per UCP er benyttet. Summert antall vurderinger under 3 for M1 til M6, og antall vurderinger over tre for M7 og M8 gir 20 arbeidstimer per UCP for alle leverandørene.

I Tabell 6-9 tar UCP_1 utgangspunkt i kravspesifikasjonen. Det resulterer i estimater rundt 1000 arbeidstimer hos alle leverandørene. Leverandør A og B ligger litt under 1000 timer med henholdsvis 966 og 995 timer. For leverandør A er dette 384 timer mer enn faktisk brukt timeantall. Imidlertid stemmer det ganske godt med leverandør B sitt faktisk brukte timeantall. 995 timer er bare 42 timer over faktisk brukt timeantall. For leverandør C og D estimeres det i overkant av 1000 timer, henholdsvis 1051 og 1030. For leverandør C er dette 706 timer mer enn brukt timeantall. For leverandør D er dette 181 timer mer enn brukt timeantall.

UCP_2 vurderer use case'ene fra kravspesifikasjonen inkludert use case'ene leverandørene selv laget i estimeringen. For leverandør B og C gjorde dette stort utslag. For leverandør B økte estimatene med 459 timer. For leverandør C økte estimatene med 243 timer. Leverandør A og D har kun lagt til ett use case hver, og estimatene økte med henholdsvis 75 og 79 timer.

UCP_3 estimerer antall timer etter at use case'ene er restrukturert. For leverandør A og B førte restruktureringen til at estimatene nærmet seg faktisk brukt tid. UCP_3 estimerte 43 timer over faktiskbrukt tid for leverandør A og 4 timer under faktisk brukt tid for leverandør B. For leverandør C estimerte UCP_3 415 timer over faktisk brukt tid. Dette er en bedring fra UCP_1 Og UCP_2. Leverandør D er eneste

leverandør der UCP_3 blir betraktelig lavere enn brukt tid. UCP_3 estimerer 184 timer lavere enn faktisk brukt timeantall.

6.3 Beregning av ujusterte funksjonspoeng

Funksjonspoeng benyttes i dette studiet som grunnlag for å estimere antall kildekodelinjer i COCOMO II og WEBMO.

6.3.1 Leverandør A

FP	Antall	Beregnet verdi
Interne logiske filer	2 - lav	14
Eksterne grensesnittfiler	2 – lav	10
Ekstern inndata	5 – lav	15
Ekstern utdata	5 – lav	20
Ekstern spørring	8 - lav	24
Ujusterte funksjonspoeng		83

Tabell 6-10 Beregning av ujusterte funksjonspoeng for leverandør A

Interne logiske filer: All informasjon som behandles i DES lagres en database. Leverandør A har delt inn databasen i to deler. Den ene delen tar for seg studieadministrasjon. Den andre delen tar for seg konfigurasjon av åpningsteksten i DES. To logiske filer telles. Delen for studieadministrasjon har ingen lagringselementer. Totalt er fire dataelementer definert. Delen for åpningstekst har kun et dataelement og et lagringselement. Begge de logiske filene kategoriseres til lav.

Eksterne grensesnittfiler: Databasen til DES skal kommunisere med databasen til Simulaweb for henting av blant annet personopplysninger. DES skal også integreres i Simulaweb. Spesifikasjonene til disse er ikke kjent. To eksterne grensesnittfiler telles. Begge kategoriseres til lav.

Ekstern inndata: Leverandør A definerer følgende ekstern inndata; nytt studie, rediger studie, slett studie, forandre velkomsttekst og tildeling av rettigheter. Samtlige refererer til enten en eller to logiske filer med maksimum 15 dataelementer. Fem ekstern inndata telles. Alle fire kategoriseres til lav.

Ekstern utdata: Leverandør A definerer følgende eksterne utdata; feilside, bekreftelsessider for ny og rediger studie, forandre velkomsttekst, tildeling av brukerrettigheter og slett studie. Samtlige eksterne utdata refererer til en logisk fil. Alle fem filer kategoriseres til lav.

Ekstern spørring: Følgende eksterne forespørsler er beskrevet av leverandør A; hent studie, eksporter studieinformasjon, aggreger rapport, list opp studier, list opp brukere, hent bruker, hent administrators åpningsside, last ned studie. Samtlige eksterne forespørsler kategoriseres til lav.

6.3.2 Leverandør B

FP	Antall	Beregnet verdi
Interne logiske filer	2 - lav	14
Eksterne grensesnittfiler	2 – lav	10
Ekstern inndata	11 – lav	33
Ekstern utdata	2 – lav	8
Ekstern spørring	9 - lav	27
Ujusterte funksjonspoeng		92

Tabell 6-11 Beregning av ujusterte funksjonspoeng for leverandør B

Interne logiske filer: Leverandør B bruker database til lagring og vedlikehold av studieinformasjon. De anbefaler å opprette en egen database for DES. Det betyr at de må forholde seg til 2 databaser, DES for studieinformasjon og Simula sin database for personinformasjon. Åpningstekst lagres i en egen tekstfil. Denne inkluderes i PHP koden. Simula sin database eksisterer fra tidligere. Tekstfilen omhandler kun en tekststreng og kategoriseres til lav. 32 dataelementer er definert. Ingen lagringselementer. Dermed telles 2 interne logiske filer som kategoriseres til lav.

Eksterne grensesnittfiler: Simulaweb sin database er allerede eksisterende. DES integreres i Simulaweb. Spesifikasjonene til disse er ukjent. To eksterne grensesnittfiler telles og kategoriseres til lav.

Ekstern inndata: Leverandør B definerer følgende ekstern inndata; nytt studie, rediger studie, slett studie, legge til, endre og fjerne relasjoner til publikasjoner og studiemateriell, legg til, rediger og fjerne knyting mellom forskningsstudier og metadata, brukeradministrasjon og endre åpningsside. Samtlige refererer til enten en eller to logiske filer med maksimum 15 dataelementer. 11 eksterne inndata telles og kategoriseres til lav.

Ekstern utdata: Leverandør B nevner ingen ting om bekreftelser eller feilsider i DES, utenom bekreftelse på slett studie som beskrives i kravspesifikasjonen. I den forbindelse telles to eksterne outputer. Bekreft slett studie og administrators åpningsside. Begge kategoriseres ti lav.

Ekstern spørring: Følgende eksterne forespørsler finnes i tilbudet fra leverandør B; søk frem eksisterende studier, eksporter studieinformasjon, aggreger rapport, list opp studier, list opp brukere, hent bruker, administrators åpningsside, åpningsside, last ned studie. Samtlige kategoriseres til lav.

6.3.3 Leverandør C

FP	Antall	Beregnet verdi
Interne logiske filer	1 – lav	7
Eksterne grensesnittfiler	2 – lav	10
Ekstern inndata	8 – lav	24
Ekstern utdata	2 – lav	8
Ekstern spørring	10 - lav	30
Ujusterte funksjonspoeng		79

Tabell 6-12 Beregning av ujusterte funksjonspoeng for leverandør C

Interne logiske filer: Leverandør C beskriver en database for vedlikehold både av studier og brukere. Ingen lagringselementer er definert. 38 dataelementer er identifisert. Den interne logiske filen kategoriseres til lav.

Eksterne grensesnittfiler: Leverandør C har to grensesnittfiler; Simulaweb og den allerede eksisterende databasen. Spesifikasjonene til disse er ikke kjent. Begge kategoriseres til lav.

Ekstern inndata: Leverandør C definerer følgende ekstern inndata; nytt studie, rediger studie, slett studie, rediger åpningstekst, rediger brukertype, legg til, rediger og slett record. Samtlige refererer til en intern logisk fil og kategoriseres dermed til lav.

Ekstern utdata: Følgende ekstern utdata er definert av leverandør C; generer redigeringsforløp (lik for alle sider) og feilmelding. Begge kategoriseres til lav.

Ekstern spørring: Følgende eksterne forespørsler er definert av leverandør C; hent studie, hent bruker, søk etter studie, skriv rapport, eksporter til kommaseparert fil, velg data å vedlikeholde, åpne og laste ned studiemateriell, rapporter aggregert studier, list opp brukere og list opp studier. Samtlige eksterne forespørsler kategoriseres til lav.

6.3.4 Leverandør D

FP	Antall	Beregnet verdi
Interne logiske filer	1 – gj.snitt	10
Eksterne grensesnittfiler	2 – lav	10
Ekstern inndata	6 – lav	18
Ekstern utdata	2 – lav	8
Ekstern spørring	9 - lav	27
Ujusterte funksjonspoeng		73

Tabell 6-13 Beregning av ujusterte funksjonspoeng for leverandør D

Interne logiske filer: I tilbudet til leverandør D nevnes kun bruk av en database for vedlikehold av studier. Denne kan grupperes i tre lagringselementer; menneske, studie og nøkkelord. Totalt er 20 unike dataelementer definert. Denne logiske filen kategoriseres til gjennomsnittlig.

Eksterne grensesnittfiler: DES har to grensesnittfiler. Disse er Simulaweb og Simula sin eksisterende database. Spesifikasjonene til disse er ikke kjent. Begge kategoriseres til lav.

Ekstern inndata: Leverandør D definerer følgende ekstern inndata; nytt studie, rediger studie, slett studie, legg til studietype og nøkkelord, tildeling av brukerrettigheter og rediger tekst på åpningsside. Disse kategoriseres til lav.

Ekstern utdata: Følgende ekstern utdata er definert av leverandør D; bekreft sletting og feilmelding. Begge kategoriseres til lav.

Ekstern spørring: Leverandør D definerer følgende eksterne forespørsler; velg og rapporter studieinformasjon, søk etter studie, rapporter aggregert studieinformasjon, åpne og last ned studie, eksporter studie til kommaseparert tekstfil, åpne kalender, list opp studier og list opp brukere. Samtlige eksterne forespørsler kategoriseres til lav.

6.3.5 Sammendrag

Leverandør	UFP
A	83
B	92
C	79
D	73

Tabell 6-14 Antall ujusterte funksjonspoeng

Antall ujusterte funksjonspoeng avhenger av detaljnivå på leverandørenes beskrivelser av DES. Leverandør B har mest omfattende beskrivelse av DES med noen flere funksjoner enn de andre leverandørene. Størrelsen på DES med leverandør B sin løsning er 92 UFP. Leverandør A og C har beskrevet DES med omtrent likt antall funksjoner. Størrelsene er henholdsvis 83 og 70 UFP. Leverandør D har beskrevet DES minst omfattende. Dette resulterte i 73 UFP. Ut fra Tabell 6-14 ser det ut som om leverandør B ønsker å lage en applikasjon som er 19 funksjonspoeng større enn leverandør D.

6.4 Estimering med COCOMO II

I COCOMO II benyttes tidlig design til estimeringen. Grunnlaget for å estimere timeantall ved hjelp av COCOMO II er antall kilo kildekodelinjer (KLOC). Først beregnes antall ujusterte funksjonspoeng (UFP), se avsnitt 6.3. Deretter multipliseres antall funksjonspoeng med antall kodelinjer (LOC) per funksjonspoeng. Tabell 2.4 i [Boehm et al. 2000c: s. 20] beskriver antall kildekodelinjer per UFP for ulike programmeringsspråk. For fjerde generasjonsspråk benyttes 20 LOC per UFP. Reifer (2002) foreslår en fornying av denne oversikten, se Vedlegg D. For objektorienterte webapplikasjoner benyttes 25 kildekodelinjer per funksjonspoeng [Reifer 2002]. Vurderingene av skaleringsfaktorer og multiplikatorer er gjort i samsvar med Boehm et al. (2000-c: s. 30-58).

Leverandør	UFP	KLOC
A	83	2,08
B	92	2,30
C	79	1,98
D	73	1,83

Tabell 6-15 Antall UFP og KLOC for de fire leverandørene

6.4.1 Skaleringsfaktorer

Tabell 6-16 viser alle vurderinger av skaleringsfaktorene for de fire leverandørene. Den nederste linja, (∑SF) summerer skaleringsfaktorene. Nedenfor forklares skaleringsfaktorenes vurderinger.

Skaleringsfaktor	Leverandør A	Leverandør B	Leverandør C	Leverandør D
Forståelse	2,48	3,72	1,24	3,72
Utviklingsfleksibilitet	2,03	3,04	1,01	3,04
Risikovurdering	2,83	1,41	1,41	2,83
Team kohesjon	2,19	2,19	1,10	3,29
Prosessmodenhet	4,68	4,68	7,80	3,12
∑SF	14,21	15,04	12,56	16,00

Tabell 6-16 Vurdering av skaleringsfaktorene

6.4.1.1 Forståelse

Utviklerne hos leverandør A gir inntrykk av å ha god erfaring med utvikling av programvare som DES. Fot tre av utviklerne er prosjektet veldig vanlig. Utviklerne vurderes å være generelt godt kjent med og ha god forståelse for denne typen prosjekter. Høy.

DES er et mindre prosjekt enn leverandør B vanligvis gjennomfører. Allikevel er funksjonaliteten i DES kjent for utviklerne. Utviklerne vurderes å være noe ukjent med denne typen prosjekter. Nominell.

Utviklerne hos leverandør C gir inntrykk av å ha god erfaring fra utvikling av programvare som DES. Blant annet uttales følgende; *"For Lev C er DES et ganske typisk mindre Web-prosjekt. De har gjort samme type prosjekter før… …Teknologien som anvendes er typisk for de og de har brukt den med hell tidligere"*. Utviklerne hos leverandør C vurderes å være godt kjent med og god forståelse for denne type utvikling. Veldig høy.

For to av utviklerne hos leverandør D er DES, med unntak av størrelsen, et helt vanlig prosjekt. Den tredje utvikleren har litt mindre erfaring med denne type prosjekt. Utviklerne vurderes å være noe ukjent. Nominell.

6.4.1.2 Utviklingsfleksibilitet

Når det gjelder fleksibilitet i utvikling stiller Simula få krav; koden skal være lett forståelig for en systemutvikler for vedlikeholding, det skal brukes UML og snapshot. Utenom dette kan leverandørene utvikle som de vil.

Leverandør A bruker mønstre som MVC og struts. Ellers gir leverandør A inntrykk av å ikke sette spesielt strenge krav til riktig oppbygging av programvaren. Med dette må leverandør A kun gjøre noen få tilpasninger. Høy.

Leverandør B er mer opptatt av å gjennomføre utviklingen så riktig som mulig. De jobber ekstra nøye med utviklingsprosessen og riktig bruk av kodestandarder. Leverandør B har pålagt seg mindre fleksibilitet enn leverandør A. Nominell.

Leverandør C bruker mønstre som MVC og struts. Ellers er de ikke spesielt opptatt av riktig bruk av kodestandarder og oppbygging. Leverandør C må kun gjøre få tilpassninger i utviklingen. Veldig høy.

Leverandør D gir inntrykk av å vurdere DES som et veldig seriøst prosjekt. De har pålagt seg strengere krav til arbeid med utviklingsprosessen i tillegg til at prosjektleder er ISO-sertifisert. Nominell.

6.4.1.3 Risikovurdering

DES er et forholdsvis lite prosjekt med stabile kundekrav. Risikoen i utviklingen her er generelt lav for alle leverandørene. Leverandør A har ikke kommentert risikoer ved utviklingen av DES. Høy. Leverandør B mener at de har vært realistiske med tanke på ekstra tid i forbindelse med risiko. Veldig høy. Leverandør C sier blant annet følgende om risiko; *"DES har en mye bedre kravspesifikasjon enn det som er vanlig… … er sikker på at de kommer i mål og risikoen er bortimot null, i og med at de har undersøkt de få risikoer de så for seg".* Veldig høy. Leverandør D har ikke vurdert eller kommentert risiko. Høy.

6.4.1.4 Team kohesjon

Utviklerne hos leverandør A sitter sammen og jobber. En av utviklerne har hatt en del kontakt med alle de andre tidligere. De tre andre utviklerne hos leverandør A er forholdsvis ukjente med hverandres arbeidsmetoder. Mot Simula har leverandør A sjeldnere kontakt enn de vanligvis har med kundene. Høy.

Utviklerne hos leverandør B virker å kjenne hverandre forholdsvis godt. To av dem jobber sammen for første gang. De virker alle fornøyd med å skulle jobbe med hverandre. Høy.

Leverandør C er et lite firma. Dette gjør at utviklerne kjenner hverandre meget godt. Utviklerne sitter i et åpent kontorlandskap. De har hyppig og god kommunikasjon. Høy.

Utviklerne hos leverandør D er litt ferske i forhold til å jobbe sammen. Allikevel er det i utviklingen av DES et lite team hvor de deler kontor og har god muntlig kommunikasjon. Nominell.

6.4.1.5 Prosessmodenhet

For å vurdere prosessmodenhet må leverandørenes CMM nivå være kjent. Ingen av leverandørene har oppgitt modenhetsnivå og en grundigere analyse av leverandørene er nødvendig. Ut fra tilbud, internettsider og intervjuer vurderes leverandør A til å være en forholdsvis stor organisasjon med en godt innkjørt utviklingsmodell. Leverandør A sitt modenhetsnivå vurderes til CMM nivå 2. Nominell.

Leverandør B er en forholdsvis stor organisasjon med en godt innkjørt utviklingsmodell. Leverandør B gir inntrykk av å ta ekstra hensyn til rutiner og metodikk i dette prosjektet. Modenhetsnivået vurderes til CMM nivå 2. Nominell.

Leverandør C er en liten bedrift. De følger ingen spesifikk utviklingsmodell i utviklingen av DES. Mye av arbeidet er ad hoc med få rutiner. Modenhetsnivået til leverandør C vurderes til CMM 1 nivå. Lav.

Leverandør D er en del av en stor internasjonal bedrift. De bruker en utviklingsmodell som er en *"egen dialekt av RUP men mindre omfattende"*. Prosjektleder hos leverandør D er sertifisert etter ISO-standarden. Det betyr at visse rutiner må gjennomføres. Modenhetsnivået til leverandør D vurderes til CMM nivå 3. Høy.

6.4.2 Multiplikatorer

Tabell 6-17 oppsummerer vurderingene av multiplikatorene i COCOMO II. For beskrivelse av multiplikatorer og formler generelt, se avsnitt 3.6.

Multiplikatorer	Leverandør A	Leverandør B	Leverandør C	Leverandør D
Stabilitet og kompleksitet	0,83	1,00	0,83	1,00
Gjenbruk	1,00	1,15	0,95	1,07
Plattformvansker	1,00	1,00	1,00	1,00
Personellets kompetanse	1,00	0,83	1,00	1,00
Personellets erfaring	0,87	0,87	1,00	1,00
Fasiliteter	1,00	1,00	1,00	1,10
Tidspress	1,00	1,00	1,00	1,00
∏EM	0,72	0,83	0,79	1,18

Tabell 6-17 Vurdering av multiplikatorer

6.4.2.1 Stabilitet og kompleksitet

Ingen spesifikke *stabilitetskrav* er nevnt. Dersom det skulle skje en feil i DES er det antakeligvis feil der data enkelt kan gjenopprettes. Dette på grunn av Simula sine fungerende rutiner innen sikkerhetskopiering. Noe vekt på stabilitet. Nominell. *Databasen* i DES er liten. *Kompleksiteten* i DES vurderes å være enkel. Operasjoner og grensesnitt er lite komplekst.

For alle leverandørene krever Simula følgende *dokumentasjon*; snapshot, noe UML og kommentarer i koden. Leverandør A bruker en utviklingsmodell som krever noen artefakter. Noe vekt på dokumentasjon. Lav. Leverandør B følger sin utviklingsmodell nøye i utviklingen av DES. Det kommenteres at DES er et fint prosjekt å teste modellen på. Stor vekt på dokumentasjon. Høy. Leverandør C lager den dokumentasjon som Simula krever. Utenom dette følger ikke leverandør noen spesifikk utviklingsmodell. Lite vekt på dokumentasjon. Meget lav. Leverandør D Lager dokumentasjonen som Simula krever. I tillegg er prosjektleder for utviklerne ISO-sertifisert. Dette fører med seg spesielle artefakter som må produseres. Stor vekt på dokumentasjon. Høy.

Oppsummert gir dette følgende;

- Leverandør A – lav
- Leverandør B – nominell
- Leverandør C – lav
- Leverandør D - nominell

6.4.2.2 Gjenbruk

Gjenbruk er et punkt der leverandørene tenker ulikt. Som beskrevet tidligere bruker leverandør A MVC mønsteret for å skille grensesnittlaget fra forretningslaget. Nominell.

Leverandør B bruker mer tid på å bygge opp programvaren riktig med tanke på kodestandarder og komponenter. Veldig høy..

Leverandør C er den leverandøren som tar minst hensyn til kodestandarder og god arkitektur med bruk av komponenter. Lav.

Leverandør D gir inntrykk av å bruke mer tid på god arkitekturisk oppbygging med komponenter og kodestandarder. Høy.

6.4.2.3 Plattformvansker

Det er ikke stilt spesielle krav til *eksekveringstid* og *minneplass*. For samtlige leverandører, nominell. Leverandørene velger å bruke samme plattform som Simula allerede benytter. Store endringer i plattformen skjer antakelig ikke oftere enn en gang i året. Stabil plattform, Nominell.

Oppsummert gir dette nominell for alle leverandørene.

6.4.2.4 Personellets kompetanse

Analysekompetanse: Utviklerne hos leverandør A gir inntrykk av at de har normal erfaring i programvareutvikling, både når det gjelder analyse og implementering. Nominell. Alle utviklerne hos leverandør B har tidligere jobbet med objektorientert analyse og design. Samtidig virker de å kunne samarbeide bra. Høy. En av utviklerne hos leverandør C har 3 – 4 års erfaring fra tidligere analysearbeid. De to andre har mest teorikunnskap om temaet. Leverandør C er et lite firma der utviklerne kjenner hverandre godt. De sitter tett og har god kommunikasjon. Nominell. Ingen av utviklerne hos leverandør D har særlig erfaring fra analyse. De har blitt kurset i analyse ved oppstart i firmaet. Det er vanskelig å si noe om utviklernes effektivitet og gjennomføring. Utviklerne virker også ferske i det å jobbe sammen. Derimot jobber de tett på samme kontor, og har god kommunikasjon. Nominell.

Programmeringskompetanse: Den generelle kompetansen til å programmere hos utviklerne i leverandør A er nominell. De gir inntrykk av å kommunisere bra. Nominell. Utviklerne hos leverandør B liker å kode. De har generelt god erfaring fra programmering. Utviklerne virker å ha god kontakt med hverandre å glede seg over at de skal jobbe sammen i utviklingen av DES. Nominell. Utviklerne hos leverandør C jobber tett. Nominell. Utviklerne hos leverandør D virker ferske i det å jobbe sammen. Derimot jobber de tett på samme kontor, og har god kommunikasjon. Nominell.

Personellets stabilitet: Hos leverandør A har en to utviklere hjulpet til, i tillegg til de tre opprinnelige. Nominell stabilitet. Leverandør B har kun hatt tre utviklere med i utviklingen av DES. Stabiliteten er veldig høy. Leverandør C har fått litt hjelp av en fjerde utvikler i utviklingen av DES. Høy. Leverandør D har kun de tre opprinnelige utviklerne innblandet i utviklingen av DES. Meget høy stabilitet.

Oppsummert gir dette følgende;

- Leverandør A – nominell
- Leverandør B – høy
- Leverandør C – nominell
- Leverandør D - nominell

6.4.2.5 Personellets erfaring

Utviklerne hos leverandør A har følgende erfaring fra denne type *applikasjoner*; en utvikler har drevet med denne type i to år, en har jobbet med to tilsvarende applikasjoner. De andre utviklerne har ikke definert antall år, men virker også delvis erfarne med denne type applikasjon. En av utviklerne hos leverandør A har ikke brukt den *plattformen* som skal benyttes i særlig stor grad. De andre utviklerne virker å ha noe erfaring. Fra intervjuene av utviklerne i leverandør A presiseres at *verktøyene* fungerer veldig bra og de påvirker ikke prosjektet i noen grad. Utviklerne bruker kjente verktøy. Når det gjelder *programmering* roser utviklerne hverandre for at de er gode programmerere. Det ser ut til at de har relativt god erfaring med programmeringsspråket. Generelt beregnes 1 - 2 års erfaring for utviklerne hos leverandør A. Høy.

Hos leverandør B er selve løsningen vanlig. Derimot er det sjelden at leverandør B har så små prosjekter der de selv står for all utvikling. Applikasjonserfaring vurderes til omtrent 1 års gjennomsnittlig erfaring blant utviklerne. Leverandør B velger å bruke samme *plattform* som Simula har fra tidligere. I tilbudet sier de Utviklerne har god kjennskap til denne plattformen. Hos utviklerne i leverandør B virker *programmeringserfaringene* å være gode. En av utviklerne har blant annet brukt Java i fem år. Når det gjelder erfaring med verktøyene som brukes har samtlige utviklere hos leverandør B valgt verktøy de kjenner. Generelt 2 års gjennomsnittlig erfaring.

Leverandør C gir inntrykk av at de har god erfaring med denne type *applikasjon*. Sett bort fra størrelsen er DES et typisk prosjekt for leverandør C. Leverandør C gir inntrykk av å ha hatt suksess med utviklingsprosjekter på denne *plattformen* tidligere. Utviklerne hos leverandør C ser ut til å ha bortimot 3 års erfaring med *programmeringsspråket* generelt. Når det gjelder verktøy bruker leverandør C *verktøy* de kjenner fra tidligere. Alle utviklerne hos leverandør C er godt kjent med disse. Nominell.

Utviklerne hos leverandør D har generelt god erfaring med denne type applikasjon. En av utviklerne har mindre enn de andre to. Dermed vurderes *applikasjonserfaring* hos teamet til en snittverdi på 1 års erfaring. Utviklerne hos leverandør D har liten erfaring med plattformen fra tidligere. Generelt i teamet vurderes plattformerfaring til gjennomsnitt på 6 måneder. En av utviklerne hos leverandør D har programmert det meste av tiden i en 3 års periode. De to andre har mindre erfaring. Når det gjelder verktøy er de forholdsvis ukjent. Generelt vurderes teamet å ha 1 års erfaring med språk og verktøy. Nominell.

Oppsummert gir dette følgende;

- Leverandør A – høy
- Leverandør B – høy
- Leverandør C – nominell
- Leverandør D - nominell

6.4.2.6 Fasiliteter

Grad av verktøybruk: Leverandør A ser ut til å bruke grunnleggende verktøy til sin livssyklus. Leverandør B ser ut til å bruke verktøy i relativt høy grad. Leverandør C velger noe verktøystøtte for å kunne generere en del kode. Leverandør D bruker verktøy i grunnleggende verktøystøtte i sin livssyklus.

Flersteds utvikling: Utviklingen foregår hos leverandør A. Utviklerne sitter tett sammen, i samme bygg og jobber med hver sine deler av DES. Simula som er kunden befinner seg i samme by. Kommunikasjonen mellom kunde og leverandør foregår både elektronisk og over telefon. Leverandør B og kunden sitter i samme by. Utviklerne mener de her har mindre kontakt med kunden enn de pleier. All utvikling skjer i leverandør B sine lokaler. Leverandør C er geografisk plassert i en annen by enn kunden. Utviklerne hos leverandør C sitter sammen og jobber. De jobber med hver sine deler av DES. All utvikling skjer i leverandør C sine lokaler. Leverandør D er lokalisert i samme by som kunden. All utvikling skjer i leverandør D sine lokaler.

Oppsummert gir dette følgende;

- Leverandør A – nominell
- Leverandør B – nominell
- Leverandør C – nominell
- Leverandør D - lav

6.4.2.7 Tidspress

For alle de fire leverandørene vurderes tidspress til nominell. Leverandørene gir inntrykk av å ha prøvd å være realistiske i estimering av tidsplan. Dette på grunn av at DES er et forskningsprosjekt.

6.4.3 Estimering

For å beregne estimert antall timer brukes verdiene funnet over i formlene for COCOMO II.

6.4.3.1 Leverandør A

For leverandør A gjelder følgende;

E = B + 0,01 * SF = 0,91 + 0,01 * 14,21 = 1,0521

Månedsverk = A * (Størrelse ^ E) * EM = 2,08 * (2 ^ 1,0521) * 0,72 = 4,58

4,58 månedsverk gir;

Innsats i timer = 4,58 månedsverk * 152 timer per månedsverk = 695,5 timer

6.4.3.2 Leverandør B

For leverandør B gjelder følgende;

E = B + 0,01 * SF = 0,91 + 0,01 * 15,04 = 1,0604

Månedsverk = A * (Størrelse ^ E) * EM = 2,94 * (2,30 ^ 1,0604) * 0,83 = 5,90

4,49 månedsverk gir;

Innsats i timer = 5,90 månedsverk * 152 timer per månedsverk = 897,56 timer

6.4.3.3 Leverandør C

For leverandør C gjelder følgende;

E = B + 0,01 * SF = 0,91 + 0,01 * 12,56 = 1,0356

Månedsverk = A * (Størrelse ^ E) * EM = 2,94 * (1,98 ^ 1,0356) * 0,79 = 4,69

4,69 månedsverk gir;

Innsats i timer = 4,69 månedsverk * 152 timer per månedsverk = 712,99 timer

6.4.3.4 Leverandør D

For leverandør D gjelder følgende;

E = B + 0,01 * SF = 0,91 + 0,01 * 16 = 1,07

Månedsverk = A * (Størrelse ^ E) * EM = 2,94 * (1,83 ^ 1,07) * 1,18 = 6,59

6,59 månedsverk gir;

Innsats i timer = 6,59 måneder * 152 timer per månedsverk = 1001,19timer

6.4.4 Sammendrag

Lev	Faktisk brukt tid	Ekspert estimat	COCOMO II
A	582	250	696
B	953	341	898
C	345	100	713
D	849	650	1001

Tabell 6-18 Sammenstilling av estimater ved bruk av COCOMO II

COCOMO II tidlig design ble benyttet for å estimere DES. For leverandør A førte det til at estimatet ble 114 timer over faktisk brukt timeantall med 799 timer. Dette er 549 timer mer enn leverandør A estimerte. Med COCOMO II ble det for leverandør B estimert 960 timer. Denne verdien ligger omtrent likt med faktisk brukt timeantall på 953 timer. For leverandør C estimerte COCOMO II 713 timer.

Dette er 327 timer over leverandør C sitt faktisk brukte timeantall på 345 timer. Selv hadde leverandør C estimert med 100 timer til utvikling av DES. Leverandør D estimerte selv 650 timer. Totalt ble 849 timer brukt. Med COCOMO II ble det estimert midt 1121 timer.

6.5 Estimering med WEBMO

I WEBMO er en del av grunnlaget for å estimere antall kildekodelinjer (LOC) antall ujusterte funksjonspoeng (UFP). I tillegg til UFP kommer andre faktorer som benyttes i webutvikling. Disse er; linker, skript, webbyggesteiner og multimedia filer. WEBMO er beskrevet i avsnitt 3.7.

6.5.1 Beregne antall webobjekter

Størrelsen i WEBMO måles i antall webobjekter. For en applikasjon som DES benyttes en størrelsesfaktor på 25 LOC per webobjekt [Reifer 2002].

6.5.1.1 Leverandør A

Faktorer	Antall	Beregnet verdi	Antall webobjekter
UFP		83	
Multimediefiler			
Webbyggesteiner	13 – lav	39	
Skript	18 – lav	36	
Linker	1 – lav 2 – gj.snitt	11	
Antall webobjekter			169
LOC			4225

Tabell 6-19 Summering av antall webobjekter for leverandør A

Ujusterte funksjonspoeng: UFP for leverandør A ble beregnet i avsnitt 0. Antall ujusterte funksjonspoeng er 83.

Multimediefiler: DES er en databaseapplikasjon for oppbevaring og vedlikehold av forskningsrapporter. Ekstra grafikk som video og lyd blir ikke benyttet. Bilder til logo er allerede eksisterende siden DES skal integreres i Simulaweb. Ingen multimediefiler telles.

Webbyggesteiner: Antall operander; logo i velkomstside er eneste unike byggestein i DES. I tillegg telles antall knapper tegnet inn i tilbudets storyboards; lagre/rediger studie har to, slett studie har to, søk studie har en, tildele rettigheter har en og endre åpningsside har en. Antall operatører; Siden leverandør A ikke definerer logo eller knappene spesifikt, vurderes generelt 5 operatører til bruk. Samtlige operatører og operander vurderes til lav. Ingen avanserte byggesteiner inkluderes i DES.

Skript: Antall operander; 10 use case. Antall operatører; åtte hyperlinker til navigasjon, åpne studie (i rediger og ny studie), list opp studie, rediger studie, endre åpningstekst, endre brukerrettigheter, list opp brukere, søk etter studie. Des har maksimum 3 aktører. Dermed kategoriseres alle elementene i til lav.

Linker: Antall linker; link for å koble grensesnitt med database, link for å koble DES databasen mot den eksisterende databasen hos Simula, link for å integrere DES i Simulaweb. Integreringen med Simulaweb foregår i PHP/html og vurderes til lav. Linkingen mot databasene foregår ved hjel av spørrelinjer/SQL og vurderes til gjennomsnittlig.

Summering av faktorene resulterer i 169 webobjekter. Antall kildekodelinjer blir 4225.

6.5.1.2 Leverandør B

Faktorer	Antall	Beregnet verdi	Antall webobjekter
UFP		92	
Multimediefiler			
Webbyggesteiner	12 – lav	36	
Skript	24 – lav	48	
Linker	1 – lav 2 – gj.snitt	11	
Antall webobjekter			187
LOC			4675

Tabell 6-20 Summering av antall webobjekter for leverandør B

Ujusterte funksjonspoeng: UFP for leverandør B ble beregnet i avsnitt 6.3.2. Antall ujusterte funksjonspoeng er 92.

Multimedia filer: DES er en databaseapplikasjon for oppbevaring og vedlikehold av forskningsrapporter. Ekstra grafikk som video og lyd blir ikke benyttet. Bilder til logo er allerede eksisterende siden DES skal integreres i Simulaweb. Dermed telles ingen multimedia filer.

Webbyggesteiner: Antall operander; leverandør B nevner ingen spesifikk bruk av logo. Leverandør B har heller ikke beskrevet noen prototype eller liknende. Derfor vurderes generelt nødvendige html knapper i DES; fire knapper for lagring, redigering og sletting av studie, en knapp for søk etter studie, en knapp for redigering av åpningstekst og en knapp for tildeling brukerrettigheter. Antall operatører; Siden leverandør B ikke definerer operandene spesifikt, telles generelt fem operatører. Samtlige operander vurderes til lav.

Skript: Antall operander; 16 use case. Antall operatører; åtte hyperlinker til navigasjon. Des har maksimum 3 aktører. Dermed kategoriseres alle elementene i til lav.

Linker: Antall linker; link for å koble grensesnitt med database, link for å koble DES databasen mot den eksisterende databasen hos Simula, link for å integrere DES i Simulaweb. Integreringen med Simulaweb foregår i PHP/html og vurderes til lav. Linkingen mot databasene foregår ved hjel av spørrelinjer/SQL og vurderes til gjennomsnittlig.

Summering av faktorene resulterer i 187 webobjekter. Antall kildekodelinjer blir 4675.

6.5.1.3 Leverandør C

Faktorer	Antall	Beregnet verdi	Antall webobjekter
UFP		79	
Multimediefiler	1 – lav	3	
Webbyggesteiner	12 – lav	36	
Skript	21 – lav	42	
Linker	1 – lav 2 – gj.snitt	11	
Antall webobjekter			171
LOC			4275

Tabell 6-21 Summering av antall webobjekter for leverandør C

Ujusterte funksjonspoeng: UFP for leverandør C ble beregnet i avsnitt 6.3.3. Antall ujusterte funksjonspoeng er 79.

Multimedia filer: DES er en databaseapplikasjon for oppbevaring og vedlikehold av forskningsrapporter. Ekstra grafikk som video og lyd blir ikke benyttet. Leverandør C velger å presentere aggregert rapport i form av et .gif bilde, dersom muligheten til fremvisning i Excel ikke er tilstede. Denne kategoriseres til lav.

Webbyggesteiner: Leverandør C nevner ingen spesifikke byggesteiner for web. Derfor vurderes generelt nødvendige byggesteiner. Antall operander; html knapper: fire knapper for lagring, redigering og sletting av studie, en knapp for søk etter studie, en knapp for redigering av åpningstekst og en knapp for tildeling brukerrettigheter. Antall operatører; Siden leverandør C ikke definerer operandene spesifikt, telles generelt fem operatører. Samtlige operander vurderes til lav.

Skript: Antall operander; 13 use case. Antall operatører; åtte hyperlinker til navigasjon. Des har maksimum 3 aktører. Dermed kategoriseres alle elementene i til lav.

Linker: Antall linker; link for å koble grensesnitt med database, link for å koble DES databasen mot den eksisterende databasen hos Simula, link for å integrere DES i Simulaweb. Integreringen med Simulaweb foregår i PHP/html og vurderes til lav. Linkingen mot databasene foregår ved hjel av spørrelinjer/SQL og vurderes til gjennomsnittlig.

Summering av faktorene resulterer i 171 webobjekter. Antall kildekodelinjer blir dermed 4275.

6.5.1.4 Leverandør D

Faktorer	Antall	Beregnet verdi	Antall webobjekter
UFP		73	
Multimediefiler	1 – lav	3	
Webbyggesteiner	12 – lav	36	
Skript	18 – lav	36	
Linker	1 – lav 2 – gj.snitt	11	
Antall webobjekter			159
LOC			3975

Tabell 6-22 Summering av antall webobjekter for leverandør D

Ujusterte funksjonspoeng: UFP for leverandør C ble beregnet i avsnitt 6.3.4. Antall ujusterte funksjonspoeng er 73.

Multimedia filer: DES er en databaseapplikasjon for oppbevaring og vedlikehold av forskningsrapporter. Ekstra grafikk som video og lyd blir ikke benyttet. Leverandør D velger å presentere aggregert rapport i form av et jpg/gif bilde. Denne kategoriseres til lav.

Webbyggesteiner: Leverandør D nevner ingen spesifikke byggesteiner for web. Derfor vurderes generelt nødvendige byggesteiner. Antall operander; html knapper: fire knapper for lagring, redigering og sletting av studie, en knapp for søk etter studie, en knapp for redigering av åpningstekst og en knapp for tildeling brukerrettigheter. Antall operatører; Siden leverandør D ikke definerer operandene spesifikt, telles generelt fem operatører. Samtlige operander vurderes til lav.

Skript: Antall operander; 10 use case. Antall operatører; åtte hyperlinker til navigasjon. Des har maksimum 3 aktører. Dermed kategoriseres alle elementene i til lav.

Linker: Antall linker; link for å koble grensesnitt med database, link for å koble DES databasen mot den eksisterende databasen hos Simula, link for å integrere DES i Simulaweb. Integreringen med Simulaweb foregår i PHP/html og vurderes til lav. Linkingen mot databasene foregår ved hjel av spørrelinjer/SQL og vurderes til gjennomsnittlig.

Summering av faktorene resulterer i 159 webobjekter. Med 25 kildekodelinjer per webobjekt blir størrelsen av DES 3975 LOC.

6.5.2 Kostnadsrivere

Tabell 6-23 viser en oversikt over vurderte kostnadsdrivere med vurderinger for hver av leverandørene i WEBMO.

Kostnadsdrivere	Leverandør A	Leverandør B	Leverandør C	Leverandør D
Stabilitet og kompleksitet	1,00	1,00	1,00	1,00
Plattform-vanskeligheter	0,87	0,87	0,87	0,87
Personellets kapabilitet	1,00	1,00	1,00	1,00
Personellets erfaring	0,87	1,00	0,87	1,19
Fasiliteter	0,85	1,00	1,00	0,85
Tidspress	1,00	1,00	1,00	1,00
Gjenbruk	1,00	1,25	1,00	1,25
Teamarbeid	0,75	0,75	0,75	1,00
Prosesseffektivitet	0,85	0,85	1,35	0,85
$\prod CD_i$	0,41	0,69	0,77	0,94

Tabell 6-23 Vurdering av kostnadsdriverne for samtlige leverandører

6.5.2.1 Stabilitet og kompleksitet

I dette punktet tenker leverandørene likt. DES er en vanlig klient/tjener løsning med full distribusjon. Noen matematiske beregninger med tanke på aggregering av rapporter. For alle leverandørene; nominell.

6.5.2.2 Plattformvanskeligheter

DES skal kjøres i Simula sine allerede eksisterende systemer. Der er det få plattformendringer, raskt nett og få ressursproblemer. Her gjelder samme vurdering for alle leverandørene. Lav.

6.5.2.3 Personellets kapabilitet

Personellets kapabilitet er vanskelig å vurdere ut fra intervjuer. Ingen av leverandørene beskriver hyppigheten i utskiftelse av ansatte. Det virker som om utviklerne hos de fire leverandørene generelt er litt ukjente og litt kjente med å jobbe sammen. Samtlige jobber tett og gir inntrykk av å ha god kommunikasjon. Nominell.

6.5.2.4 Personellets erfaring

Utviklerne hos leverandør A har generelt jobbet med denne type programmering i lengre tid. Det nevnes at de to som programmerer hos leverandør A er dyktige og selvdrevne programmerere, DES er typisk deres arbeidsområder. Antall år med erfaring i programmeringsspråket nevnes ikke. Når det gjelder verktøy har leverandørene generelt valgt verktøy de kjenner fra tidligere. Det betyr at de kjenner verktøyene rimelig bra. En av utviklerne kjenner ikke noe særlig til MySQL. Ellers har de en del erfaring med Apache og Tomcat, Visio og Eclipse. Alle verktøy er generelt blitt brukt de siste ni måneder. Selve type prosjekt har utviklerne også en del erfaring i. Gjennomsnittlig for utviklerne hos leverandør A vurderes erfaring med verktøy, programmeringsspråk og plattform til mellom 1 og 3 år. Høy.

I tilbudet sier leverandør B følgende; *"Prosjektdeltakerne er blitt valgt på bakgrunn av sine forutsetninger for å forstå og løse de spesifikke utfordringene*

som Simula står ovenfor." Programmeringserfaringene til utviklerne ser ut til å være gode. En av utviklerne har blant annet brukt Java i fem år. Generelt vurderes utviklernes programmeringserfaring til nominell, 1 år i gjennomsnitt. Når det gjelder erfaring med verktøyene som brukes, har samtlige utviklere hos leverandør B valgt verktøy de kjenner. Leverandør B velger å bruke samme plattform som Simula har fra tidligere. I tilbudet sier de *"DES-løsningen blir implementert ved hjelp av Simulas valgte og prefererte teknologiske løsning slik det er beskrevet... Lev B ser ingen umiddelbare tekniske utfordringer knyttet til oppsettet som beskrevet."* Utviklerne har god kjennskap til denne plattformen. Generelt vurderes plattform, språk og verktøyerfaring til nominell.

Utviklerne hos leverandør C har relativt god erfaring med programmeringsspråk og plattform. Det eneste som er litt ukjent for noen av utviklerne er bruk av PHP. Ellers har de flere års erfaring med både plattform og programmeringsspråk. Når det gjelder verktøy har de selv valgt hvilke verktøy de bruker. Utviklerne hos leverandør C vurderes til å ha mellom 1 og 3 års erfaring med programmeringsspråk, plattform og verktøy. Høy.

Utviklerne hos leverandør D har liten erfaring med Linux og MySql fra tidligere. Tomcat har utviklerne mer erfaring med. En av utviklerne hos leverandør D har programmert meste parten av tida i en 3 års periode. De andre to har mindre erfaring. Når det gjelder verktøy er de forholdsvis ukjent med MySql og Netbeans. Generelt vurderes teamet å ha 6 måneders erfaring med språk og verktøy. Lav.

6.5.2.5 Fasiliteter

Utviklerne hos leverandør A jobber i samme bygning nær hverandre. I tillegg gir de inntrykk av å ha relativt integrerte verktøy og metoder. Høy.

Utviklerne hos leverandør B sitter tett sammen og jobber. Utviklingen av DES velger de å gjennomføre som en test av deres metode og verktøy. I den forbindelse velger de å bruke mer tid på utviklingsprosessen, som om den skulle være mindre kjent. *"Metodikken blir mer vektlagt i dette prosjektet enn det som er vanlig. Siden den er en fremtredende del av prosjektet blir den også prioritert mer og det blir satt av mer tid til forarbeidet…"*. Nominell.

Leverandør C har to utviklere som gjør hver sine deler av DES. Utviklerne kjenner hverandre godt og jobber tett. Siden kunden som her er Simula befinner seg i en annen by vurderes fasiliteter til nominell.

Leverandør D sitter tett og jobber. De har flere års erfaring fra firmaet. Samme bygning, integrerte verktøy og metoderutiner. Høy.

6.5.2.6 Tidspress

Leverandør A definerer følgende i sin kravspesifikasjon:*"…og det er derfor ikke planlagt med arbeid ut over normal arbeidstid"*. De gir uttrykk for å ha vært realistiske i estimeringen. Nominell.

Leverandør B gir uttrykk av å ha vært realistiske i sine estimeringer, uten å få det travelt. Nominell.

Da leverandør C laget tilbudet estimerte de med å kunne bruke god tid. Senere har de fått mer å gjøre slik at DES blir et prosjekt som gjøres innimellom andre prosjekter dersom tiden tilsier det. Nominell.

Leverandør D mener at de har vært realistiske i sine estimeringer. Nominell.

6.5.2.7 Gjenbruk

Leverandør A gir ikke inntrykk av å ha planlagt særlig gjenbruk. Uplanlagt gjenbruk, nominell.

Leverandør B gir inntrykk av å ta dette med gjenbruk og komponentbasert utvikling mer alvorlig enn de andre leverandørene. Noe planlagt gjenbruk. Høy.

Leverandør C gir ikke innrykk av å ha planlagt gjenbruk. Nominell.

Leverandør D gir inntrykk av å ta jobbe mer med komponentbasert utvikling som en begynnelse for gjenbruk. Høy.

6.5.2.8 Teamarbeid

Utviklerne hos leverandør A har alle jobbet en stund hos bedriften. Selv om ikke alle har jobbet med hverandre tidligere sitter de tett og deler leverandør A sin visjon for utvikling. Selve teamet som skal gjennomføre utviklingen har få utviklere, noe som gjør kommunikasjonen enklere. Delt visjon og sterk team kohesjon, høy.

Utviklerne hos leverandør B virker å kjenne hverandre forholdsvis godt. To av dem jobber sammen for første gang. De virker alle fornøyd med å skulle jobbe med hverandre. Teamkohesjon hos leverandør B vurderes til delt visjon og sterk team kohesjon. Høy.

Leverandør C er et lite firma. Dette gjør at utviklerne kjenner hverandre meget godt. I tillegg sitter de i et åpent kontorlandskap. Utviklerne føler de har hyppig og god kommunikasjon. Team kohesjonen hos leverandør C vurderes til høy.

Utviklerne hos leverandør D er litt ferske i forhold til å jobbe sammen. Allikevel er det i utviklingen av DES et lite team hvor de har mye muntlig kommunikasjon. Team kohesjonen vurderes til nominell.

6.5.2.9 Prosesseffektivitet

Leverandør A bruker en egenutviklet prosess. I tillegg har utviklerne god erfaring fra liknende prosjekter. Høy.

Leverandør B bruker en egenutviklet tilpasset evolusjonær inkrementell utviklingsprosess i utviklingen av DES. Høy.

Leverandør C følger ingen spesifikk utviklingsprosess. Selv kommenterer de sin prosess; *"… code and fix and pray*." Prosesseffektiviteten hos leverandør C vurderes til veldig lav.

Leverandør D jobber med en egenutviklet utviklingsprosess. Utviklerne har blitt kurset i denne og mener den sitter implisitt i tankegangen. Høy.

6.5.3 Estimering

DES er en enkel databaseapplikasjon for web. I den forbindelse vurderes DES i kategorien; webbasert informasjonstjenester. Der utgjør konstantene i formlene for DES følgende; A = 2.1, B = 2.0, P1 = 1.00 og P2 = 0.5.

6.5.3.1 Leverandør A

For leverandør A gjelder følgende;

Innsats = $A\prod cd_i(\text{Størrelse})^{P1}$ = 2,1 * 0,41 * 4,225 KLOC ^ 1 = 3,64 månedsverk

Med 152 timer per månedsverk estimeres;

Antall timer = 3,64 månedsverk * 152 timer/månedsverk = 553,13 timer.

6.5.3.2 Leverandør B

For leverandør B gjelder følgende;

Innsats = $A\prod cd_i(\text{Størrelse})^{P1}$ = 2,1 * 0,69 * 4,675 KLOC ^ 1 = 6,80 månedsverk

Med 152 timer per månedsverk estimeres;

Antall timer = 6,80 månedsverk * 152 timer/månedsverk = 1034,56 timer.

6.5.3.3 Leverandør C

For leverandør C gjelder følgende;

Innsats = $A\prod cd_i(\text{Størrelse})^{P1}$ = 2,1 * 0,77 * 4,275 KLOC ^ 1 = 6,88 månedsverk

Med 152 timer per månedsverk estimeres;

Antall timer = 6,88 månedsverk * 152 timer/månedsverk = 1045,76 timer.

6.5.3.4 Leverandør D

For leverandør D gjelder følgende;

Innsats = $A\prod cd_i(\text{Størrelse})^{P1}$ = 2,1 * 0,93 * 3,975 KLOC ^ 1 = 7,81 månedsverk

Med 152 timer per månedsverk estimeres;

Antall timer = 7,81 månedsverk * 152 timer/månedsverk = 1186,35 timer.

6.5.4 Sammendrag

Lev	Faktisk brukt tid	Ekspert estimat	WEBMO
A	582	250	553
B	953	341	1035
C	345	100	1046
D	849	650	1186

Tabell 6-24 Sammenstilling av estimater ved bruk av WEBMO

DES ble kategorisert til en webbasert informasjonstjeneste. Dette førte til følgende faktorer ved beregning av timeantall: A = 2,1 og P1 = 1,00. For leverandør A ble det med WEBMO estimert 553 timer. Dette er 303 timer mer enn leverandør A selv estimerte, og 29 timer mindre enn faktisk brukt tid. For leverandør B estimerte WEBMO 1035 timer. Dette er 82 timer mer enn faktisk brukt tid som var 953 timer. For leverandør C ble det med WEBMO estimert 1046 timer. Dette er 701 timer mer enn faktisk brukt tid. For leverandør D ble det estimert 1186 timer med WEBMO. Selv hadde de estimert 650 timer. Faktisk brukt tid for leverandør D lå omtrent i midten med 849 timer.

6.6 Estimatenes nøyaktighet

For å måle estimaters nøyaktighet i programvareutvikling benyttes vanligvis målet MRE (Magnitude of Relative Error). Formelen for MRE er;

$$\mathbf{MRE = |brukt\ tid - estimat| / brukt\ tid}$$

MRE er ikke optimal for måling av nøyaktighet da systematisk skjevhet og spredning ikke skilles. Uansett hvor mye et programvareutviklingsprosjekt er under- eller overestimert kan MRE aldri bli høyere enn 1. Jørgensen og Sjøberg (2004) beskriver en metode som er bedre enn MRE ved blant annet store differanser mellom brukt tid og estimat. Målet kalles BRE (Balanced Relative Error) og formlene er [Jørgensen og Sjøberg 2004];

$$\frac{\text{brukt tid} - \text{estimat}}{\text{brukt tid}}, \text{ brukt tid} \leq \text{estimat}$$

$$\frac{\text{brukt tid} - \text{estimat}}{\text{estimat}}, \text{ brukt tid} > \text{estimat}$$

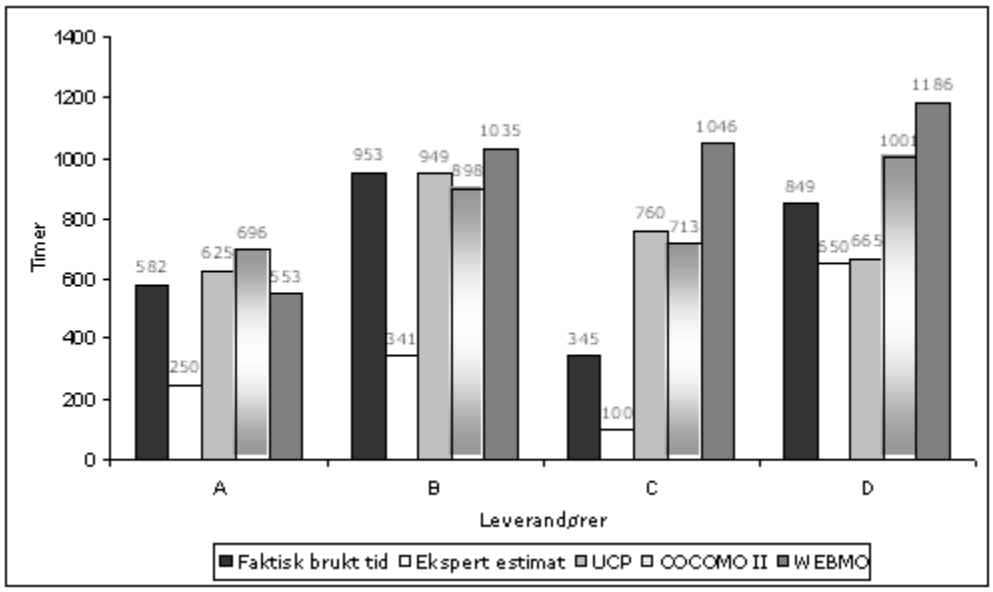
Tabell 6-25 Formler for BRE [Jørgensen og Sjøberg 2004]

Faktisk brukt tid, ekspertenes og estimeringsmodellenes estimater puttes inn i formlene i Tabell 6-25. Tabell 6-26 viser BRE for alle estimatene. Jo nærmere 0 BRE er, desto bedre er estimatet. Det er tydelig at ekspertene selv har underestimert prosjektene sine kraftig. Med unntak av leverandør C er estimatene fra UCP-metoden nærmest faktisk brukt tid. Leverandør C skiller seg ut i alle estimeringsmetodene ved betraktelig overestimering. Estimatene fra COCOMO II og WEBMO ligger også innenfor akseptabelt BRE-nivå rundt 0.

Leverandør	Ekspert estimat	UCP_3	COCOMO II	WEBMO
A	1,33	-0,07	-0,20	0,12
B	1,79	0,00	0,16	-0,09
C	2,45	-1,20	-3,68	-2,03
D	0,31	0,22	-0,18	-0,40

Tabell 6-26 BRE for alle estimater

En annen måte å se metodenes presisjon på er vist i Figur 6-8 der første svarte, søyle for hver leverandør utgjør faktisk brukt tid. For ekspertenes egne estimater og estimeringsmodellene er det om å gjøre å komme nærmest samme høyde som faktisk brukt tid. Figuren viser at generelt ligger UCP-metoden og COCOMO II nærmest faktisk brukt tid.



Figur 6-8 Tidsbruk, faktisk brukt tid og estimater

6.7 Sammendrag

Lev	Faktisk brukt tid	Ekspert estimat	UCP_3	COCOMO II	WEBMO
A	582	250	625	696	553
B	953	341	949	898	1035
C	345	100	760	713	1046
D	849	650	665	1001	1186

Tabell 6-27 Alle timeestimater for DES avrundet

Først ga UCP-metoden for høye estimater. Grunnen til dette var detaljnivået i use case'ene. Ved å restrukturere use case'ene passet estimatene bedre med faktisk brukt tid. Derfor vil UCP_3 bli brukt i diskusjonen.

For leverandør A estimerte UCP_3, COCOMO II og WEBMO ganske nærme faktisk brukt tid. UCP_3 estimerte 43 timer høyere, COCOMO II estimerte 114 timer høyere og WEBMO estimerte 29 timer lavere. Leverandør A er en vel etablert bedrift med godt etablerte rutiner for programvareutvikling. Ekspertene hos leverandør A hadde alle noe erfaring fra denne type utviklingsprosjekter. De estimerte selv 332 timer lavere enn faktisk brukt timeantall.

For leverandør B estimerer UCP_3, COCOMO II og WEBMO ganske nærme faktisk brukt timeantall. UCP_3 estimerte 4 timer høyere, COCOMO II estimerte 55 timer

lavere og WEBMO estimerte 82 timer høyere. Leverandør B gir utrykk for å bruke utviklingen av DES til å teste seg selv og utviklingsmetoden. Dette fører til at de følger utviklingsmetoden til punkt og prikke, og at de setter av mer tid til de forskjellige delene enn normalt. I tillegg har utviklerne generelt en del erfaring fra utvikling av liknende funksjonalitet.

Leverandør C estimerte selv et timeantall på 100 timer. De endte opp med å bruke 349 timer. Estimeringsmodellene estimerer et høyere timeantall. UCP_3 estimerte 415 timer høyere, COCOMO II estimerte 368 timer høyere og WEBMO estimerte 701 timer høyere. Leverandør C er et lite og ungt firma. De har ikke etablert noen spesifikk utviklingsmodell, og arbeider ad hoc.

For leverandør D estimerte UCP_3 nærmest brukt timeantall med 184 timer lavere. COCOMO II og WEBMO estimerte henholdsvis 152 og 337 timer høyere enn faktisk brukt timeantall. Utviklerne hos leverandør D virket å være mer ukjent med utviklingsplattform og PHP enn utviklerne hos de andre leverandørene og estimerte i utgangspunktet høyest timeantall fra begynnelsen. Leverandør D en godt etablert internasjonal bedrift der utviklerne er kurset i bedriftens utviklings-modell. De ser også ut til å ha et ønske om å gjøre utviklingen så riktig som mulig med tanke på kodestruktur og oppbygging.

Generelt ligger estimatene fra use case poeng og COCOMO II mellom 600 og 1000 arbeidstimer. WEBMO estimerer rundt 1000 arbeidstimer med unntak av estimatene for leverandør A. Et første inntrykk er at modellene estimerer riktigst når utviklerne skal følge en utviklingsmodell nøye, og utvikle riktig med tanke på blant annet kvalitet, gjenbruk, arkitektur.

7 SAMMENLIKNING OG DISKUSJON

Dette kapittelet diskuterer UCP, FPA, COCOMO II og WEBMO. Først settes størrelsesfaktorene modellene bruker opp imot hverandre. Dette gjelder ujusterte use case poeng (UUCP), ujusterte funksjonspoeng (UFP) og ujusterte webobjekt. Hovedfokus er likheter og ulikheter mellom UUCP og UFP. Videre sammenliknes modellenes kostnadsdrivere. Her vurderes hvilken informasjon som må være kjent for å lage riktige estimater, i hvilke tilfeller modellene kan være nyttige, når de bør kombineres med andre modeller og hvilken kvalitet modellene antar. Til slutt diskuteres estimatene.

7.1 Måling av programvarens størrelse

Størrelsesfaktorene UUCP, UFP og webobjekt er beskrevet i avsnittene 4.2, 3.3.1 og 3.7. I dette avsnittet sammenliknes områdene som berøres når størrelse på programvare skal estimeres. I tillegg vurderes spørsmålet om størrelsesmålene er så like at de kan mappes direkte over i hverandre.

7.1.1 Nødvendig kunnskap om systemet

For å beregne programvarens størrelse krever UUCP og UFP noe ulik kunnskap om systemet som skal lages. Følgende må være kjent for å beregne UUCP;

- Antall ønsker/mål kunden har for systemet under utvikling
- Målene må være tilstrekkelig spesifisert, slik at transaksjoner kan identifiseres på et nivå med en aktør som hendelsesstarter og system som respondent
- Hvordan ulike aktører kommuniserer med systemet

For å beregne UFP må følgende være kjent;

- Hver enkelt funksjon i systemet som skal lages, dette gjelder funksjoner for inndata, utdata og spørringer
- Antall logiske filer i systemet. Dette gjelder eksterne grensesnittfiler og interne logiske filer
- Antall logiske filer hver funksjon refererer til
- Antall dataelementer i hver funksjon og logiske fil, felter og attributter
- Antall lagringselementer per logiske fil der strukturen for lagring må være kjent

7.1.2 Størrelsesfaktorene for DES

Første likhetstrekk ved størrelsesmålene er framgangsmåten for å beregne dem. Min studie viser likhetstrekkene i fremgangsmåten;

- Ovenfra og ned metoder definert fra brukers perspektiv med utgangspunkt i hele systemet som brytes ned i mindre elementer
- For hvert element vurderes kompleksitet etter skalaen lav, gjennomsnittlig og høy
- For hver vurdering gjelder en konstant der antall elementer per vurdering multipliseres med konstanten
- Produktet for samtlige vurderinger summeres og summen utgjør det

ujusterte målet på programvarens størrelse
- Størrelsesmålene er uavhengig av teknologi og programmeringsspråk

Leverandørene imellom estimerer med en variasjon i $UUCP_2$ og $UUCP_3$, se Tabell 7-1. Hovedgrunnen til variasjonen er leverandørenes valg av ulikt antall tilleggs use case. Jo flere use case, desto høyere antall UUCP. Leverandørene opererer også med ulikt detaljnivå i use case'ene. Antall transaksjoner i "samme" use case kan dermed bli telt ulikt. Dette viser at dersom UUCP skal være et fast mål på et system, forutsettes visse restriksjoner i use case beskrivelsene.

Variasjonen i antall UFP leverandørene imellom samvarierer ikke likt som for UUCP, se Tabell 7-1. Grunnen til at leverandør C og D avviker fra samvariasjonen er løsningsskissen beskrevet i tilbudene. Jeg har funnet at;

- Leverandør C og D definerte en logisk fil til henholdsvis lav og gjennomsnittlig. Leverandør A og B har definert to lave logiske filer. Totalt gir to lave logiske filer et høyere antall UFP
- Leverandør C har definert en del ekstra use case, men ikke ekstra funksjoner. I tillegg definerer leverandør C færre eksterne utdata enn leverandør A. Oppsummert gir dette lavere antall UFP enn for leverandør A
- Leverandør D har definert flere eksterne inndata og eksterne spørringer enn leverandør A. Allikevel gir færre logiske filer og færre eksterne utdata et lavere antall UFP

Antall webobjekter hos leverandør A, B og C samvarierer med UUCP. Grunnen til at leverandør C ikke samvarierer for UFP, men samvarierer igjen for webobjekt er leverandør C sitt tillegg på tre use case mer enn leverandør A. Dette økte antall webobjekt i forhold til UFP. Grunnen til at leverandør D beregner færre webobjekt enn leverandør A, henger igjen fra UFP. Summen av tillegget til UFP for å beregne webobjekt er lik for leverandør A og D.

Størrelsesmål	Leverandør A	Leverandør B	Leverandør C	Leverandør D
$UUCP_2$	70	95	80	70
$UUCP_3$	42	62	47	42
UFP	83	92	79	73
Webobjekt	169	187	171	159

Tabell 7-1 Ujusterte størrelsesmål for DES

7.1.3 Klassifisering av elementer i systemet

7.1.3.1 Transaksjoner visa funksjoner

UCP-metoden og FPA vurderer de samme funksjonelle kravene til et system, men vurderingene er noe forskjellig. Min studie viser at;

- Begge metodene ser på hendelsesforløpene i use case'ene
- UCP-metoden studerer hele hendelsesforløpet i en transaksjon
- En transaksjon i UCP-metoden tilsvarer primært eksterne inndata og eksterne spørringer i FPA

Longstreet (2003) tar også med ekstern utdata når han beskriver oppdelingen av en transaksjon. Det er enkelt å si at en transaksjon tilsvarer disse tre

funksjonstypene, og ofte lager systemet ekstern utdata som resultat av en hendelse startet av aktør. Jeg har imidlertid ikke vurdert alternativ flyt som omhandler feilmeldinger eller genererte rapporter, ved telling av transaksjoner, og mener dermed at ekstern utdata er et område som kun delvis dekkes av en transaksjon.

7.1.3.2 Systemets grenser og arkitektur

Når det gjelder aspektene utenfor systemets grense vurderer metodene noe ulikt. Min studie viser at;

- UCP-metoden vurderer grensesnittet mot brukere og andre systemer, dette gjelder både primæraktørenes og støtteaktørenes kommunikasjon med systemet
- FPA forholder seg kun til lagring av data, som primært utgjør noen av systemets støtteaktører
- For å vurdere disse, og også de andre funksjonstypene, studerer FPA kompleksiteten i den underliggende datamodellen. Dette ignoreres i UCP-metoden

Også i UCP-metoden kan datamodell eller arkitektur utgjøre en forskjell på estimater og faktisk brukt tid. Anda et al. (2002) beskriver en mulig nødvendighet i å estimere arkitektur i tillegg til UCP når kunden er ukjent og arkitekturen usikker. Hvordan dette gjøres best mulig er foreløpig noe usikkert. Et forslag fra Anda et al. (2002) er å bytte ut omgivelsesfaktor M7 til å gjelde arkitektur. Dette vil føre til en litt upresis estimering av arkitektur da oppbygging av arkitektur tar omtrent like lang tid uavhengig av systemets størrelse. Ved å benytte omgivelsesfaktoren vil den prosentvise tiden forbundet med arkitektur alltid være lik og dermed kun gjelde for en viss type prosjekter. En annen mulighet er å ta hensyn til arkitekturen allerede ved vurdering av use case'enes kompleksitet [Anda et al. 2002]. Her tror jeg egenskaper ved FPA kan trekkes inn for å få med vurdering av arkitektur dersom det er nødvendig. Longstreet (2003) presiserer at metodene utfyller hverandre og dermed kan benyttes sammen, noe som også var intensjonen til Karner da UCP-metoden ble presentert i 1993. Her presiserer jeg at dersom UCP-metoden skal være brukervennlig også for en kunde, bør den ikke bli særlig mer kompleks. Det kan være nødvendig å kun benytte UCP-metoden for en kunde for så å støtte estimatene med FPA for en leverandør. Jeg vil tro at hovedønsket for de fleste er å kun ha en metode å forholde seg til. Derfor foreslår jeg en oppdeling av UCP-metoden, der en enkel versjon benyttes av kunder, og en noe mer teknisk versjon benyttes leverandører der også arkitektur kan innlemmes.

Dette studiet viser også at;

- Vurdering av use case i UCP-metoden har noe høyere verdi enn vurdering av funksjoner i FPA

Dette kan være fordi transaksjonene i UCP-metoden vurderes med mindre detaljkunnskap om systemet enn ved vurdering av funksjonene i FPA, samtidig som det telles færre use case enn funksjoner. Dermed må vurderingen av ett use case dekke mer av utviklingen enn det vurderingen av en funksjon må.

7.1.3.3 Modellenes fleksibilitet og mapping

Modellene har ulike krav til fleksibilitet. Min studie viser at;

- UCP-metoden er generelt mer fleksibel og enklere å beregne når gode use case foreligger

- FPA er en mer presist definert med klare kriterier for vurderinger
- Dette gjør at FPA ser på en noe snevrere del av funksjonaliteten

Slik vil direkte mapping, det vil si å sette UUCP lik UFP, ikke bli riktig. I studiet av DES viser variasjonen i antall UFP og antall UUCP dette godt. Imidlertid forskes det på muligheten i å telle ujusterte funksjonspoeng fra use case. Fetcke et al. (1997) har beskrevet denne typen mapping ved hjelp av en firestegs omformingsregel. Han viser at mapping er fult mulig og uavhengig av teknologi og prosjekter. Fordelen med dette er, som også Longstreet (2003) forklarer, at funksjonspoeng kan telles ut fra kravspesifikasjonen allerede tidlig i foranalyse-fasen. Begrensningen ved Fetcke et al. (1997) sine omformingsregler er at de kun gjelder for Jacobsons objektorienterte modell OOSE. Longstreet (2003) presenterer også et forslag på hvordan beregne funksjonspoeng ut ifra use case. Begge presiserer at problemet ved FPA har vært ukomplette og inkonsistente kravspesifikasjoner, og at estimatene forbedres ved å innlemme use case i estimeringen. Jeg tror også at ved mapping andre vegen;

- UCP-metoden støttes ved å ta i bruk noen tekniske elementer fra FPA ved vurdering av use case'enes eller aktørenes kompleksitet

kan forbedre estimater i UCP-metoden.

7.1.3.4 Et eksempel

Tabell 7-2 beskriver use case'et Slett Studie fra DES. Dette use case'et og nødvendig informasjon omkring brukes som eksempel for å forklare klassifisering av elementene i DES.

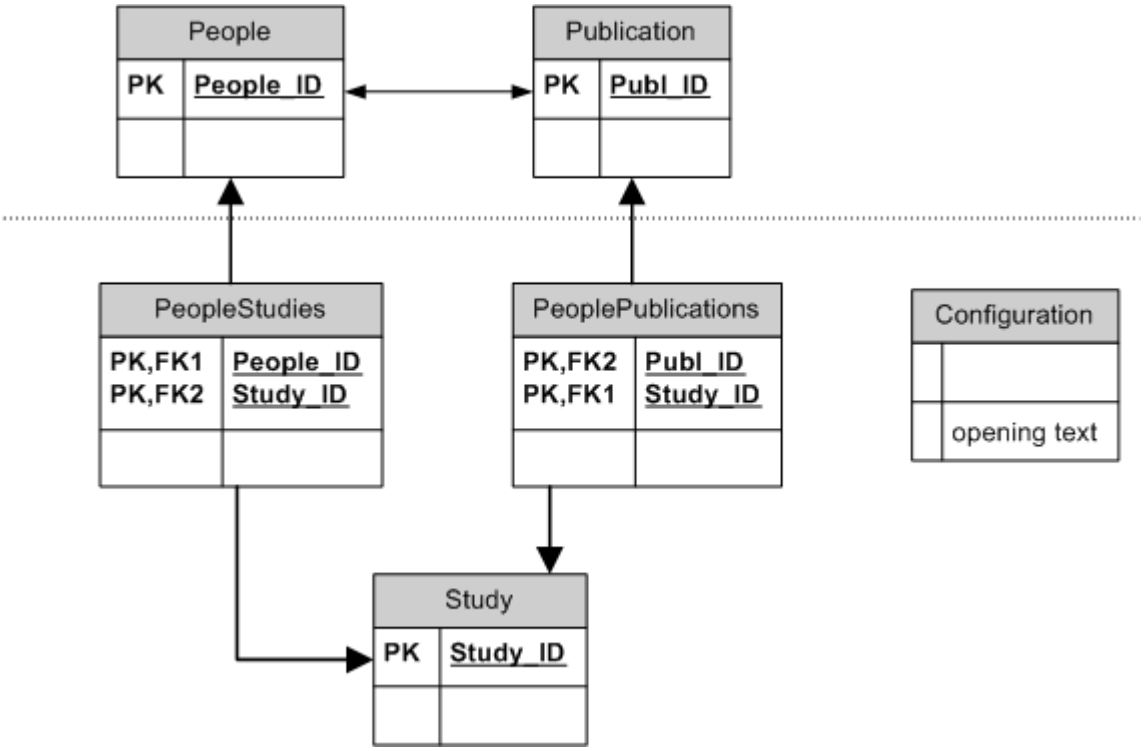
Use Case navn:	Slett Studie
Iterasjon:	1
Sammendrag:	Sletter studie fra database
Prebetingelse:	Innlogget
Postbetingelse:	Studie slettet fra database
Aktør:	Databaseadministrator, Studieadministrator
Hovedflyt:	1. Aktør logg inn 2. System lister opp studier sortert etter Studie Navn, Avsluttet Studie eller Studieeier 3. Aktør velger å slette studie 4. System ber om bekreftelse 5. Aktør bekrefter 6. System sletter studie 7. System klart for nye kommandoer
Alternativ flyt:	A-1: *. Aktør kan når som helt avslutte A-2: 5a. 1. Aktør avbryter sletting 2. Fortsett fra hovedflyt punkt 5
Forfatter:	
Dato:	

Tabell 7-2 Det opprinnelige use case'et "Rediger Studie" fra DES

Slett Studie består av tre transaksjoner og kategoriseres til enkelt i UCP-metoden. Punkt 1 til 2, 3 til 4 og 5 til 7 utgjør de tre transaksjonene. I FPA studeres hvert punkt i hovedflyten bedre;

- Punkt 1, Aktør logg inn, vurderes ikke

- Punkt 2 utgjør en ekstern spørring
- Punkt 3 utgjør ekstern inndata
- Punkt 4 utgjør ekstern utdata
- Punkt 5 utgjør kun svar på punkt 4 og er ikke en funksjon
- Punkt 6 utgjør handlingen i punkt 3 og er ikke en funksjon
- Punkt 7 er ikke en funksjon



Figur 7-1 Eksempel på datamodell[4] for DES

Use case'et i Tabell 7-2 beskriver to aktører;

- Databaseadministrator
- Studieadministrator

Aktørene er begge roller spilt av mennesker som kommuniserer med DES gjennom et grafisk brukergrensesnitt. I UCP-metoden vurderes aktørene til komplekse. Figur 7-1 viser et utkast av databasen i DES. Tabellene over den stiplede linjen er en allerede eksisterende database utenfor DES;

- Publications og People henger sammen og danner en ekstern grensesnittfil
- Tabellene inneholder et dataelement hver og vurderes til lav

Tabellene under den stiplede linjen i Figur 7-1 beskriver interne logiske filer i DES;

- PeopleStudies, PeoplePublications og Study tilsvarer en logisk gruppering og dermed en intern logisk fil
- I tillegg til 15 dataelementer definert i kravspesifikasjon inneholder

[4] Datamodellen er hentet fra leverandør A sitt tilbud

tabellene 3 dataelementer og kun et lagringselement, og vurderes til lav
- Configuration utgjør en logisk gruppering og er en intern logisk fil
- Tabellen har et dataelement og vurderes til lav

Totalt gir dette;

```
(2 komplekse aktører * 3) + (1 enkelt use case * 5) = 11 UUCP,  og

(1 EI * 3) + (1 EU * 4) + (1 ES * 3) + (2 ILF * 7) + (1 EIF * 5) =
29 UFP. (For forkortelser se avsnitt 3.3 eller kapittel 9.)
```

7.1.4 Webobjekt

Første del av webobjekt er beregning av antall UFP. Deretter vurderer webobjekt fire aspekter verken UCP-metoden eller FPA ser på. For å vurdere disse kreves følgende kunnskap;

- Antall linker, html, XML, SQL og lignende som brukes til å
    - linke XML/HTML data
    - generere automatiske rapporter
    - integrere forhåndsdefinert logikk
    - integrere applikasjoner dynamisk og binde dem til databasen eller andre applikasjoner
- Antall multimediefiler med informasjon om alle typer multimediefiler nødvendig som lydfiler, videofiler, bilder og unike operasjoner som åpne, lukke, lagre, klipp ut
- Antall script som use case, script og unike skripteoperasjoner som åpne og navigering
- Antall webbyggesteiner som
    - active X, agenter, COM, OLE
    - align, center, draw, edit

Beregning av antall webobjekt krever kunnskap om filtyper, struktur, skripting og utforming av webapplikasjonen under utvikling. For en kunde uten særlig teknisk innsikt vil det være vanskelig å vurdere webobjekt riktig. En leverandør bør kunne lage riktige estimater. Allikevel vil det ved estimering i tidlig fase på bakgrunn use case mangle nødvendig kunnskap. Dette viste seg ved estimering av DES. Ingen av leverandørene hadde beskrevet elementer i brukergrensesnittet på det nivå WEBMO krever.

7.1.5 Bruk av ujusterte størrelsesfaktorer

For en kunde uten særlig teknisk innsikt men med klare ønsker for et fremtidig system kan UCP-metoden være til hjelp, når kunden skal velge leverandør. UCP-metoden krever ingen teknisk kunnskap om systemet under utvikling. Kunden trenger kun å detaljere sine ønsker til et nivå der samhandling mellom aktør og system vises i tillegg til å identifisere typen aktør. Slik vil UUCP kunne gi en god pekepinn på omfang, tid og eventuelle kostnader ved valg av leverandør.

En leverandør med god teknisk innsikt og god erfaring fra programvareutvikling vil også kunne benytte seg av UCP-metoden. UUCP vil gi en god pekepinn av systemets størrelse før ytterligere analyse er gjennomført. Det kan være en nødvendighet i noen tilfeller å trekke inn arkitektur i beregningen. Slik kan

leverandør enten bruke en tilpasset versjon av UCP-metoden eller benytte den sammen med FPA.

På grunn av krav til tekniske detaljer i FPA viser dette studiet at UUCP er bedre i bruk til estimering i tidlig fase. Jeg har funnet at;

- Den samme kunden som ovenfor har ikke riktige forutsetninger for å estimere UFP
- Leverandøren bør ha de riktige forutsetningene, men det er ikke sikkert at de kjenner systemet godt nok ved estimering av UFP i tidlig fase

7.2 Vurdering av kostnadsdrivere

Som tidligere forklart er sammenlikningen av modellenes kostnadsdrivere avgrenset til å gjelde UCP-metoden og COCOMO II. UFP benyttes i COCOMO II som størrelsesmål og selve timeestimatet er et resultat av formlene og kostnadsdriverne i COCOMO II. Hensikten med å sammenlikne UCP-metoden med FP i avsnitt 7.1 og COCOMO II i dette avsnittet er at FP og COCOMO II er atskillig mer utbredte modeller. Det er derfor en antagelse for denne oppgaven at UCP-metoden kan forbedres ved en systematisk sammenlikning med FP og COCOMO II. Dette gjør det også nyttig å se på når de ulike modellene er best egnet.

7.2.1 Tekniske faktorer

Tabell 7-3 gir en oversikt over tekniske faktorer i UCP-metoden med paralleller til COCOMO II. Tre tekniske faktorer i UCP-metoden vurderer delvis samme områder innenfor programvareutvikling som tre faktorer i COCOMO II. For hver felles teknisk faktor og hver teknisk faktor jeg finner relevant for webprosjekter, presenterer jeg et forslag for hvordan den tekniske faktoren kan vurderes i UCP-metoden.

7.2.1.1 Felles tekniske faktorer

Kompleks prosessering

I UCP-metoden vurderes kompleks prosessering ut fra typen use case, det vil si typen og antall transaksjoner. Vurderingen er subjektiv, og vurderer den graden faktoren påvirker utviklingen etter følgende skala, som er den samme for samtlige tekniske faktorer i UCP-metoden;

- 0 - Irrelevant for utviklingen
- 1 - Liten innflytelse på
- 2 - Moderat innflytelse
- 3 - Gjennomsnittlig innflytelse
- 4 - Betydelig innflytelse
- 5 - Essensielt for utviklingen

COCOMO II vurderer kompleksitet og stabilitet i;

- Databasestørrelse i liten, moderat, stor og veldig stor database
- Produktets kompleksitet i veldig enkel, enkel, noe komplekst, moderat, komplekst, veldig komplekst og ekstremt komplekst system
- Vektlegging av driftssikkerhet og dokumentasjon i veldig lite, lite, noe, vanlig, sterk, veldig sterk og ekstrem vektlegging

For denne faktoren er også COCOMO II sin vurdering noe subjektiv. Allikevel er den ikke så fleksibel som UCP-metodens vurdering. Når kompleksitet vurderes i UCP-metoden kan tips til vurderingsområder hentes fra COCOMO II;

- 0 – 1 behov for en lite kompleks datamodell, lite teknisk dokumentasjon, detaljert use case på nivå der hoveddelen av use case utgjør CRUD-funksjoner
- 2 – 3 behov for noe større lagringsplass, noe mer teknisk dokumentasjon, use case er hevet opp et nivå til administrasjons use case med inkluderte eller utvidede CRUD funksjoner, og flere andre funksjoner
- 4 – 5 behov for mye lagringsplass, mye teknisk dokumentasjon og store use case der hovedfunksjonaliteten utgjør avansert funkjsonalitet

Gjenbrukbar kode

Begge modellene krever informasjon om den graden utviklingen påvirkes av gjenbruk. I COCOMO II vurderes typen gjenbrukbarhet etter følgende skala;

- Ingen gjenbruk
- Gjenbruk på tvers av prosjekter
- Gjenbruk på tvers av programmer
- Gjenbruk på tvers av produktlinje
- Gjenbruk på tvers av multiple produktlinjer

I UCP-metoden vurderes den grad gjenbruk påvirker utviklingen etter skalaen vist under kompleks prosessering. Her kan UCP-metoden benytte omtrent samme skala som COCOMO II;

- 0 – 1 tilsvarer ingen gjenbruk
- 2 tilsvarer gjenbruk på tvers av prosjekter
- 3 tilsvarer gjenbruk på tvers av programmer
- 4 tilsvarer gjenbruk på tvers av produktlinje
- 5 tilsvarer gjenbruk på tvers av multiple produktlinjer

Flyttbarhet

Den siste tekniske faktoren som er felles for UCP-metoden og COCOMO II er flyttbarhet. COCOMO II sin faktor plattformvansker vurderer blant annet plattformens stabilitet. Plattformvansker vurderes etter;

- Plattform stabilitet i veldig stabil, stabil, noe ustabil, ustabil og meget ustabil
- Tid og lagrings faktor i under 50%, 65%, 80% og 90% av total eksekveringstid og minneplass

I UCP-metoden kreves kunnskap hvor stor påvirkning en eventuell flytting av systemet mellom plattformer, har på utviklingen. Her anbefaler jeg å vurdere COCOMO II sitt område, stabilitet i plattform, i tillegg til den påvirkningen en eventuell flytting vil ha på utvikling;

- 0 – 1 tilsvarer veldig stabil plattform, utviklingen trenger ikke ta hensynt til en mulig flytting
- 2 – 3 tilsvarer noe ustabil plattform, utviklerne må ta hensyn til en mulig flytting 1 til 2 ganger per år
- 4 – 5 tilsvarer ustabil plattfom, utviklerne må ta hensyn til en mulig flytting oftere enn 2 ganger per år

UCP-metoden	COCOMO II
Distribuert system Mengden distribusjon Hva distribueres	
Responstid Krav til responstid i forhold til normal responstid	
Sluttbrukereffektivitet Navigasjon	
Kompleks prosessering Komplekse use case	*Stabilitet og kompleksitet* Produktets kompleksitet Databasens størrelse Vektlegging av dokumentasjon og driftssikkerhet
Gjenbrukbar kode Andel gjenbrukbar kode	*Gjenbruk* Hvordan gjøre gjenbruk, på kryss av prosjekter, programmer etc.
Enkelt å installere Grad av installasjonsvanskeligheter	
Brukbarhet Lett forståelig system	
Flyttbarhet Mulighet for at systemet må flyttes mellom plattformer	*Plattformvansker* Plattformens stabilitet Eksekveringstid og lagringsplass
Enkelt å forandre Om programmet skal kunne forandres av bruker	
Flerbruk Om flere har tilgang til systemet samtidig	
Sikkerhetsregler Spesifikke sikkerhetsregler	
Tilbyr tilgang for tredje part Skal programmet tilby bibliotek eller lignende for andre systemer?	
Brukeropplæring Grad av brukeropplæring	

Tabell 7-3 Krav til kunnskap om tekniske faktorer i UCP-metoden med paralleller til COCOMO II

7.2.1.2 Tekniske faktorer til bruk i DES og andre webapplikasjoner

Ikke alle de tekniske faktorene hadde relevans for utviklingen av DES. Tekniske faktorer som er med på å gi ulikt estimeringsresultat leverandørene imellom er (UCP-metoden / COCOMO II);

- Gjenbrukbar kode / Gjenbruk
- Enkelt å forandre / Kompleksitet og stabilitet
- Enkelt å installere

Tabell 6-6 i avsnitt 6.2.3 for UCP-metoden og Tabell 6-16, Tabell 6-17 i avsnitt 6.4 for COCOMO II viser at to eller flere av leverandørene har vurdert disse faktorene ulikt. Slik finner jeg faktorene ekstra relevante og ekstra utslagsgivende ved estimering av webbaserte systemer. I tillegg til disse faktorene er andre tekniske faktorer vurdert spesielt for DES, men disse er vurdert like for de fire leverandørene. Dette gjelder;

- Fasiliteter for brukeropplæring
- Tilbyr direkte tilgang til tredje parter

Disse faktorene er med på å gjøre estimatene spesielle for DES, og at de vil kunne vurderes annerledes for andre webapplikasjoner. Grunnen til dette er DES sin kravspesifikasjon som spesifiserer at i dette tilfellet skal brukeropplæring være unødvendig og at DES ikke skal tilby tilgang for tredje parts systemer. I andre websystemer vil dette være faktorer som kan være bestemt annerledes. I tillegg er faktorene;

- Sluttbrukereffektivitet og
- Brukbarhet

to faktorer jeg alltid vil vurdere som viktige faktorer, og bør vurderes høyt i alle typer websystemer.

De resterende tekniske faktorene virker ikke særlig tilpasset utvikling av webapplikasjoner. De er antageligvis tenkt på utvikling av annen type programvare eller programvare fra tiden Karner (1993) la frem UCP-metoden. Dette viser også generelt at de tekniske faktorene som er felles for UCP-metoden og COCOMO II utgjør viktige faktorer for webutvikling.

7.2.1.3 Vurdering av UCP-metodens resterende relevante tekniske faktorer

Jeg foreslår følgende nivåvurdering av de tekniske faktorene jeg finner relevante i webutvikling;

Enkelt å installere
Her bør det tas hensyn til hvordan innstalleringen skal skje;

- 0 – 1 Tjener, database eller andre nødvendigheter er allerede installert, filer kan lastes rett ut, testmiljø er likt som kjørende miljø
- 2 – 3 Utviklerne gjør all installering selv, må sette opp test- og kjørende miljø
- 4 – 5 Webprogrammet må lages slik at andre enkelt kan ta seg av innstalleringen

Fasiliteter for brukeropplæring
Her bør brukere og systemets hensikt vurderes;

- 0 – 1 Systemet skal være selvforklarende med stor, målgruppe der brukeropplæring skal være unødvendig
- 2 – 3 Systemet har en snevrere målgruppe og en administrasjonsdel der enkel brukeropplæring for administrasjonen må gjennomføres
- 4 – 5 Systemet har en bestemt målgruppe og utfører noe mer avanserte oppgaver, samtlige brukere trenger en innføring i bruk

Tilbyr direkte tilgang til tredje parter
Her bør typen tredje parts systemer vurderes;

- 0 – 1 Ingen tredje parter skal ha tilgang til systemet
- 2 – 3 Andre websystemer skal ha tilgang
- 4 – 5 Alle mulige systemer skal ha tilgang

Sluttbrukereffektivitet
Her bør typen system og typen brukere vurderes;

- 0 – 1 Ingen hensyn til sluttbrukereffektivitet, ekstra avansert system og brukeregruppe
- 2 – 3 Noe hensyn til sluttbrukereffektivitet, mindre avansert system og brukergruppe
- 4 – 5 Stort hensyn til sluttbrukereffektivitet, systemet krever enkel og logisk oppbygging, og navigasjon skal være lett for alle typer mennesker

Brukbarhet
Her bør typen system og typen brukere vurderes;

- 0 – 1 Ingen hensyn til brukbarhet, system og brukere er ekstra avansert og skal ha grundig opplæring
- 2 – 3 Noe hensyn til brukbarhet, system er noe mindre avansert og kun noe brukeropplæring gjennomføres
- 4 – 5 Brukbarhet er ekstra viktig, hvem som helst kan bruke systemet, og det må være lett å forstå for alle

7.2.1.4 Vurdering av UCP-metodens mindre relevante tekniske faktorer

Webutvikling skiller seg noe fra tradisjonell systemutvikling [Reifer 2002]. Jeg har funnet at flere tekniske faktorer i UCP-metoden kun til et visst nivå er relevante i webutvikling;

- Distribuert system – en webapplikasjon er et enkelt distribuert system, distribuerte databaser eller andre distribuerte tjenester vil imidlertid regnes som en støtteaktør i use case diagrammet og tas hensyn til i vektlegging av aktører, vurderingene blir i de fleste tilfeller 0 – 2
- Responstid – vil i de fleste tilfeller avhenge av Internet hastigheten og tjeneren og ikke av gode programmeringsløsninger, vurderingene blir i de fleste tilfeller 0 – 2
- Flerbruk – et webbasert system vil alltid brukes av flere mennesker, dermed er tjener oppsatt for dette. Problem med at flere endrer data samtidig løses vanligvis med enkle standard metoder i programmerings-språk, vurderingen blir i de fleste tilfeller 0 – 2
- Sikkerhetsregler – ofte ligger oppstarts-, sikkerhetskopierings- og gjenopprettelsesrutiner allerede klart. Dette må sjeldent programmeres inn eller tenkes på separat ved utvikling av websystemer, vurderingen blir i de fleste tilfeller 0 – 2

Dett kan tyde på at Karner (1993) definerte tekniske faktorer med tanke på større og mer komplekse systemer enn det webprogrammer vanligvis er. Dette gjør at vurderingene for de fleste tekniske faktorer generelt er gjennomsnittlig og lavere, noe som igjen er med på å redusere timeestimatet.

7.2.2 Omgivelsesfaktorer

Tabell 7-4 viser en oversikt over omgivelsesfaktorene i UCP-metoden med paralleller til COCOMO II. Dersom ikke annet er oppgitt, bruker jeg Ribu (2001) som grunnlag for vurderingene av omgivelsesfaktorene i UCP-metoden. Fem omgivelsesfaktorer i UCP-metoden berører delvis samme områder innenfor programvareutvikling som fire omgivelsesfaktorer i COCOMO II.

UCP	COCOMO II
Kjennskap til utviklingsprosess Kjennskap og erfaring med utviklingsprosess	*Prosessmodenhet* CMM nivå
Applikasjonserfaring Tidligere erfaring med tilsvarende applikasjoner	*Personellets erfaring* Antall års erfaring med tilsvarende applikasjoner, plattform, språk og verktøy
Objektorientert erfaring Erfaring med objektorientering	*Personellets kompetanse* Teamets kompetanse i analyse og programmering i prosent av best mulig kompetanse Prosentvis fravær i bedriften
Analysekompetanse Erfaring med analysearbeid	
Motivasjon Graden av motivasjon hos utviklerne	*Team kohesjon* Teambygging, erfaring i å operere som et team, vilje og motivasjon, konsistens i utviklernes kultur og synspunkt
Stabile krav Kjennskap til brukerne med tanke på stabilitet Grad av stabilitet	
Deltidsarbeid Antall deltid og heltid i utviklingen	
Vansker med programmeringsspråk Erfaring med programmering	

Tabell 7-4 Krav til kunnskap om omgivelsesfaktorene i UCP-metoden med paralleller ti COCOMO II

7.2.2.1 Kjennskap til utviklingsprosess

I UCP-metoden vurderes kjennskap til utviklingsprosess etter følgende skala;

- Utviklerne er ukjent med utviklingsprosess
- Utviklerne har teoretisk kjennskap til utviklingsprosess
- En eller flere utviklere har brukt utviklingsprosessen en eller flere ganger
- Minst halvparten av utviklerne har erfaring i bruk av utviklingsprosessen
- Alle utviklerne har brukt utviklingsprosessen i flere ulike prosjekter

COCOMO II vurderer bedriftens prosessmodenhet etter CMM-nivå. Bedriftens CMM-nivå avgjør vurderingen, der CMM-nivå 1 gir lavest vurdering, CMM-nivå 2 gir nest laveste vurdering, og så videre.

7.2.2.2 Applikasjonserfaring

Begge modellene krever informasjon om utviklernes erfaring. UCP-metoden vurderer antall års erfaring i utvikling av liknende applikasjoner. COCOMO II trekker inn flere faktorer. Dette gjelder utviklernes gjennomsnittlige antall års erfaring med;

- Tilsvarende applikasjonsutvikling
- Plattform
- Programmeringsspråk
- Verktøy

UCP-metodens omgivelsesfaktor vansker med programmeringsspråk vurderer også antall års erfaring med programmering generelt.

7.2.2.3 Objektorientert erfaring og analysekompetanse

De to neste omgivelsesfaktorene i UCP-metoden er objektorientert erfaring og analysekompetanse. Begge vurderes ut fra utviklernes generelle antall års erfaring med objektorientering og analyse. COCOMO II sin faktor personellets kompetanse vurderer analyse og objektorientert erfaring litt annerledes. Her vurderes utviklergruppens generelle evne til kommunikasjon, samarbeid, analyse, design, programmering og effektivitet. Dette vurderes prosentvis av best mulige evner. I tillegg vurderes den totale fraværsprosenten i bedriften under denne faktoren.

7.2.2.4 Motivasjon

Siste felles faktor er motivasjon. I UCP-metoden vurderes hvorvidt utviklerne har lav motivasjon, er interesserte i å gjøre en god jobb, eller er ekstra motiverte og inspirerte. Faktoren team kohesjon i COCOMO II berører så vidt området motivasjon. Teamkohesjon vurderer hvorvidt;

- Utviklerne har erfaring i å operere som et team
- Det gjennomføres teambygging for å oppnå delt visjon og forbindelser
- Det er konsistens i utviklernes synspunkter og kultur
- Utviklerne kan håndtere andre utvikleres synspunkter

7.2.2.5 Omgivelsesfaktorene til bruk i DES og annen webutvikling

Omgivelsesfaktorer som er med på å gi ulikt estimeringsresultat leverandørene imellom er (UCP-metoden / COCOMO II);

- Kjennskap til utviklingsprosess / Prosessmodenhet
- Applikasjonserfaring, vansker med programmeringsspråk / Personellets erfaring
- Objektorientert erfaring / Personellets kompetanse
- Analysekompetanse (kun i UCP-metoden)
- Motivasjon / Team kohesjon
- Forståelse (kun i COCOMO II)
- Utviklingsfleksibilitet (kun i COCOMO II)

Tabell 6-7 i avsnitt 6.2.4 for UCP-metoden og Tabell 6-16, Tabell 6-17 i avsnitt 6.4 for COCOMO II viser at to eller flere av leverandørene har vurdert disse faktorene ulikt. Slik finner jeg faktorene relevante og ekstra utslagsgivende ved estimering av DES, og også andre webbaserte systemer. Det er også tidligere kjent at faktorer som omhandler kompetanse og erfaring gir utslag på

utviklingstid. Intervjuene som var utgangspunkt for vurderingene av omgivelsesfaktorer ga et helhetsinntrykk av noe ulik kompetanse og erfaring, derav ulike vurderinger. Intervjuene har også blant annet fokusert på kjennskap til utviklingsprosess og hvordan den er fulgt samt motivasjonen hos utviklerne.

I tillegg til disse faktorene er også resterende omgivelsesfaktorer i UCP-metoden vurdert spesielt for DES. I dette prosjektet er de blitt vurdert like for de fire leverandørene men kan vurderes annerledes for andre webapplikasjoner;

- Da utviklingen av DES var grunnlag for forskning var det fra begynnelsen av gitt at kravene skulle være relativt stabile. I et annet utviklingsprosjekt er kunden annerledes og det er ikke på forhånd gitt en så detaljert kravspesifikasjon
- Antall deltidsarbeidende vil også kontinuerlig variere fra prosjekt til prosjekt, og dermed få ulik vurdering

7.2.3 Bruk av kostnadsdrivere

Det at en kostnadsdriver i UCP-metoden også vurderes i COCOMO II støtter opp om at den er relevant for programvareutvikling. Av de tekniske faktorene som er like eller relevante for webutvikling, gir COCOMO II flere gode innspill til viktige områder som kan vurderes også i UCP-metoden. Selv om COCOMO II kun vurderer 11 kostnadsdrivere, mens UCP-metoden vurderer 21, dekker COCOMO II flere aspekter ved programvareutvikling;

- Hver kostnadsdriver i UCP-metoden vurderer kun et område
- Foruten gjenbruk vurderer hver kostnadsdriver i COCOMO II mellom to og fire områder etter forhåndsdefinerte kriterier

UCP-metoden er enklere å benytte siden ingen detaljkunnskap kreves. COCOMO II derimot krever noe detaljkunnskap om samtlige områder i kostnadsdriverne. UCP-metoden er mer fleksibel. Dette viser at vurderinger kan tilpasses prosjekter og organisasjoner ettersom utviklere blir kjent med metoden. De forhåndsbestemte kostnadsdriverne i COCOMO II gjør metoden tyngre og mer omfattende å bruke.

Fleksibiliteten i UCP-metoden kan gjøre det vanskelig å vurdere alle kostnadsdriverne riktig. I den forbindelse har Ribu (2001) foreslått kriterier for vurdering av omgivelsesfaktorene. Dette er veiledende kriterier som vurderer omgivelsesfaktorene i UCP-metoden på en liknende måte som omgivelses-faktorene fra COCOMO II. Jeg har fulgt dette videre, og foreslått fleksible, men veiledende kriterier for vurdering av relevante tekniske faktorer. Ribu (2001) foreslår også å utelate de tekniske faktorene i UCP-metoden, noe som også gjenspeiles av Merrick (2005c). Ovenfor diskuterte jeg muligheten med en egen versjon av UCP-metoden til bruk for kunder;

- Jeg anbefaler at kunder uten teknisk innsikt utelater teknisk faktor og muligens omgivelsesfaktor for dermed å benytte UCP-metoden liknende Merrick (2005c) sitt forslag, dette gjør metoden enklest mulig
- Jeg anbefaler leverandører å inkludere teknisk faktor for estimering av websystemer på nivå med DES da utelatelse av teknisk faktor øker estimatene med omkring 100 timer, og utgjør relativt stor forskjell i et websystem på størrelse med DES

Når det gjelder COCOMO II har jeg funnet at;

- En kunde uten særlig teknisk innsikt ikke vil ha riktig forutsetninger for å gi

en god vurdering av kostnadsdriverne

- En leverandør bør ha nok kunnskap om systemet under utvikling til å gi en god vurdering av kostnadsdriverne, men noe analyse må gjøres først
- UCP-metoden egner seg bedre til tidlig estimering med utgangspunkt i en enkel kravspesifikasjon der funksjonelle krav er beskrevet i use case

Til slutt setter jeg opp en liten oversikt over de faktorene COCOMO II vurderer som UCP-metoden ikke berører, se Tabell 7-5.

COCOMO II
Risikovurdering Risikoledelse Tid og budsjett Prosent av utvikling for å etablere arkitektur Nivå på usikkerheter Antall kritiske risikoelementer
Utviklingsfleksibilitet Eksterne grensesnitt spesifikasjoner Forhåndsdefinerte krav Behov for tidlig ferdigstillelse
Forståelse Organisasjonens forståelse av produktet Erfaring med relaterte systemer Behov for algoritmer eller prosesserende arkitektur
Tidspress Prosentvis tidspress i forhold til et normalt prosjekt
Fasiliteter Type kommunikasjon mellom utviklere og lokasjon

Tabell 7-5 Krav til kunnskap om kostnadsdrivere i COCOMO II som ikke gjenspeiles i UCP-metoden

7.3 Innflytelse på estimeringen

Estimeringsmetodene varierer noe når det gjelder hvor mye innsats som skyldes størrelse og hvor mye innsats som skyldes kostnadsdrivere. Ved å flytte vurderingen av en kostnadsdriver ett nivå opp eller ned;

- økes/reduseres estimatet med mellom 10 og 60 timer i UCP-metoden
- økes/reduseres estimatet med mellom 100 og 200 timer i COCOMO II
- økes/reduseres estimatet med mellom 100 og 300 timer i WEBMO

For et prosjekt som DES der utviklingstiden er under 1000 timer, gjør 200 - 300 timer stor forskjell. Dersom COCOMO II eller WEBMO skulle estimere et prosjekt på omtrent 10 000 timer vil ikke 200 - 300 timers differanse ha den samme effekten. Det skal sies at antallet timer, ved endring av kostnadsdrivers vurdering når et prosjekt tar rundt 10 000 timer, er høyere.

Ved estimering av et lite system som DES viser dette studiet at følgende avgjør den største delen av estimatene;

- UCP-metoden – Antall UUCP
- COCOMO II – Kostnadsdrivere
- WEBMO - Kostnadsdrivere

For UCP-metoden ser vi fra avsnitt 6.2 at UCP_1 estimerer rundt 1 000 arbeidstimer for alle leverandørene, med en variasjon på 70 timer. Her estimeres med identiske use case. Ved å inkludere alle use case, $UUCP_2$, eller restrukturere use case'ene, $UUCP_3$, endres estimert timeantall betraktelig. Dette støtter opp om at use case'enes antall og struktur har størst innflytelse på estimering av innsats.

Med unntak av leverandør A sitt estimat på omtrent 500 timer, ligger estimatene fra WEBMO i overkant av 1000 arbeidstimer. Siden leverandør A estimerer et høyere antall webobjekt enn leverandør D er ikke størrelsesmålet utslagsgivende faktor. Derimot har kostnadsdriverne hos leverandør A generelt blitt vurdert lavest. Dette gir et lavere estimat, og støtter min vurdering om at kostnadsdriverne i WEBMO har større innflytelse på estimert innsats enn antall webobjekt, i et system som DES.

En forholdsregel er at ett nivå uriktig vurdering av en kostnadsdriver for COCOMO II eller WEBMO, kan føre til en urettferdig vurdering av modellene. Siden forskjellen mellom to nivåer i UCP-metoden ikke har samme påvirkning vil estimatene fra UCP-metoden være sikrere. Uansett har jeg vurdert leverandørene ut fra like premisser, slik at forholdet leverandørene imellom er riktig.

Jeg tolker denne variasjonen mellom UCP-metoden og COCOMO II/WEBMO som et tegn på at COCOMO II og WEBMO er modeller beregnet på utvikling av noe større systemer. For større systemer vil antageligvis antall kildekodelinjer avgjøre estimatene i COCOMO II og WEBMO.

7.4 Estimeringsmodellene

7.4.1 UCP-metoden

Leverandørene har lagt til et ulikt antall use case; alt fra ett til seks. Use case'ene lag til er også detaljert i ulik grad.

Leverandør A;

- Lagt til ett use case uten beskrivelse
- Beskriver de andre use case'ene i form av storyboards med vekt på elementer i brukergrensesnitt

Leverandør B;

- Lagt til seks use case
- Beskriver kun use case'ene mål med vekt på presentasjon av data

Leverandør C;

- Lagt til fire use case
- Beskriver use case'ene bevisst strukturert på nivå med use case'ene fra kravspesifikasjonen, vekt på gangen i use case'ene

Leverandør D;

- Lagt til to use case
- Beskriver use case'enes mål med vekt på struktur i skjermbilder

Leverandør C har mest detaljerte og strukturerte use case. De andre leverandørene fokuserer mer på elementer tilhørende brukergrensesnittet. Dette kan tyde på at leverandør C med god struktur har kommet lengst i analysefasen ved nedskriving av tilbudene. Konsekvensen er mer detaljerte use case som igjen gir høyere estimater. Dette kan føre til noe for høye estimater, da analysefasen nesten er over og utviklerne kan raskere sette i gang implementasjonen. I slike tilfeller kan en alternativ metode i UCP-metoden, tilsvarende de tre metodene i COCOMO II, være til hjelp. Dette kommer jeg tilbake til i avsnittet om videre arbeid.

Resultatene fra dette studiet viser flere faktorer som påvirker estimert timeantall i UCP-metoden;

- Antall use case
- Antall inkluderte og utvidede use case
- Antall transaksjoner per use case

7.4.1.1 Timefaktor i UCP-metoden

Karner (1993) anbefaler en timefaktor på 20 timer per UUCP. I dette studiet er Schneider og Winters (2001) sin metode benyttet, som her resulterte i 20 timer per UCP. Hadde Schneiders og Winters (2001) metode resultert i en høyere timefaktor ville estimatene blitt betraktelig for høye. Spørsmålet er om det kunne vært mulig å bruke en lavere timefaktor i estimering av webprosjekter når use case'ene er forholdsvis godt detaljert. Ribu (2001) mener at timefaktoren bør velges på grunnlag av typen bedrift. Faktorer som kvalitet og prosjektstyring er viktige i timefaktoren. Estimeringsresultatene fra dette studiet tyder på at;

- UCP-metoden legger vekt på god kvalitet og prosjektstyring i estimater av websystemer
- At 20 timer per UUCP gjelder alle mindre websystemer med godt strukturerte use case

7.4.2 COCOMO II

Boehm et al. (2000c) forklarer at et av målene med COCOMO II var å forenkle definisjonene av kostnadsdriverne. Etter å ha laget estimater i COCOMO II oppfatter jeg fortsatt at noen av kostnadsdriverne er vanskelige å sette. Dette gjelder blant annet ved vurdering av forståelse, prosentvis fravær og personellets kompetanse;

- Disse faktorene målt i prosent er vanskelig å vurdere for en ekstern person

7.4.2.1 Størrelsesfaktor i COCOMO II

Boehm et al. (2000c) anbefaler å bruke en faktor på 20 kildekodelinjer per UFP for fjerdegenerasjonsspråk. Reifer (2002) benytter en noe justert omregnings-tabell, se Vedlegg D, tilpasset dagens webprogrammering. For objektorientert webprogrammering gjelder en faktor på 25 kildekodelinjer per UFP. Da DES er en typisk objektorientert webapplikasjon har jeg valgt å benytte denne faktoren ved estimering av DES. Hadde jeg brukt faktoren 20 kildekodelinjer per UFP ville estimatene blitt redusert med 150 til 200 timer for alle leverandørene. Dette hadde ført til estimater på nivå med leverandør C sin utviklingstid.

7.4.3 WEBMO

I WEBMO brukes samme antall kildekodelinjer per webobjekt som antall kildekodelinjer per UFP i COCOMO II. Siden WEBMO teller flere elementer fører dette naturlig til et høyere antall kildekodelinjer, som igjen resulterer i høyere estimert arbeidstimer. En grunn til dette kan være at WEBMO antar en generelt høyere kvalitet enn COCOMO II. Mer om dette i neste avsnitt.

7.5 Kvalitetsaspektet

Først en oppsummering av ekspertenes estimater sammen med faktisk brukt tid og estimeringsmodellenes estimater.

Lev	Faktisk brukt tid	Ekspert estimat	UCP_3	COCOMO II	WEBMO
A	582	250	625	696	553
B	953	341	949	898	1035
C	345	100	760	713	1046
D	849	650	665	1001	1186

Tabell 7-6 Estimater hentet fra avsnitt 6.7

Tabell 7-6 viser oversikt over alle estimater. Studiet har gitt følgende resultat;

- Leverandør A estimeres bra med UCP-metoden og COCOMO II. For begge, bedre enn ekspert
- Leverandør B estimeres bra med UCP-metoden og COCOMO II. For begge bedre enn ekspert
- Leverandør C overestimeres kraftig med UCP-metoden og COCOMO II. For begge, bedre ekspert
- Leverandør D underestimeres noe med UCP-metoden, omtrent som ekspert, COCOMO II overestimerer tilsvarende
- WEBMO estimerer bra for leverandør A og B, overestimerer noe for leverandør D, og overestimerer kraftig for leverandør C, bedre en ekspert for leverandør A, B og D

Leverandørene i dette studiet har utviklet samme funksjonalitet. Allikevel varierer kvalitet, tidsbruk og antall kildekodelinjer. Dette viser at tid som brukes på utviklingsprosjekter ikke kun avhenger av den funksjonaliteten som skal utvikles. Generelt tenker leverandørene ulikt ved følgende faktorer;

- Gjenbruk
- Enkelt å forandre
- Dokumentasjon
- Kjennskap til og bruk av utviklingsprosess

Fra intervjuene ser leverandør B og D ut til å legge mest vekt på disse faktorene, leverandør A noe mindre og leverandør C minst. Dette studiet viser at;

- Leverandørenes ønske om kvalitet i utvikling og på ferdig produkt gjenspeiles i vurderingen av disse faktorene

De tre estimeringsmodellene overestimerer betraktelig for leverandør C. Det kan være fordi leverandør C ser ut til å ha kommet lengre i analysearbeidet enn modellene antar. Mest tenkelig grunn er at leverandør C velger å levere det ferdige produktet med betraktelig lavere kvalitet som en minimumsløsning.

Forskjellen i utviklingstid hos leverandørene har ikke bare med kvalitet å gjøre, men også antall kildekodelinjer. En rask studie av kildekodelinjer, se Tabell 7-7, viser at leverandør B skriver nesten dobbelt så mye kildekode som de andre. Det er ikke gjennomført en nøyaktig analyse av hva slags kode det er snakk om. Allikevel betyr det at leverandør B antakelig har gjort en unødvendig stor innsats i utviklingen av DES. Dette gjenspeiles også i faktisk brukt timeantall.

Leverandør	LOC
A	7 937
B	14 549
C	7 208
D	8 293

Tabell 7-7 Antall kildekodelinjer per leverandør

Leverandørenes kodekvalitet er studert av to uavhengige personer. En forsker tilknyttet Simula samt en ekstern konsulent. Studiene er basert på følgende kriterier;

- Bruk av design og arkitekturmønstre
- Hvorvidt generelle programmeringsprinsipper er fulgt
- Rent kodetekniske ferdigheter
- Bruk av standard Java API'er, for eksempel Collections
- Dokumentasjon i koden
- Karaktergivning i en skala fra 1 til 6 der 1 er dårligste karakter

Leverandør A får karakteren 5 som er beste karakter gitt;

- God bruk av tolags arkitektur med MVC/struts og DAO
- Godt atskilt presentasjons og forretningslogikk

Med noen flere kildekodelinjer har leverandør D oppnådd samme karakter som leverandør A;

- God bruk av trelagsarkitektur med fornuftig bruk av MVC, DAO og andre design mønstre
- Godt atskilt presentasjons og forretningslogikk
- Bra kodedokumentasjon
- Enkelt å legge til ny funksjonalitet
- Lite bruk av tredjeparts biblioteker gjør koden lett å sette seg inn i

Leverandør B tildeles karakteren 3;

- Inneholder hjemmelagede rammeverk som er vanskelig å sette seg inn i
- Umotivert bruk av mønstre
- Mye utkommentert kode
- Mange ubrukte konstanter som er definert
- Riktig bruk av Collections og logging

Leverandør C sin løsning kommer klart ut som den svakeste med karakteren 2;

- Blandet presentasjons og forretningslogikk
- Java klasser er ikke inndelt i pakker
- Lite og dårlig dokumentasjon med en del ut utkommentert kode
- Vanskelig å utvide koden med ny funksjonalitet
- Uelegant bruk av formattering og parsing av datoer
- Fordelen med denne implementasjonen er at den er enkel og minimal, enkle feil kan rettes raskt

For leverandør C estimerer UCP_3 og COCOMO II i overkant av 700 arbeidstimer. Hadde leverandør C utviklet tilnærmet kodekvalitet som leverandør A og D burde faktisk brukt timeantall ligge på dette nivå.

- Dette studiet bekrefter at en webapplikasjon kan utvikles raskt som en minimumsløsning, på bekostning av god kvalitet
- Økt kvalitet tar lenger tid, og denne studien viser dette spesielt i forhold til vurderingene av ovennevnte kvalitetskriterier fra modellenes kostnads-drivere
- Studiet viser at måten estimeringen gjennomføres på kan avhenge av leverandørenes ønske om kodekvalitet. Ved eventuelt bevisst utvikling av lavere kodestandard kan det være mulig å benytte mer generelle og mindre detaljerte use case. Slikk UCP-metoden er nå vil dette redusere både UFP og UUCP. Slik kan UCP-metodens antagelser om kvalitet ved estimerings-tidspunktet planlegges

Kvalitetsanalysene i dette studiet, sammen med antall kildekodelinjer, estimater og faktisk brukt tid viser at;

- WEBMO antar ekstremt høy kodekvalitet
- UCP-metoden og COCOMO II antar en noe lavere kvalitet, men fortsatt meget høy kvalitet

I tillegg til ønsket om god kvalitet og utviklingsprosess påvirker mange andre faktorer utviklingstid. Dette studiet er en grov analyse av forskjeller utviklingsmodeller imellom, med hovedvekt på en overordnet analyse av kodekvalitet. En grundigere analyse vil være mer omfattende og tilhører fremtidig arbeid.

8 EVALUERING OG KONKLUSJON

8.1 Sammendrag

Hovedmålet med dette studiet er å teste UCP-metoden ved estimering av webapplikasjoner. Forutsetninger om kunnskapskrav og brukervennlighet i UCP-metoden sammen med estimatene, sammenliknes med funksjonspoeng, COCOMO II og WEBMO. Som grunnlag for denne sammenlikningen studeres utviklingen av en webbasert forskningsdatabase kalt DES. Fire svært ulike leverandører har utviklet hver sin versjon. Ved hjelp av kravspesifikasjon, leverandørenes tilbud, intervjuer og regneark med faktisk brukt timeantall ble studien gjennomført.

Ved å sammenlikne metodene viser det seg at de berører mange av de samme aspektene ved programvareutvikling. Allikevel vurderes aspektene ulikt. Generelt gir UCP-metoden rom for mer subjektive vurderinger enn de andre modellene som har strengere regler for vurdering. UCP-metoden tar utgangspunkt i use case ved å telle antall transaksjoner per use case. En bevisst struktur i use case'ene kan være med på å justere estimatene.

Generelt treffer estimatene beregnet ved hjelp av estimeringsmodellene bedre enn ekspertenes egne estimater. WEBMO overestimerer en del for alle leverandørene. UCP-metoden og COCOMO II estimerer bra for leverandørene som har utviklet god kvalitet.

8.1.1 Hovedforskningsspørsmål

En studie av use case poeng metodens estimater i forhold til eksperters estimater ved webutvikling. Hvor egnet er use case poeng metoden ved bruk til estimering av webbaserte programmer?
Dette studiet støtter tidligere resultater ved studier av UCP-metoden. Metoden kan være til god støtte for eksperter ved estimering av webutviklingsprosjekter, der god kvalitet på utvikling, dokumentasjon og kode er i fokus. Det ser også ut til at UCP-metoden kan benyttes av en kunde for å få hjelp til å velge programvareleverandør.

8.1.2 Underpunkter

Sammenlikning av informasjonskrav for størrelsesmålene use case poeng, funksjonspoeng og webobjekter samt use case poeng metodens og COCOMO II's kostnadsdrivere.
Estimeringsmodellene dekker delvis de samme aspektene innenfor programvareutvikling. UCP og FP tar begge utgangspunkt i funksjonelle krav fra et brukerperspektiv. Ut over det vurderes funksjonelle krav på ulike måter. FPA forutsetter betraktelig mer teknisk/faglig kunnskap enn UCP-metoden.

Det samme gjelder for modellenes kostnadsdrivere. De viktigste kostnadsdriverne i estimeringen, ser ut til å være felles for både UCP-metoden og COCOMO II. Disse kostnadsdriverne utgjør også grunnlaget for variasjonene i estimatene. Ut fra de felles kostnadsdriverne er det tydelig at kvalitetsaspektet

sammen med utviklernes erfaring, utgjør noen av de viktigste faktorene i programvareutvikling.

Felles omgivelsesfaktorer UCP-metoden / COCOMO II;

- Kjennskap til utviklingsprosess / Prosessmodenhet
- Applikasjonserfaring / Personellets erfaring
- Objektorientert erfaring og Analysekompetanse / Personellets kompetanse
- Motivasjon / Team kohesjon

Felles tekniske faktorer UCP-metoden / COCOMO II;

- Kompleks prosessering / Stabilitet og kompleksitet
- Gjenbrukbar kode / Gjenbruk
- Flyttbarhet / Plattformvansker

Studie av tekniske faktorer i use case poeng metoden med tanke på webbaserte programmer.

Et generelt overblikk over de tekniske faktorene i UCP-metoden viser at de fleste vurderes til gjennomsnittlig eller lavere. Etter en gjennomgang av de tekniske faktorene, kan det tyde på at noen hovedsakelig er laget for vurdering av større og mer omfattende systemer enn en enkel webapplikasjon. For de fire leverandørene er de tekniske faktorene kun med på å justere ned det estimerte timeantallet, med omtrent 100 timer. I dette studiet gir justeringen med teknisk faktor et bedre estimat.

Disse tekniske faktorene ser ut til å være noe relevante ved estimering av webapplikasjoner, for samtlige har jeg laget en grov veiledning for hvordan de kan vurderes;

- Sluttbrukereffektivitet
- Kompleks prosessering
- Gjenbrukbar kode
- Brukbarhet
- Enkelt å forandre
- Brukeropplæring
- Tilby tilgang til tredje parter
- Enkelt å installere
- Flyttbarhet

Disse derimot ser ut til å være noe overflødig;

- Distribuert system
- Responstid
- Flerbruk
- Sikkerhetsregler

Sammenlikning av kvaliteten på leverandørenes leverte og godkjente webapplikasjon i forhold estimeringsmodellenes estimater.

WEBMO ser ut til enten å anta meget høy kvalitet, eller å være for omfattende for utvikling av mindre webapplikasjoner. Imidlertid ser UCP-metoden og COCOMO II ut til å anta god kvalitet innenfor følgende aspekter;

- Dokumentasjon

- Utviklingsprosess
- Kodestandarder

Studien viser tydelig at webapplikasjoner kan utvikles raskt og billig på bekostning av disse faktorene, og som vanlig, tar god kvalitet lengre tid å utvikle.

8.1.3 Avgrensninger

DES er et lite system med lav kompleksitet. Dette gjør at resultatene fra dette studiet ikke kan generaliseres til å gjelde alle typer webutviklingsprosjekter. Resultatene i dette studiet antas kun å gjelde for estimering av;

- Systemer med lav kompleksitet, vedlikeholdssystemer der hovedmålet er lagre/endre/slette og hente informasjon
- Systemer med stabil kravspesifikasjon
- Systemer med kravspesifikasjon godt detaljert av kunder som vet hva de ønsker og hva som kan gå, men kanskje ikke er like gode på det tekniske
- Utvikling som krever små utviklingsteam, med en prosjektleder og 2 til 3 utviklere
- Systemer som krever kort utviklingstid, 2 til 3 måneder og 500 til 1000 arbeidstimer

Styrken ved studiet er at resultatene kommer fra en studie av fire ulike utviklingsbedrifter. Dette gjør studiet til en multippel case studie og styrker dermed konklusjonen. Imidlertid kan noen aspekter endre seg i større webprosjekter;

- DES var lite og hadde godt strukturert funksjonalitet. Dette gjorde restruktureringen av use case enkel. I en større case med større og mer komplekse use case vil use case antakelig være annerledes strukturert, noe som igjen kan føre til en annerledes restrukturering
- I et større prosjekt kan kostnadsdriverne ha en annen effekt og kanskje vurderes annerledes

Studiet er avgrenset til i hovedsak å gjelde sammenlikning av størrelsesmål, kostnadsdrivere og estimater.

- For størrelsesfaktorer sammenliknes kun use case poeng og funksjonspoeng. Webobjekt diskuteres generelt til slutt. Funksjonspoeng utgjør utgangspunktet for antall kildekodelinjer i COCOMO II og WEBMO og er dermed et naturlig utgangspunkt for sammenlikning med use case poeng
- For kostnadsdrivere sammenliknes kun kostnadsdriverne i UCP-metoden og COCOMO II. COCOMO II er valgt til sammenlikning med UCP-metoden fordi den er en stor, velkjent og utprøvd estimeringsmodell, og anses som et godt grunnlag for sammenlikning og videreutvikling av use case poeng
- Mange aspekter påvirker utviklingstid. I dette studiet er det lagt vekt på kvalitet i utvikling, kodeoppbygging og struktur

Av blant annet tidsmessige grunner er kun en modell valgt for utdypende sammenlikning med UCP-metoden. En grundig analyse av flere modeller hører med i større studier og videre arbeid. Estimeringsmodellen WEBMO er med i studiet, kun for å ha et tredje estimat til grunn i sammenlikningen av timeestimatene. WEBMO ble valgt fordi den er videreutviklet fra COCOMO II, med den hensikt å spesifisere en estimeringsmodell mot webutvikling.

8.2 Gjennomføring

8.2.1 Forskningsmetode

Dette studiet er en multippel case studie. Forskningsmetoden er analyse av tekster og prosjektdokumenter. All datainnsamling ble gjennomført høsten 2003 og jeg startet dette studiet våren 2004. En fordel her er at all data foreligger ved oppstart. Jeg vet omfanget og hva jeg har å gå ut ifra. Det eneste som må gjøres er å gjøre seg kjent med dataene. En annen fordel er at jeg går inn som en objektiv vurderer av leverandørenes estimering. Dette gjør studien rettferdig ovenfor de ulike leverandørene. Imidlertid foreligger mye data som ikke er tilpasset mine forskningsspørsmål. Det er en utfordring å skille ut de data nødvendig for mitt tilfelle.

Ved å gjennomføre studiet på denne måten jobber jeg som en ekstern person som studerer estimeringen av DES fra yttersiden. Dette fører til at jeg kun ser det som står i dokumentene. Det kan være en utfordring å tolke all tekst riktig. I tillegg kan jeg ikke kontakte leverandørene dersom jeg skulle mangle nødvendig informasjon. Denne informasjonen må dermed antas ut fra det som er tilgjengelig. Dersom jeg hadde deltatt i datainnsamlingen ville jeg blitt bedre kjent med leverandørene. Antakelig ville jeg fått noe annet inntrykk enn det jeg har fått gjennom tekstene.

Det faktisk brukte timeantallet var meg ikke kjent ved estimeringstidspunkt. Først etter ferdig første utkast av estimeringene ble jeg kjent med leverandørenes faktisk brukt timeantall.

8.2.2 Gjennomgang av kravspesifikasjon og tilbud

Kravspesifikasjonen ble brukt til å forstå DES. Den ble brukt som utgangspunkt for å vurdere funksjonalitet og kompleksitet. Tilbudene ble brukt til å forstå tenkemåten til de ulike leverandørene. Noen hadde laget detaljerte beskrivelser, mens andre skildret løsningen mer generelt. Leverandørenes løsningsskisser er med på å lage ulike estimater leverandørene imellom.

8.2.3 Gjennomgang av intervjuer

For å vurdere estimeringsmodellenes omgivelsesfaktorer gikk jeg igjennom intervjutranskripsjonene slik at jeg ble kjent med leverandørene og utviklerne. En vanskelighet med intervjuene var å vite hvilken informasjon som var riktig informasjon til bruk i estimeringsmodellene. Spesielt vanskelig var det for COCOMO II, WEBMO og FPA siden ingen av spørsmålene var rettet direkte mot disse metodene. Antakelig ble noen spørsmål stilt med tanke på omgivelsesfaktorene i UCP-metoden. Disse spørsmålene ble svart subjektivt. Utviklerne kunne blant annet uttrykke seg på denne måten; *"Han har noe erfaring med det, siden det er en naturlig del av jobben å starte med…"*. Vanskeligheten her ligger i å vurdere *"noe erfaring"* i henhold til skalaen i UCP-metoden. En av Silverman's (2001) åtte huskeregler påpeker det viktige i å unngå å behandle aktørenes synspunkter som hele sannheten. Her beskriver han den naive intervjuer som behandler svarene i et intervju som nok informasjon til å gjøre en kvalitativ analyse. I dette studiet brukes denne huskeregelen til å analysere svarene kritisk slik at kostnadsdriverne kan vurderes på best mulig måte. Dette førte til at intervjuene ble brukt som en rettledning for å danne et helhetsinntrykk av utviklingsteamene.

Selve datasorteringen ble gjort ved først å sette opp alle kostnadsdriverne i rekkefølge. Deretter ble sitater som passet under hver kostnadsdriver notert. Slik ble det gjort en helhetsvurdering for hver kostnadsdriver ut fra uttalelser i intervjuene.

Dersom jeg hadde fulgt utviklingen av DES fra begynnelsen og dermed deltatt i intervjuene kunne vurderingen av kostnadsdriverne blitt noe annerledes. Om dette hadde forbedret eller forverret estimatene er vanskelig å si. Ved deltakelse i intervjuer og observasjon kunne inntrykket av blant annet erfaring, teamkohesjon og utviklingsprosess blitt noe annerledes.

8.2.4 Gjennomgang av regneark

Som sammenlikningsgrunnlag for faktisk brukt tid og estimatene, brukte jeg et regneark der timer per use case per leverandør var ført. Regnearket ble også brukt til å finne timer per utvikler og totalt antall timer.

8.2.5 Hva har jeg lært om estimering

Dette studiet har lært meg å virkelig forstå hensikten ved estimering. Det er viktig for en kunde å vite omfanget av et utviklingsprosjekt, og å kunne forholde seg til dette. Fornøyde kunder oppnås blant annet ved å kunne levere et produkt innenfor gitt tidsramme og pris. Dermed er det en balansegang for leverandørene å gi best mulig pris til mest mulig realistisk virkelighet og samtidig vinne anbudsrunden.

Studiet har også gitt meg god innsikt i hvilke estimeringsmetoder som finnes, og hvordan jeg kan bruke disse i fremtiden til hjelp i mitt eget arbeid.

8.3 Subjektiv vurdering

Generelt er vurderingene gjort etter helhetsinntrykk av utviklingsteamene. Det å kun basere helhetsinntrykket på de tilgjengelige intervjuene kan føre til små feil-vurderinger. Faktorer som ikke direkte ble gjenfunnet i tekstene ble vurdert til normalt eller gjennomsnittlig. Mange faktorer av denne typen kan også føre til små vurderingsfeil.

I COCOMO II, FPA og WEBMO ble vurderingene gjort på grunnlag av forhånds-definerte kriterier. UCP-metoden vurderer faktorene etter påvirkning på utviklingen eller grad av erfaring. De vurderingene som er gjort i dette studiet preges av min oppfattelse om hva som er normalt i webutvikling. Det er vanskelig å si om dette stemmer overens med Karner sin oppfatning av program-vareutvikling i 1993.

Det eneste leverandørene har felles er kravspesifikasjonen utviklet av Simula. Ut fra den har leverandørene laget ulike løsninger på DES. Dette fremstilles ulikt hos alle leverandørene. Som resultat er detaljeringsnivået i tilbudene ulike. Noe av forskjellen estimatene imellom kan komme av ulikt detaljnivå i tilbudene.

WEBMO er en forholdsvis ny estimeringsmodell beskrevet av Reifer (2000). Den er foreløpig lite dokumentert. Siden jeg kun har funnet Reifer (2000) sin beskrivelse var den litt vanskelig å forstå. De andre modellene har spredd større interesse, både i opplæring av metodene og i forskning på metodene. Ved telling av de forskjellige attributtene i WEBMO, var det tydelig at leverandørenes tilbud

ikke var skrevet med tanke på WEBMO. Noen steder måtte jeg lage standarder tilpasset DES og som var lik for alle leverandørene.

I WEBMO er DES kategorisert til å være en webbasert informasjonstjeneste. Denne kategoriseringen avgjør konstantene i formlene for estimering. Graden av feilestimering ved vurdering til gal kategori er ikke vurdert i dette studiet.

Vurderingene som gjøres for å tilegne kostnadsdriverne riktig nivå er alle subjektive fra mitt perspektiv. Mye informasjon var tilgjengelig, noe som har hjulpet meg å få et inntrykk av utviklerne hos leverandørene. Dette fører til at en annen forsker antakeligvis ville fått andre estimeringsresultater, men disse resultatene trenger ikke å være mer riktig enn mine resultater.

8.4 Konklusjon

Resultatene av dette studiet viser at eksperter kan benytte UCP-metoden eller COCOMO II som støtte i estimering av webutviklingsporsjekter. Basert på at ekspertenes estimater skal være realistiske ser det ut til at ekspertene selv undervurderer webutvikling. Estimeringsmodellene treffer nærmere faktisk brukt timeantall enn ekspertenes egne estimater.

En estimeringsmodell for å lage et tidlig estimat bør være enkel å forstå, og det bør være enkelt å finne tilstrekkelig informasjon. Her vurderes to hovedområder;

- *Størrelsesfaktor*. COCOMO II omgjør UFP til antall kildekodelinjer. I beregningen av UFP kreves en god del teknisk innsikt samt planlegging av lagringsstruktur. Beregning av UUCP krever kun bevisst strukturering av mål eller ønsker ved systemet samt oversikt over hva eller hvem som skal kommunisere med systemet
- *Kostnadsdrivere*. UCP og COCOMO II berører mange av de samme områdene ved programvareutvikling. Imidlertid vurderes ikke områdene på samme måte. UCP-metoden gir rom for en mer subjektiv vurdering mens COCOMO II følger forhåndsbestemte regler

Ved tidlig estimering ser UCP-metoden ut til å være enklest å bruke. FPA, COCOMO II og WEBMO ser ut til å kreve noe mer analyse av systemet før estimeringen kan gjennomføres.

UCP-metoden og COCOMO II ser ut til å anta god kvalitet ved estimering. WEBMO inntrykk av å overestimere, og være for omfattende for webapplikasjoner tilsvarende DES. Resultatene bekrefter også muligheten i å utvikle webapplikasjoner fort og billig på bekostning av fremtidig gjenbruk og kvalitet i utvikling og kode.

UCP-metoden kan benyttes av en kunde som skal velge leverandør blant ett sett tilbud. Det er jo ikke slik at den billigste og raskeste løsningen trenger å være den beste. Dermed kan UCP-metoden gi kunden en pekepinn ved valg av leverandør. Både fremgangsmåten og ingen krav om teknisk innsikt, øker brukervennligheten av UCP-metoden. De andre estimeringsmodellene vil være for omfattende for en kunde uten særskilt teknisk innsikt.

8.5 Videre arbeid

Da DES er et lite webprogram med enkel funksjonalitet gjelder resultatene, som allerede nevnt, kun i tilsvarende tilfeller. En videre studie av de fire

estimeringsmetodene i bruk for store webutviklingsprosjekter vil være nødvendig for å finne en mer generell trend for estimering i webutvikling. Videre vil en mer detaljert studie av FPA, WEBMO og UCP gi mer informasjon om forskjellene på metodene.

Boehm et al. (2000c:s.20) viser til en konverteringstabell fra UFP til kildekodelinjer. For fjerdegenerasjonsspråk som brukes i DES tilegnes 20 LOC per UFP. I følge Reifer (2002) bør objektorienterte webprogrammer bruke faktoren 25 kildekodelinjer per UFP. I tillegg er deler av DES programmert i Java. For rene Java program skal faktoren 53 kildekodelinjer per UFP brukes. Ved en prosentvis oppdeling av antall UFP i fjerdegenerasjonsspråk og antall UFP i Java kan estimatene i COCOMO II og WEBMO kanskje beregnes annerledes.

Generelt konkluderer studiet med at UCP-metoden antar et relativt høyt kvalitetsnivå. Det kan her jobbes videre med en vurdering av hvilket detaljnivå i use case som antar hvilken kodekvalitet. Slik kan utviklerne på forhånd planlegge kvalitet og estimere deretter.

COCOMO II består av flere interne metoder. Hvilke metoder som brukes avhenger av hvor langt utviklerne har kommet i livssyklusen. En mulighet for UCP-metoden kunne vært å dele den inn i tilsvarende deler. Detaljerte use case er ofte et tegn på at utviklere har kommet lenger i utviklingsprosessen. Imidlertid gir detaljerte use case høyere estimater. Dersom detaljeringsnivået i use case avgjør hvilken fremgangsmåte innad i UCP-metoden som benyttes, vil kanskje UCP-metoden kunne brukes i flere steg i utviklingen.

Dette studiet konkluderer med at det kan være mulig for kunder uten særlig teknisk innsikt, å benytte seg av UCP-metoden ved valg av leverandør. Noe mer forskning kan gjøres ved å gjøre forsøk der kunden selv estimerer ved hjelp av UCP-metoden. I tillegg foreslår jeg videre forskning på muligheten å dele UCP-metoden opp i en versjon for kunder og en noe mer teknisk versjon for leverandører.

9 ORDLISTE

COCOMO	COnstructive COst Modell
CRUD	Create, Read, Update, Delete – funksjoner
DBMS	Databasesystem
DES	The Simula Database of Empirical Studies,
EF	Omgivelsesfaktor
EI	Ekstern Inndata
Ekspert	Beskriver en person med noen års erfaring innenfor system-utvikling
EIF	Extern Interface File (Ekstern grensesnitt fil) i FPA
EM	Multiplikator i COCOMO II
ES	Ekstern Spørring
EU	Ekstern Utdata
FP / FPA	Funksjonspoeng / Funksjonspoenganalyse
ILF	Intern Logisk Fil
KLOC	Tusen kildekodelinjer (Kilo kildekodelinjer)
LOC	Kildekodelinjer
MVC	Modell View Controller, et mønster for web programmering
OS	Operativsystem
SF	Skaleringsfaktor i COCOMO II
Simula	Kortversjon for avdeling for Software Engineering ved Simula Research Laboratory
Simulaweb	Det eksisterende websystemet hos Simula Research Laboratory før utvikling av DES
SRL	Simula Research Laboratory
TCF	Teknisk kompleksitets faktor
UCP	Use case poeng
UCW	Use Case Weight
UFP	Ujusterte funksjonspoeng
UUCP	Ujusterte use case poeng
VAF	Value Adjustment Factor i funksjonspoenganalyse
WEBMO	WEB MOdel

BIBLIOGRAFI

[Adler og Adler 1994] P. A. Adler, P. Adler. "Observational Techniques". I: N. K. Denzin and Y. S. Lincoln. *Handbook of Qualitative Research*. Thousand Oaks California, Sage Publications Inc. 1994, s. 377 – 392.

[Adolph og Bramble 2003] S. Adolph, P. Bramble. *Patterns for effective use cases*. Boston, Pearson Education inc, 2003.

[Anda et al. 2001] B. Anda, H. Dreiem, D. I. K. Sjøberg, M. Jørgensen. "Estimating Software Development Effort based on Use Cases – Experiences from Industry". I: M. Gogolla, C. Kobryn (EDS.). *UML 2001 – 4th International Conference on the Unified Modeling Language*, Toronto, Canada, Oktober 1 – 5, s. 487 – 502, LNCS 2185, Springer-Verlag.

[Anda 2002] B. Anda. "Comparing Effort Estimates Based on Use Case Points with Expert Estimates". I: *EASE 2002 – Empirical Assessment in Software Engineering*, Keele, UK, April 8 – 10, 2002.

[Anda et al. 2002] B. Anda, E. Angelvik, K. Ribu. "Improving Estimation Practices by Applying Use Case Models". I: M. Oivo, S. Komi-Sirviö (Eds.). *PROFES 2002 – 4th International Conference on Product Focused Software Process Improvement*. Rovaniemi, Finland, December 9 – 11, 2002, s. 383 – 397, LNCS 2559, Springer-Verlag.

[Boehm 1981] B. W. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[Boehm et al. 1998] B. W. Boehm, S. Chulani, C. Bradford. *Calibrating the COCOMO II Post-Architecture Model*. Center for Software Engineering, Computer Science Department, University of Southern California, USA, 1998. Tilgang: http://sunset.usc.edu/Research_Group/Sunita/down/calpap.php [Sitert: 20.12.2004]

[Boehm et al. 2000a] B. W. Boehm, C. Abets, S. Chulani. *Software Development Cost Estimation Approaches - A Survey*. Technical Report USC-CSE-2000-505, University of Southern California, Center for Software Engineering, USA, 2000. Tilgang: http://sunset.usc.edu/publications/TECHRPTS/2000/usccse2000-505/usccse 2000-505.pdf [Sitert: 20.12.2004]

[Boehm et al. 2000b] B. W. Boehm, R. E. Fairley. *Software Estimation Perspectives*. IEEE Software. November/desember 2000. Tilgang: http://www.computer.org/software/so2000/pdf/s6022.pdf [Sitert: 20.12.2004]

[Boehm et al. 2000c] B. W. Boehm, C. Abets, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, B. Steece. *Software Cost Estimation with COCOMO II*. New Jersey, Prentice Hall PTR, 2000.

[Cockburn 2001] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2001.

[Creswell 2003] J. W. Creswell. *Research Design; Qualitative, Quantitative and Mixed Methods Approaches*. Second Edition, Californua USA, Sage Publications, Inc, 2003.

[Fetcke et al. 1998] T. Fetcke, A. Abran T.H. Nguyen. "Mapping the OO-Jacobson Approach into Function Point Analysis". I: R. Ege, M. Singh, and B. Meyer. *Technology of Object-Oriented Languages and Systems* Santa-Barbara, California, USA, Juli 28-Aug. 1, 1997, Eds., IEEE Computer Society, s. 192-202. Tilgang: http://user.cs.tu-berlin.de/~fetcke/ papers/Fetcke1997 .pdf [Sitert: 22.02.2005]

[Garmus og Herron 2004] D. Garmus, D. Herron. *Function Point Analysis – Measurement in Practices for Successful Software Projects*. Addison-Weseley. 3. trykking januar 2004, Copyright 2001.

[Gundersen 1996] D. Gudersen. *Norske synonymer blå ordbok*. Kunnskapsforlaget Ascheoug og Gyldendal, 1996.

[Hansen 2002] T. B. Hansen. *Måling og estimering*. Kurs leksjon, TISIP, 19. mars 2002.

[Hjertø 2002] G. Hjertø. *Innføring av et kvalitetssystem for systemutvikling*. Kurs leksjon, TISIP, 15. august 2002.

[Jørgensen og Carelius 2004] M. Jørgensen, G. J. Carelius. *An Empirical Study of Software Project Bidding*. IEEE Transactions on Software Engineering, Vol 30, No. Simula Research Laboratory, 2004. Tilgang: http://www.simula.no/photo/desbidding16.pdf [Sitert: 21.12.2004]

[Jørgensen og Sjøberg 2004] M. Jørgensen, D. I. K. Sjøberg. *An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy.* Journal of Information and Software Technology, 45 (3), Mars 2003, s. 123-136. Tilgang: http://www.simula.no/ people_publication.php?people_id=36&internal_people=y [Sitert: 18.03.2005]

[Jacobson et al. 1992] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard. *Object Oriented Software Engineering: A Use Case-driven Approach.Addison-Weseley, 1992.*

[Karner 1993] G. Karner. *Resource Estimation for Objectory Projects*. Objective Systems SF AB, 1993. Tilgang: http://www.bfpug.com.br/Artigos/UCP/Resource Estimation for Objectory Projects.pdf [Sitert: 21.12.2004]

[Krutchen 2003] P. Krutchen. *The Rational Unified Process, an Introduction*. Third Edition. Addison – Wesley, desember 2003.

[Kulak og Guiney 2004] D. Kulak, E. Guiney. *Use Cases – Requirements in context*. Pearson Education, Inc. 2004.

[Larman 2002] G. Larman. A*pplying UML And Patterns. An Introduction to Object-Oriented Analysis and Design and the Unified Process.* New Jersey, Prentice Hall PTR, 2002.

[Longstreet 2003] D. Longstreet. *Use Cases and Function Points*. Longstreet Consulting Inc. 2003. Tilgang: http://www.ifpug.com/Articles/usecases.htm [Sitert: 20.12.2004]

[Longstreet 2004a] D. Longstreet. *Fundamentals of Function Point Analysis*. Longstreet Consulting Inc. 2004. Tilgang: http://www.ifpug.com/fpafund.htm [Sitert: 20.12.2004]

[Longstreet 2004b] D. Longstreet. *Function Points Analysis Training Course*. Longstreet Consulting Inc. Oktober 2004. Tilgang: http://www.ifpug.com/Function%20Point%20Training%20Booklet%20New.pdf [Sitert: 20.12.2004]

[Mathiassen et al. 2000] L. Mathiassen, A. Munk-Madsen, P. A. Nielsen, J. Stage. *Object oriented analysis & design*. Marko Publishing ApS, Danmark, 2000.

[Merrick 2005b] P. J. Merrick. *Problems with Use Case Points Method.* Tilgang: http://www.uea.ac.uk/~a168955/effort_estimation/problems_UCPM.html [Sitert: 21.12.2004]

[Merrick 2005c] P. J. Merrick. *Simplification of the UCPM.* Tilgang: http://www.uea.ac.uk/~a168955/effort_estimation/simplification_UCPM.html [Sitert: 11.01.2005]

[OMG 2003] Object Management Group. *OMG Unified Modeling Language Specification.* Versjon 1.5, 01.03.2003. Tilgang: http://www.omg.org/gettingstarted/what_is_uml.htm [Sitert: 08.01.2005]

[Paul et al. 1995] M. C Paul, C. V. Weber, B. Curtis, M. B. Chrissis et al. *The Capability Maturity Model. Guidelines for Improving the Software process*. Carnegie Mellon University. Software Engineering Institute. SEI Series in software engineering. Addison- Wesley 1995.

[Quinn 1987] P. M. Quinn. *How to use qualitative methods in evaluation*. SAGE Publications 1987.

[Reifer 2002] D. J. Reifer. *Estimating Web Development Costs: There Are Differences*. Reifer Consultants, Inc. California, USA. Juni 2002. Tilgang: http://www.reifer.com/documents/webcosts.pdf -- 2003

[Reifer 2001] D. J. Reifer. *Web objects counting conventions*. Web Objects White Paper. Reifer Consultants, Inc, California, USA. 01.03.2001. Tilgang: http://www.reifer.com/documents/WOCounting.doc [Sitert: 20.12.2004]

[Reifer 2000] D. J. Reifer. *Web Development: Estimating Quick-to-Market Software*. IEEE Software. November/Desember 2000: 57-64. Tilgang: http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=895169 [Sitert: 20.12.2004]

[Ribu 2001] K. Ribu. *Estimating Object-Oriented Software Projects with Use Cases*. Master of Science Thesis, Universitetet i Oslo, 7. november 2001. Tilgang: http://heim.ifi.uio.no/~kribu/oppgave.pdf [Sitert: 21.12.2004]

[Roetzheim 2000] W. H. Roetzheim. *Estimating Internet Development*. Software Development Magazine, 28. august 2001. Tilgang: http://www.allfreetech.com/ShowMain.asp?MainID=28 [Sitert: 21.12.2004]

[Schneider og Winters 2001] G. Schneider, J. P. Winters. *Applying Use Cases - a practical guide.* Second editon, Addison-Wesley, mars 2001.

[Silverman 2001] D. Silverman. Interpreting Qualitative Data: Methods for Analyzing Talk, Text and Interaction. Second Edition, SAGE Publications, 2001.

[Symons 2001] C. Symons. *Come Back Function Point Analysis (Modernised) – All Is Forgiven!* Software Measurement Services Ltd, 2001. Tilgang: http://www.gifpa.co.uk/library/Papers/Symons/fesmafpa2001/paper.html [Sitert: 06.04.2005]

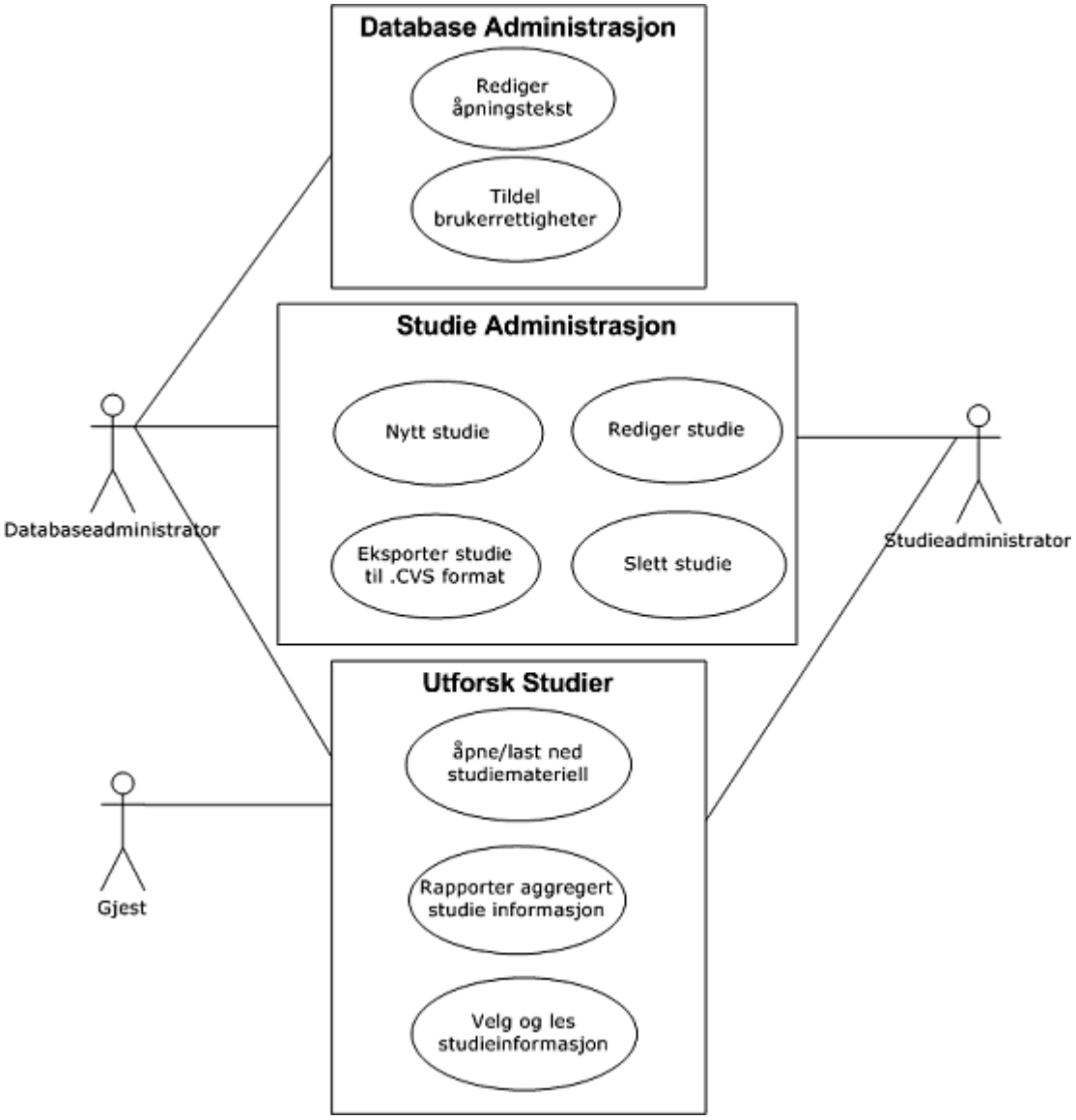
[Treble og Douglas 1995] S. Treble, N. Douglas. *Sizing and Estimating Software in Practice: Making MKII Function Points work*. McGraw-Hill Book Company Europe(UK), 1995.

[Valacich et al. 2001], G. Valacich, Hoffer. *Essentials of System Analysis & Design*. Prentice Hall, 2001, s. 39 - 57.

[Walsham 2002] G. Walsham. "Interpretive Case Studies in IS Research: Nature and Method". I: M. D. Mayers, D. Avison. *Qualitative Research in Information Systems*. London, Sage Publications, 2002, s. 103 – 113.

[Yin 1994] R. K. Yin. "Designing Case Studies". I: R. K. Yin. *Case Study Research: Design and Methods*. Second Edition. Applied Social Research Methods Series, Volume 5. NY, London, Sage Publications, Inc. 1994. s. 18 – 53.

Vedlegg A DES USE CASE DIAGRAM



Vedlegg B TABELLER I COCOMO

B.1 COCOMO 81

Konstanter ved bruk av Basic COCOMO 81 hentet fra Hansen (2002).

Modus	A	b	c	d
Organic	2,4	1,05	2,5	0,38
Moderate	3	1,12	2,5	0,35
Embedded	3,6	1,2	2,5	0,32

B.2 COCOMO II

B.2.1 Applikasjonssammensetning

Utgangspunktet for applikasjonssammensetning er objektpoeng tilpasset COCOMO II i form av applikasjonspoeng. Applikasjonssammensetning brukes typisk under grensesnittets prototyping tidlig i utviklingsprosessen. I den fasen er størrelse vanskelig å forutsi. Prosedyren ved bruk av applikasjonssammensetning er som følger [Hansen 202]:

1. Estimer antall skjermbilder, rapporter og programvaremoduler som må skrives i et tredjegenerasjonsspråk.
2. Klassifiser hvert element som enkelt, middels eller komplekst og multipliser med aktuell vekt.
3. Summer og få antall applikasjonspoeng (AP).
4. Estimer andel gjenbruk. Beregn nye applikasjonspoeng (NAP) med formelen:

```
NAP = AP * (100-%gjenbrukt)/100
```

5. Bestem produktivitetsrate (PROD) ut fra følgende skjema;

Utviklers erfaring	Meget Liten	Liten	Nominell	Stor	Meget Stor
Modenhet til applikasjons generator	Meget Liten	Liten	Nominell	Stor	Meget Stor
Produktivitet (PROD)	4	7	13	25	50

6. Beregn estimerte månedsverk (MV) der et månedsverk regnes som 152 timer, med følgende formel;

```
MV = NAP / PROD
```

B.2.2 Kostnadsdrivere tidlig design modell

Kostnadsdrivere fra tidlig design modell hentet fra Boehm et al. (2000c).

Driver		Ekstra Liten	Meget Liten	Liten	Nominell	Stor	Meget Stor	Ekstra Stor
RCPX Stabilitet og kompleksitet	EM_1	0,49	0,60	0,83	1,00	1,33	1,91	2,72
RUSE Gjenbruk	EM_2			0,95	1,00	1,07	1,15	1,24
PDIF Plattformvansker	EM_3			0,87	1,00	1,29	1,81	2,61
PERS Personellets kompetanse	EM_4	2,12	1,62	1,26	1,00	0,83	0,63	0,50
PREX Personellets erfaring	EM_5	1,59	1,33	1,12	1,00	0,87	0,74	0,62
FCIL Fasiliteter	EM_6	1,43	1,30	1,10	1,00	0,87	0,73	0,62
SCED Tidspress	EM_7		1,43	1,14	1,00	1,00	1,00	

B.2.3 Eksponentielle skaleringsfaktorer i COCOMO II

Eksponentielle skaleringsfaktorer i COCOMO II hentet fra Boehm et al. (2000c)

Driver		Ekstra Liten	Meget Liten	Liten	Nominell	Stor	Meget Stor	Ekstra Stor
PREC Forståelse	SF_1		6,20	4,96	3,72	2,48	1,24	0,00
FLEX Utviklingsfleksibilitet	SF_2		5,07	4,05	3,04	2,03	1,01	0,00
RESL Risikovurdering	SF_3		7,07	5,65	4,24	2,83	1,41	0,00
TEAM Teamkohesjon	SF_4		5,48	4,38	3,29	2,19	1,10	0,00
PMAT Prosessmodenhet	SF_5		7,80	6,24	4,68	3,12	1,56	0,00

B.2.4 Multiplikatorer i postarkitektonisk modell

Multiplikatorer i postarkitektonisk modell. Hentet fra Boehm et al. (2000c).

Multiplikatorer		Postarkitektonisk metode
RELY	Stabilitet	I hvilken grad systemet krever stabilitet. Dersom konsekvensen av en feil i programvaren er liten vurderes denne veldig lavt. Dersom utfallet av en feil i programvaren risikerer menneskelige liv vurderes denne veldig høyt.
DATA	Databasestørrelse	Databasens størrelse. Fanger den effekten store datakrav har på programutviklingen. Vurderes ved å beregne D/P, antall bytes i testdatabasen delt på antall kildekodelinjer. Lavere enn 10 gir lav, høyere enn 1000 gir veldig høy.
CPLX	Kompleksitet	Grad av kompleksitet. Her vurderes Kontrolloperasjoner, matematiske operasjoner, enhetsavhengige operasjoner, datastyring og brukergrensesnittets styringsoperasjoner.
RUSE	Gjenbruk	Grad av tilleggsinnsats for å lage komponenter til gjenbruk i fremtidige applikasjoner. Intet gjenbruk vurderes til lav, gjenbruk over flere applikasjonsgrenser vurderes til veldig høy.
DOCU	Dokumentasjonskrav	I hvilken grad det kreves til dokumentasjon. Dersom flere livssyklusbehov er udekket vurderes lav. Høy bruk av dokumentasjon i livssyklusmodeller vurderes til veldig høy.
TIME	Eksekveringstid	Krav til målbar eksekveringstid. Vurderes i prosent av mulig eksekveringstid. Mindre enn 50 % vurderes til normalt, 95 % vurderes til veldig høy.
STOR	Krav til minneplass	Grad av lagringsplass. Vurderes i prosent av mulig lagringsplass. Mindre enn 50 % vurderes til normalt, 95 % vurderes til veldig høy.
PVOL	Plattformvansker	Grad av ustabilheter og problemer ved plattform (OS, DBMS osv.) der programmet kaller for å utføre operasjoner. Store forandringer hver 12 måned, og mindre forandringer en gang per måned vurderes til lav. Store forandringer hver 2. uke, og mindre forandringer hver 2. dag vurderes til veldig høy.
ACAP	Analysekompetanse	Kapabiliteten generelt til analyse og design. Her vurderes utviklernes evne tilsamarbeid, kommunikasjon og effektivitet og nøyaktighet. Herunder vurderes ikke utviklernes erfaring. 15. prosentsats er dårlige analytikere og vurderes til veldig lav. 90. prosentsats er de beste analytikerne og vurderes til veldig høy.
PCAP	Programmerings-kompetanse	Kapabilitet, effektivitet og nøyaktighet samt evnen til kommunikasjon og samarbeid. Ser på

		programmererne som et team. 15. prosentsats er dårlige programmerere og vurderes til veldig lav. 90. prosentsats er de beste programmererne og vurderes til høy.
PCON	Personellets stabilitet	Grad av stabilitet i staben. 48 % per år gir veldig lav kontinuitet. 3 % per år gir veldig høy kontinuitet.
APEX	Applikasjonserfaring	Grad av erfaring med denne type applikasjon i teamet. Under 2 måneders gjennomsnittlig erfaring vurderes til veldig lav. 6 år eller høyere gjennomsnittlig erfaring vurderes til høy.
PLEX	Plattformerfaring	Grad av erfaring med denne type plattform. Under 2 måneders gjennomsnittlig erfaring vurderes til veldig lav. 6 år og mer erfaring vurderes til veldig høy.
LTEX	Språk- og verktøyerfaring	Grad av erfaring med programmeringsspråk og verktøybruk i teamet. Under 2 måneders gjennomsnittlig erfaring vurderes til veldig lav. 6 år og mer erfaring vurderes til veldig høy.
TOOL	Grad av verktøybruk	Grad av verktøy bruk. Enkle rediger og kode verktøy vurderes til lav mens mer avanserte støtteverktøy for livssyklusvurderes til veldig høy.
SITE	Flersteds utvikling	Her vurderes to punkter; utviklernes lokasjon og støtte for kommunikasjon. Internasjonalt prosjekt med noe kontakt gjennom e-post og telefon vurderes til lav. Samme bygning med kommunikasjon gjennom bredbånd, video og lignende vurderes til høy.
SCED	Tidspress	Grad av arbeid under tidspress i forhold til nominelt. 75 % av nominelt vurderes til veldig lav. 160 % av nominelt vurderes til veldig høy.

Vedlegg C WEBMO'S KOSTNADSDRIVERE

Kostnadsdrivere i WEBMO med vurderingskriterier hentet fra Reifer (2002).

Kostnadsdriver	Veldig lav	Lav	Nominell	Høy	Veldig høy
Stabilitet og kompleksitet (CPLX)	Kun klient, enkel matte og I/O, ingen distribusjon, stabiliteter ingen faktor.	Klient/tjener, noe matte, filstyring, begrenset distribusjon, enkel å gjen-opprette.	Klient/tjener, full distribusjon, integrasjon, moderate gjen-opprettelses krav.	Klient/tjener, vid distribusjon, mattematikk intensivitet, høye tap ved feil.	Klient/tjener, full distribusjon, samarbeidssystem, soft real-time, fare ved feil.
Verdi	0,63	0,85	1,00	1,30	1,67
Plattform-vanskeligheter (PDIF)	Sjeldne plattform forandringer, raskt nett, ingen resurs-begrensninger.	Få plattform forandringer, raskt nett, få ressurs-problemer	Stabil plattform, ok nett ytelse, må holde oversikt over ressursbruk.	Hyppige plattform-forandringer, tregt nett, mangel på ressurser.	Ustabil plattform, dårlig ytelse, begrensede ressurser.
Verdi	0,75	0,87	1,00	1,21	1,41
Personellets kapabilitet (PERS)	15 prosentsats, store for-sinkelser på grunn av utskiftning i arbeids-stokk	35 prosentsats, mindre forsinkelser på grunn av utskiftning i arbeids-stokk	55 prosentsats, få forsink-elser på grunn av utskiftning i arbeids-stokk	75 prosentsats, sjeldne forsinkelser på grunn av utskiftning i arbeids-stokk	90 prosentsats, ingen forsink-elser på grunn av utskiftning i arbeids-stokk
Verdi	1,55	1,35	1,00	0,75	0,58
Personellets erfaring (PREX)	≤ 2 måneder, begrenset verktøy, språk og plattform erfaring.	≤ 6 måneder, noe verktøy, språk og plattform erfaring.	≤ 1 år, gjennomsnittlig verktøy, språk og plattform erfaring.	≤ 3 år, mer enn gjennomsnittlig verktøy, språk og plattform erfaring.	≤ 6 år, stor verktøy, språk og plattform erfaring.
Verdi	1,35	1,19	1,00	0,87	0,71
Fasiliteter (FCIL)	Internasjonalt, intet samarbeid,	Multisite, noe samarbeid,	One complex, team,	Samme bygning, teamarbeid,	Co-lokalisert, integrerte

	språk-verktøy.	basis CASE, noen metoder.	livssyklus-metoder, gode verktøy.	integrert verktøy og metoder.	samarbeids metoder/verktøy etc.
Verdi	1,35	1,13	1,00	0,85	0,68
Tidsfaktorer (SCED)	Må korte ned, 75 % av nominell verdi.	Må korte ned, 85 % av nominell verdi.	Hold som det er, nominell verdi.	Kan slappe av litt, 120 % av nominell verdi.	Kan utvide, 140 % av nominell verdi.
Verdi	1,35	1,15	1,00	1,05	1,10
Gjenbruk (RUSE)	Ikke i bruk.	Ikke i bruk.	Uplanlagt gjenbruk.	Planlagt gjenbruk av komponent biblioteker.	Systematisk gjenbruk basert på arkitektur.
Verdi	--	--	1,00	0,75	0,62
Teamarbeid (TEAM)	Ingen delt visjon, ingen team kohesjon.	Litt delt visjon, marginalt effektive team og teamarbeid.	Noe delt visjon, fungerende team.	Betydelig delt visjon, sterk team-kohesjon.	Vidt delt visjon, eksepsjonell team-kohesjon.
Verdi	1,45	1,31	1,00	0,75	0,62
Prosess-effektivitet (PEFF)	Ad hoc, stol på helter.	Prosjekt-basert prosess, stol på ledelsen.	Rasjonalisert prosess, stol på prosessen.	Effektiv prosess, den beste måten å utføre jobb på.	Effektiv prosess, folk vil bruke den.
Verdi	1,35	1,20	1,00	0,85	0,65

Vedlegg D Language Expansion Factors

Language Expansion Factors hentet fra Reifer (2002).

Språk	LEF
1 generasjonsspråk standardverdi	320
C	128
2 generasjonsspråk standardverdi	107
COBOL (ANSI85)	91
FORTRAN 107	107
PASCAL	91
3 generasjonsspråk standardverdi	80
C++	53
Java for WEB	32
LISP	64
ORACLE	38
Visual Basic	40
Visual C++	34
Web standard – visuelle språk	35
Objektorientering standardverdi	29
EIFFEL	20
PERL	22
Smaltalk	20
Web – standard objektorienterte språk	25
4 generasjonsspråk standardverdi	20
Crystal Reports	20
Programgenerator standardverdi	16
HTML	15
SQL for Web	10
Regneark standardverdi	6
Excel	6
Screen Painter	6
5 generasjonsspråk standardverdi	5
XML	6
MATHCAD	5

Vedlegg E RESERVASJONSSYSTEM FOR ROM

Use Case navn:	UC1: Hent Rom
Iterasjon:	1
Sammendrag:	Henter oversikt over et spesifikt rom
Prebetingelse:	Innlogget som administrator
Postbetingelse:	Hente informasjon om et rom
Aktør:	Administrator
Hovedflyt:	1. Aktør ønsker å hente informasjon om et rom 2. Inkluder UC4: Hent Romoversikt 3. System viser oversikt over alle rom 4. Aktør velger rom 5. System viser data om valgt rom 6. System klart for nye kommandoer
Alternativ flyt:	A-1: *. Aktør kan når som helt avslutte A-2: 1a. 1. Aktør søker opp rom ved å taste romnummer 2. System henter rom 3. Fortsett fra hovedflyt punkt 5 A-3: A-2: 1. Aktør taster ugyldig rom 2. System fortsetter hovedflyt punkt 1
Utvidet flyt:	U-1: 3a. 1. UC3: Legg til Rom 2. Fortsett punkt 6 i hovedflyt U-2: 5a. 1. UC2: Slett Rom 2. Fortsett punkt 6 i hovedflyt
Forfatter:	
Dato:	

Use Case navn:	UC2: Slett Rom
Iterasjon:	1
Sammendrag:	Aktør ønsker å slette et registrert rom
Prebetingelse:	Innlogget som administrator
Postbetingelse:	Rom slettet
Aktør:	Administrator
Hovedflyt:	1. Aktør velger slett rom 2. System ber om bekreftelse på slett rom 3. Aktør bekrefter sletting av rom 4. System sletter rom
Alternativ flyt:	A-1: *. Aktør kan når som helt avslutte A-2: 4a. 1. Aktør avbryter sletting av rom 2. System returner til hovedflyt punkt 1 3. System klart for nye kommandoer

Forfatter:	
Dato:	

Use Case navn:	UC3: Legg til Rom
Iterasjon:	1
Sammendrag:	Aktør ønsker å legge til nytt rom
Prebetingelse:	Innlogget som administrator
Postbetingelse:	Nytt rom registrert
Aktør:	Administrator
Hovedflyt:	1. System viser alle registrerte rom 2. Aktør velger registrer nytt rom 3. System viser skjema for registrering av rom 4. Aktør fyller skjema og bekrefter 5. System validerer skjema 5. System legger til nyt rom 6. System klart for nye kommandoer
Alternativ flyt:	A-1: *. Aktør kan når som helt avslutte A-2: 5a. 1. Feil verdi i skjema 2. System gir beskjed om dette 3. Fortsett fra punkt 4
Forfatter:	
Dato:	

Use Case navn:	UC4: Hent Romoversikt
Iterasjon:	1
Sammendrag:	Returnerer alle rom
Prebetingelse:	UC1 eller UC5 er startet
Postbetingelse:	Oversikt over rom sendt til UC1 eller UC5
Aktør:	Administrator, Lærer
Hovedflyt:	1. Aktør ønsker oversikt over alle rom 2. System returnerer oversikt over alle rom
Alternativ flyt:	A-1: *. Aktør kan når som helt avslutte
Forfatter:	
Dato:	

Use Case navn:	UC5: Reserver Rom
Iterasjon:	1
Sammendrag:	Aktør ønsker å reservere et rom for bruk
Prebetingelse:	Innlogget som aktør
Postbetingelse:	Rom registrert
Aktør:	Administrator, Lærer
Hovedflyt:	1. Aktør velger reserver rom 2. Inkluder UC4: Hent Romoversikt 3. System viser liste over alle rom 4. Aktør velger rom 5. System viser informasjon om rom 6. Aktør taster ønsket reservasjonstid 7. System validerer inndata 8. System reserverer rom for aktør 9. System klart for nye kommandoer
Alternativ flyt:	A-1:

	*. Aktør kan når som helt avslutte A-2: 7a. 1. Aktør har tastet ugyldig tidsverdi 2. System returner beskjed om feil 3. Fortsett hovedflyt punkt 6
Forfatter:	
Dato:	

Use Case navn:	UC6: Slett Reservasjon
Iterasjon:	1
Sammendrag:	Aktør ønsker å slette reservasjon
Prebetingelse:	Innlogget som aktør
Postbetingelse:	Reservasjon slettet
Aktør:	Administrator, Lærer
Hovedflyt:	1. Aktør velger slett reservasjon 2. System viser alle aktørs reservasjoner 3. Aktør velger slett ønsket reservasjon 4. Systemet ber om bekreftelse på sletting 5. Aktør bekrefter 6. System sletter reservasjon 7. System klart for nye kommandoer
Alternativ flyt:	A-1: *. Aktør kan når som helt avslutte A-2: 7a. 1. Aktør avbryter sletting 2. Fortsett hovedflyt punkt 7
Forfatter:	
Dato:	