Sammendrag

Rational Unified Process (RUP) har blitt en populær utviklingsmetodikk, og mange IT-system utvikles i henhold til RUP sin tilnærming. Motivasjonen for å benytte seg av RUP kan være et ønske om stabil og forutsigbar prosjektgjennomføring. Samtidig er det en økende interesse for brukersentrert systemutvikling. Det stilles stadig høyere krav til brukevennlighet i moderne IT-system.

Målet for denne oppgaven er å se på forholdet mellom brukersentrert systemutvikling og Rational Unified Process (RUP). Hvordan kan vi integrere brukersentrerte metoder i RUP?

For å se nærmere på dette har jeg gjennomført en casestudie hvor jeg ser nærmere på den faktiske anvendelsen av RUP og brukersentrerte metoder. Min casestudie viser at brukersentrert systemutvikling kan være vanskelig å gjennomføre på grunn av organisatoriske hindringer. Forskjellige elementer som kontrakt, forholdet mellom ledelse og bruker, økonomi, tid tilgjengelig med mer, legger føringer for hvilke metoder man kan bruke og hvilken form en systemutviklingsprosess kan ha. Dette er aspekter man må ta høyde for hvis man ønsker å integrere brukersentrerte metoder i RUP.

I denne oppgaven ser jeg nærmere på hvordan organisatoriske aspekter påvirker utviklingsprosessen. I tillegg foreslår jeg nye roller, aktiviteter og artefakter egnet for å gjøre RUP mer brukersentrert.

Forord

Hovedoppgaven har blitt gjennomført ved Institutt for datateknikk og informasjonsvitenskap, Gruppe for Systemarbeid og menneske-maskin-interaksjon. Oppgaven er initiert av førsteamanuensis Dag Svanæs, som også har vært min veileder gjennom hele arbeidet.

En viktig del av oppgaven er en case-studie hvor jeg fulgte et systemutviklingsprosjekt i et konsulentselskap. Jeg vil gjerne takke bedriften for at jeg fikk lov til å følge deres arbeid og at de ansatte tok seg tid til å svare på mine spørsmål. Jeg vil også takke Kjetil Nyseth, Ingvald Skaug, Inge Solvoll, Håvard Wigtil og Linda Grønstad for korrekturlesing, faglige innspill og inspirasjon.

Jeg vil takke min veileder Dag Svanæs for livlige diskusjoner, motiverende hjelp og støtte fra begynnelse til slutt.

Figurliste

Tabelliste

1 Introduksjon

1.1 Rational Unified Process og brukervennlighet

The Rational Unified Process (RUP) er en systemutviklingsprosess som er utviklet og markedsført av Rational Software[1]. Hensikten med RUP er å gi bedrifter og utviklingsprosjekter en oppskrift på hvordan de skal drive systemutvikling. En påstand fra RUPs utviklere er at vi ikke bare skal stole på flinke ansatte. Dette hevdes av Ivar Jacobson, Grady Booch og James Rumbaugh i boken The Unified Software Development Process [Jacobsen et al. 1999]. De hevder at for å produsere bra programvare om og om igjen, må man ha en solid og reproduserbar prosess å forholde seg til. Man kan ikke alene stole på talentet til enkeltansatte. En slik tilnærming er for ustabil og har en høy risiko. Mangel på en utviklingsprosess vil føre til ad hoc utvikling, noe som igjen fører til at man er avhengig av heroisk innsats fra enkeltansatte. RUP tar derfor sikte på å gi organisasjoner som driver systemutvikling en prosess som kan hjelpe dem å jobbe på en strukturert og forutsigbar måte.

Programvare blir i stadig større grad en viktig del av vårt hverdagsliv. Stadig flere arbeidstakere må forholde seg til stadig flere IT-system på arbeidsplassen. Dette fører ofte til store endringer på disse arbeidsplassene. Også hjemme benytter vi oss av flere IT-system. Siden programvare tar en så stor del av vårt hverdagsliv er det viktig at disse IT systemene er brukervennlige, både av etiske og økonomiske grunner. IT-system som ikke er brukervennlige kan blant annet føre til økonomiske tap eller i verste fall skader eller dødsfall. Artikkelen ”Challenges of HCI Design and Implementation” oppsummerer en rekke studier som viser fordelene ved å satse på brukervennlighet[Myers 1994].

RUP har blitt en populær utviklingsmetodikk, og mange IT-system utvikles i henhold til RUP. Dette gir Rational Software et stort ansvar. Enkelte stiller spørsmålstegn ved RUPs evne i forhold til utvikling av brukervennlige IT-system [Cronin 2003, Göransson 2003]. Det finnes en rekke alternative metoder til RUP hvor hovedfokus er brukervennlighet. I disse metodene er det hovedfokus på menneskelige faktorer, og man henter inspirasjon fra fagfelt som kognitiv psykologi og antropologi. Disse metodene omtales ofte som brukersentrerte metoder. Har RUP noe å lære fra disse metodene? I denne hovedfagsoppgaven skal jeg se nærmere på følgende:

> **Hvis vi tar som utgangspunkt at brukersentrerte teknikker fører til økt brukervennlighet: I hvor stor grad er RUP brukersentrert, og hvilke grep kan vi eventuelt gjøre for å få en mer brukersentrert utviklingsprosess?**

[1] IBM eier Rational Software, men drives som en separat avdeling. Jeg vil derfor kun omtale Rational Software.

1.2 Om Rational Unified Process

Philippe Kruchten var i perioden 1996-2003 "Director of Process Development" i Rational Software. Ifølge Phillippe Kruchten har en systemutviklingsprosess fire roller [Kruchten 2000]:

1. Gi veiledning i forhold til hvilke rekkefølge aktiviteter skal ha i et prosjekt.
2. Spesifisere hvilke artefakter som skal lages og når de skal lages.
3. Instruere oppgavene til individuelle utviklere og til hele gruppen.
4. Tilby kriteria for å overvåke og måle prosjektets produkter og aktiviteter

RUP baserer seg på "*best practices*" [Kruchten 2000]. Best practices er metoder og tilnærminger som er brukt av vellykkede utviklingsorganisasjoner. Dette er metoder som har vist sin verdi over tid og som er anerkjente. Disse best practices er ifølge Kruchten resultatet av kunnskap hentet fra flere tusen vellykkede prosjekt. Kruchten oppgir seks best practices:

Programvare skal utvikles iterativt
I tradisjonell fossefallutvikling er kravinnsamling og design gjennomført i begynnelsen av prosjektet. Kruchten hevder at dette plasserer risiko fram i tid. Det er umulig å vite alt på første forsøk og forutsetningen endres underveis. Og når man driver fossefallsutvikling vil nye krav underveis føre til forsinkelser i prosjektet. Det er derfor viktig å bryte prosjektet ned i en serie med iterasjoner. Ved hjelp av disse iterasjonene kan man kontinuerlig verifisere produktet opp mot potensielle brukere og andre som her en interesse i systemet. Dette kan bidra til å identifisere risiki tidlig.

Fokus på kravhåndtering
Kravene til et IT-system er stadig i endring, og det er ifølge Kruchten umulig å definere alle krav i begynnelsen av et prosjekt. Etter hvert som brukere får en større forståelse for systemet, så endres kravene til systemet. Det er derfor viktig for systemutviklerne å holde fokus. De må dokumentere ønsker, finne de viktige kravene, og ta vanskelige avgjørelser.

Programvare skal modelleres visuelt
En modell er en forenkling av virkeligheten som beskriver et system fra en bestemt synsvinkel. Visuelle modeller gjør det lettere for systemutviklere å forstå ulike aspekter av et system. Dette kan for eksempel være arbeidsoppgavene til sluttbrukeren eller detaljer knyttet til systemets kildekode. Modeller gjør det lettere for systemutviklerne å kommunisere med hverandre og andre knyttet til prosjektet. RUP bruker Unified Modeling Language (UML) som standard modelleringsspråk.

Kvaliteten på programmet skal verifiseres kontinuerlig
Jo lengre tid det går før man fikser et problem, jo dyrere er det. Det er ifølge Kruchten mellom 100 og 1000 ganger dyrere å reparere et problem etter at et system er distribuert. Derfor er det viktig å verifiser kvaliteten på et system ofte. Det er viktig å verifisere funksjonaliteten, påliteligheten og ytelsen til systemet.

Kontroll over programvareendringer
I et utviklingsprosjekt er det ofte mange utviklere. I tillegg er de kanskje spredd over flere geografiske lokasjoner, mens de jobber med forskjellige produkter, i forskjellige utgivelser og mot flere plattformer. Da er det viktig å koordinering av disse aktivitetene. Med slik kompleksitet er det ifølge Kruchten viktig å holde oversikt over endringer som blir gjort.

Bruk komponentbasert arkitektur
Å ha en komponentbasert arkitektur åpner blant annet for gjenbruk av komponenter og høyere bruk av standardkomponenter.

RUP er ikke bare en guide for systemutviklere. Rational Software ønsker at RUP skal bli en standard for hvordan man skal drive systemutviklingsprosjekter. Noe som åpner for en felles prosess for alle som er involvert i et systemutviklingsprosjekt. Rational ønsker at RUP skal være et bindeledd mellom systemutviklere, kundene, brukere og ledelsen. RUP er blitt en populær utviklingsmetodikk som benyttes av mange systemutviklingsmiljø og har derfor kommet et stykke på vei for å nå denne målsetningen.

RUP er ikke bare en utviklingsmetodikk, det er også et kommersielt produkt. Rational er kommersielt firma som ønsker å tjene penger. Dette gjør Rational ved å selge forskjellige produkter og tjenester knyttet til RUP. Blant annet selger de applikasjoner som kan hjelpe team med å styreprosjekter eller tegne UML-diagrammer. De selger også konsulenttjenester knyttet til RUP. De kan bidra med å integrere metoden i en bedrift eller holde kurs for ansatte.

Jeg vil i denne oppgaven bruke de originale engelske begrepene når jeg omtaler RUP-begreper. Dette vil gjøre det lettere for leseren å se min oppgave i forhold til andre tekster om RUP.

1.3 Brukervennlighet

1.3.1 Hva er brukervennlighet?

Jeg velger i denne oppgaven å oversette det engelske ordet usability med brukervennlighet. Brukervennlighet er det mest brukte oversettelsen for dette ordet. Et annet alternativ kunne vært brukskvalitet. Ordet brukervennlig brukes i mange sammenhenger og blir ofte tillagt forskjellige meninger. Mange tilegner ordet betydningen "lett å lære", men brukervennlig kan være mye mer. En definisjon av brukervennlighet er gitt i ISO 9241.

> *"The effectiveness, efficiency, and satisfaction with which specified users achieve specified goals in particular environments." [ISO 9241-11:1998, def 3.1]*

En annen og mer omfattende definisjon er gitt av Jakob Nielsen [Nielsen 2003] som foreslår fem kategorier innen brukervennlighet. Disse er:

Lett å lære
Det første erfaringen en bruker har med et system er å lære systemet. Det er viktig at systemet er lett å lære. Det er også viktig at man kan begynne å bruke systemet uten å ha lært alt. Det fleste vil ikke lære hele systemet før de begynner å bruke det, så programmet bør ta hensyn til dette og advare brukere som er i ferd med å gjøre alvorlige feil. Systemet bør kunne la brukere få korrigere mindre feil.

Effektiv i bruk
Effektivitet i denne konteksten er knyttet til prestasjonene til en erfaren bruker. En erfaren bruker er en som har lært alt som han eller hun treger å lære om systemet.

Lett å huske
Det er ikke bare viktig at systemet er lett å lære, det må også være lett å huske.
Det finnes mange system som kun benyttes ved bestemte anledninger. På samme måte som det er viktig for brukeren å fort lære et nytt system, så er det viktig å huske et system som han eller hun bruker av og til. Eksempler på slike program kan være programmer for å generere kvartalsrapporter eller et skatteprogram.

Få feil
Brukeren bør begå så få feil som overhodet mulig. En feil er en handling som ikke fører fram mot det opprinnelige målet. Noen feil blir korrigert umiddelbart av brukeren. Den enste konsekvensen med slike feil er lavere effektivitet. Andre feil er langt mer destruktive og kan føre drastisk redusert produktivitet eller tap av arbeid. Man bør gjøre alt man kan for å unngå slike feil.

Subjektiv fornøydhet
Hvor behagelig det for den enkelte bruker å bruke systemet.

Det finnes mange andre kvalitetsfaktorer. Jakob Nielsen påpeker at blant annet nytteverdi kommer i tillegg til hans definisjon. Han sier i tillegg at mange av de teknikkene som benyttes for å bedre brukervennlighet til et system også egner seg til å kartlegge nytteverdien til et system.

Nielsens siste kategori er subjektiv fornøydhet. I hvilken grad kan vi måle dette? Software Usability Measurement (SUMI) [Kirakowski 1994] er et forsøk på å måle den subjektive oppfattelsen av et system kvantitativt. På en side kan vi måle hvor effektivt systemet er i bruk, og hvor mange feil en bruker gjør. Hensikten til SUMI er derimot å måle hvordan systemet oppfattes av brukeren. Dette gjøres ved hjelp av spørreskjema. Resultatet av undersøkelsen brytes ned i fem kategorier:

> *"Affect, Efficiency, Helpfulness, Control, and Learnability."*

SUMI er et eksempel på hvordan man kan måle brukervennlighet. I tillegg til metoder for å måle brukervennlighet, så har man også måter å rapportere hvor brukervennlig et system er. The Common Industry Standard[Theofanos 2005] er en ISO (ISO/IEC 25062) standard for hvordan et systems brukervennlighet skal rapporteres. Rapporten brukes primært for å rapportere brukervennlighetstester med kvalitative måleenheter. Rapporten følger vanlig vitenskaplig form, og har følgende kapitler: oppsummering, introduksjon, metode og resultat. Rapporten har to målgrupper: De som er ansvarlig for brukervennlighet, og interessenter til IT-systemet. Interessentene kan for eksempel bruke rapporten til å ta strategiske avgjørelser.

1.3.2 Hvorfor er det viktig?

Å utvikle brukervennlige IT-system er ikke problemfritt. Det krever blant annet tid, penger og kompetanse. Hvis brukervennlighet koster mye, er det da kostnytteeffektivt å bruke ressurser på brukervennlighet? Jeg mener at det er viktig å fokusere på brukervennlighet på grunn av økonomiske og etiske grunner.

Flere studier viser muligheter for store økonomiske fordeler. En oversikt over slike studier kan man finne i [Myers 94].

- En studie viste at man ved å øke brukervennligheten på en mindre applikasjon brukt av 23 000 markedsføringspersonell sparte $41 700. Samme studie viser også hvordan man sparte $6 800 000 på en større applikasjon som var brukt av 240 000 ansatte. Disse resultatene ble oppnådd på grunn av raskere utførelse av arbeid, mindre feil, mindre avbud for brukere, mindre behov for brukerstøtte, mindre behov for opplæring og reduksjon i antall endringer etter lansering.

- En annen studie viser hvordan NYNEX sparte $2 millioner per år ved å ikke kjøpe et nytt system for sine telefonoperatører. Ved å gjøre en brukervennlighetsevaluering konkluderte de med at det nye systemet var mindre effektivt i bruk.

- En matematisk modell basert på elleve studier viser at man kan spare $39 000 i et mindre prosjekt, $613 000 i et mellomstort prosjekt og $8 200 000 i et større prosjekt.

- En annen studie fant at fordelene ved å fokusere på brukervennlighet kan gi innsparinger på opptil 5000 ganger de opprinnelige kostnadene.

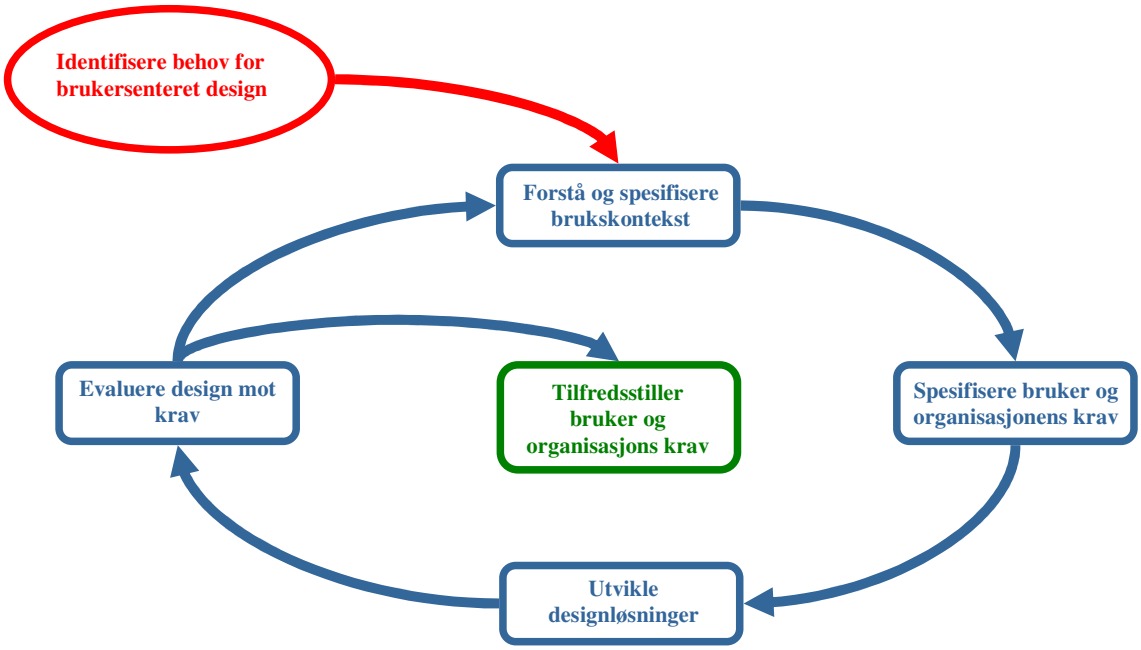
1.4 Brukersentrert systemutvikling

Hvordan lager man så brukervennlige IT-system? Her finnes det en rekke ulike tilnærminger, men det er ingen enighet blant praktikanter og forskere om hvordan man skal gjennomføre brukersentrert systemutvikling. Noe av grunnen til dette er at hvilken tilnærming som egner seg best er forskjellig fra gang til gang. En annen grunn er at brukersentrert systemutvikling er en relativt ung faggren. Brukersentrert systemutvikling har derfor ikke fått satt seg slik andre etablerte faggrener har fått muligheten til.

Det er derimot noen faktorer som går igjen i ulike brukersentrerte tilnærminger.

- Fokus på menneskelige faktorer
- Fokus på å forstå systemets brukskontekst
- Iterativ utvikling og brukertesting
- Bruker ofte verktøy og metoder som er inspirert av faggrener som psykologi, antropologi og grafisk design.

Disse trekkene finner vi også i ISO 13407, som er en internasjonal standard for hvordan man skal gå fram for å utvikle brukervennlige IT-system. ISO 13407 tar sikte på å være en guide for ledere og systemutviklere. Denne guiden er prosessbasert og forteller hvilke aktiviteter som bør være del i et utviklingsprosjekt som har behov for brukersentrert design. I figur 1.1 kan du se den overordnede prosessen til ISO 13407. Jeg vil komme tilbake til ISO 13407 i kapittel 2.2.2.



Figur 1.1 Prosessdiagram ISO 13407

Brukersentrert systemutvikling er ikke nok alene. Systemets design må stå i forhold til andre viktige faktorer. Et forsøk på å beskrive dette er laget av Donald Norman [s.39-41, Norman 1998]. Han sammenligner et IT-system med en trebeint krakk. For å stå er kraken avhengig av tre ben. Disse er forretninger, teknologi og design.

Forretninger
Skal man selge en løsning, så trenger denne å ha et marked. Videre, så må man ha kunder som er villige til å betale en pris som står i still med utviklingskostnadene. Man bør også tenke strategisk i forhold til sine konkurrenter.

Teknologi
Uten et system som lar seg implementere er man like langt. All design bør være praktisk gjennomførbart innen rimelig tid og med eksisterende ressurser.

Design
Et IT-system må være brukervennlig og ha nytteverdi for sine brukere.

1.5 Min tilnærming

Hvis vi tar som utgangspunkt at brukersentrerte teknikker fører til økt brukervennlighet: I hvor stor grad er RUP brukersentrert, og hvilke grep kan vi eventuelt gjøre for å få en mer brukersentrert utviklingsprosess? På bakgrunn av denne problemstillingen stiller jeg følgende forskningsspørsmål:

- Teori: Hvordan er RUPs tilnærming til systemutvikling sammenlignet med brukersentrerte metoder?

- Empiri: Hvilke faktorer påvirker den faktiske anvendelsen av brukersentrert metode i RUP-prosjekter?

- Anbefalinger: Hvordan kan brukersentrerte metoder integreres i RUP?

Jeg har valgt to tilnærminger for å svare på disse spørsmålene: teoretisk og empirisk.

Teori
I kapittel 2 kan du lese om RUP og brukersentrert design. Hva er filosofien bak, og hvordan er metodene bygd opp? I kapittel 4 vil jeg se nærmere på RUP og brukersentrert design fra et teoretisk perspektiv. Jeg vil sammenligne RUP og brukersentrert design slik de er beskrevet av de som har utviklet disse metodene. Hva er forskjellene? Er de komplementære eller i konflikt? Baserer de seg på ulik filosofi? Jeg vil også se nærmere på noe av kritikken som er kommet mot RUP i forhold til brukervennlighet.

Empiri
I kapittel 5 vil jeg presentere en case-studie jeg har gjort i forbindelse med oppgaven. I case-studie hadde jeg fokus på faktisk bruk av RUP og brukersentrert design. Jeg vil her se nærmere på kravhåndtering, brukermedvirkning og grensesnittdesign i et RUP-prosjekt som hadde som mål å drive brukersentrert systemutvikling. I kapittel 6 vil jeg analysere disse dataene. Jeg vil gi en vurdering på hvor moden bruken av RUP var, og hvor moden bruken av brukersentrerte metoder var. Jeg har i denne oppgaven gjennomført en casestudie av et bestemt prosjekt i en bestemt bedrift. I et slikt prosjekt kan det være mange faktorer som påvirker det endelige resultatet. Jeg kan derfor ikke generalisere mine funn.

Anbefalinger
På bakgrunn av dette vil jeg i kapittel 7 komme med en rekke forslag til forbedringer.

I kapittel 3 vil jeg se nærmere på mitt forskningsdesign.

2 Brukersentrert systemutvikling og RUP

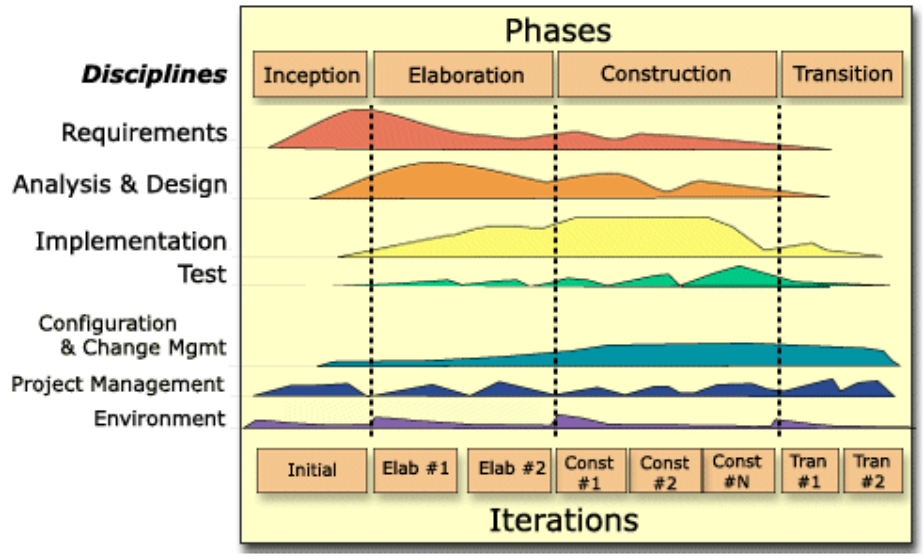
Jeg vil i dette kapittelet se nærmere på utformingen til RUP. Jeg vil i tillegg se nærmere på brukersentrert design. Jeg vil fokusere på ISO 13407, som er en internasjonal standard for hvordan man skal gå fram for å utvikle brukervennlige IT-system.

2.1 RUP

RUP er delt opp i ni disipliner. Jeg vil i det følgende gi en oversikt over disse disiplinene og hvordan de henger sammen. Jeg vil presentere alle disipliner, men jeg vil se ekstra nærme på kravhåndtering og forretningsmodellering siden disse er de mest relevant for denne oppgaven. Jeg vil ikke gi noen detaljert beskrivelse av disse disiplinene, men jeg vil prøve å få fram filosofien bak og litt om hvordan de utføres. Beskrivelsene av disse disiplinene er basert på [Jacobsen et al. 1999, Kruchten 2000, Kruchten et al. 2001, Leffingell & Widrig 2000]

2.1.1 Prosess og disipliner

Før jeg går nærmere innpå de enkelte disiplinene vil jeg si litt om hvordan de henger sammen. En disiplin i RUP er et sett med arbeidsprosesser. Hver arbeidsprosess har fire grunnelementer. Disse er roller, aktiviteter, artefakter og arbeidsflyt. Figur 2.1 viser RUPs to dimensjoner. Disse dimensjonene er den horisontale aksen som representerer tid og den vertikale som representerer innhold. I den vertikale aksen ser vi alle disiplinene som utgjør RUP. I den horisontale aksen ser vi hvor mye tid som skal brukes på de forskjellige disiplinene og hvordan dette henger sammen med iterasjonene i RUP. Denne delen er delt opp i fire faser. Disse er Inception, Elaboration, Construction og Transition. I denne figuren ser vi at hoveddelen av arbeidet i forbindelse med kravdisiplinen, skjer i Inception og Elaboration fasen.



Figur 2.1 Oversikt over faser og disipliner i RUP

Hver disiplin består av et sett med prosesser, og hver prosess har fire grunnelementer. Disse er roller, aktiviteter, artefakter og arbeidsflyt.

Roller
En medarbeider i et RUP prosjekt kan ha en eller flere roller. Dette kan være roller som System-Analyst, Use-Case Specifier og User-Interface-Designer.

Aktiviteter
Til de ulike rollene er det knyttet forskjellige aktiviteter. Disse aktivistene kan vare alt fra noen timer til flere dager. Flere av aktivitetene gjentas flere ganger i løpet av prosjektet.

Arbeidsflyt
For hver aktivitet er det beskrevet en arbeidsflyt. Denne arbeidsflyten skal gi prosjektdeltagerne en detaljert guide på hvordan de skal utføre en aktivitet.

Artefakter
Resultatet av de ulike aktivitetene er ofte en eller flere artefakter. Artefakter er alt håndfast som blir produsert av de ulike prosjektdeltagerne. I RUP har man en rekke standarder for hvordan slike artefakter skal være. Eksempler på en artefakt kan være et Use-Case modeller, prosjektets visjonsdokument eller en prototyp.

2.1.2 Forretningsmodellering

Hvorfor
IT-system er ikke lengre små enkle generiske verktøy som brukes til å løse enkeltoppgaver. Mange IT-system er sentrale brikker i bedrifters daglige drift. Eksempler på slike system er kundebehandlingssystem, logistikksystem og økonomisystem.

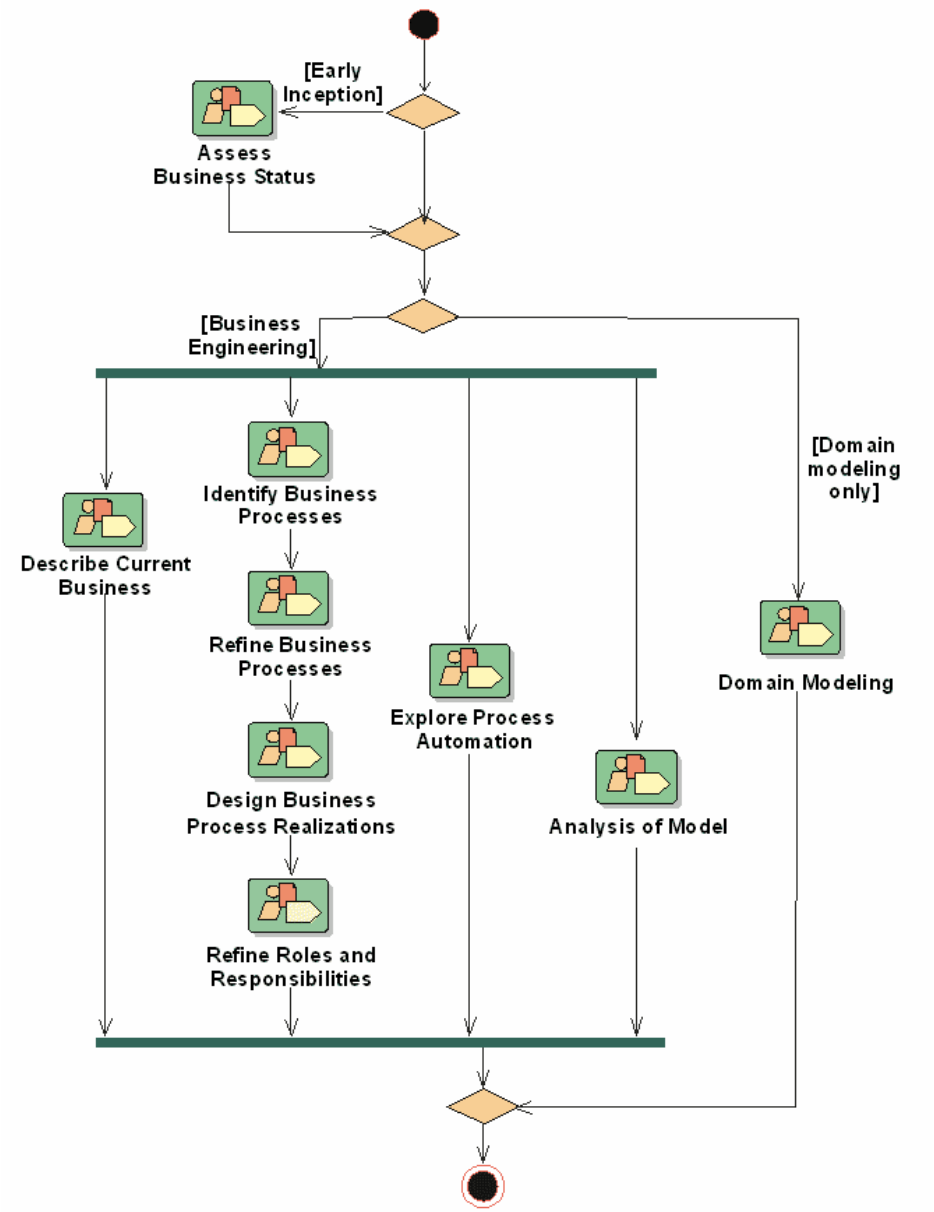
Det er ofte store utfordringer knyttet til å introdusere eller endre et IT-system i en organisasjon. Det er derfor viktig i systemutvikling at vi ser hvordan forretningen funger og hvordan et eventuelt IT-system henger sammen med dette. Derfor har man i RUP en egen disiplin for dette. Philippe Kruchten [2000] hevder at hensikten med forretningsmodellering er:

- Å forstå strukturen og dynamikken i en organisasjon hvor et IT-system skal innføres.
- Å forstå problemer i organisasjonen og identifisere mulige forbedringer.
- Å sørge for at kunder, sluttbrukere og utviklere har en felles forståelse av organisasjonen.
- Å finne systemkravene til organisasjonen.

[Leffingell & Widrig 2000] hevder at denne kunnskapen gjør oss i stand til å svare på en del viktige spørsmål:

- Hvorfor bygge et system?
- Hvor skal dette systemet plasseres?
- Hva slags funksjonalitet skal systemet ha?
- Når skal vi bruke manuell prosessering?
- Bør vi restrukturere organisasjonen for å løse dette problemet?

Det er ikke bestandig forretningsmodellering er nødvendig, og nøyaktig hvordan forretningsmodellering utføres er veldig avhengig av kontekst og behov.



Figur 2.2 Arbeidsflytdiagram for forretningsmodellering

Hvordan

RUPs arbeidsflytdiagram for forretningsmodellering gir et innblikk i hvilke steg man må igjennom(se figur 2.2). Men nøyaktig hvordan man skal gjennomføre forretningsmodellering er som nevnt avhengig av kontekst og behov. Kruchten skisserer seks mulige scenarioer.

Organisasjonskart

Forretningsmodellering trenger ikke å være en omfattende aktivitet. Kruchten hevder at noen ganger trenger man kun noen enkle diagram som beskriver organisasjonen, og de viktigste prosessene fra forretningsmodelleringsdisiplinen. I dette tilfellet er

forretningsmodellering en mindre aktivitet som hovedsakelig foregår i Inception-fasen.

Domenemodellering
Hvis hovedhensikten med systemet er å behandle informasjon (for eksempel et ordrebehandlingssystem), så bør man modellere denne informasjonen. Domenemodellering er en svært vanlig i RUP-prosjekt, og gjennomføres ofte i Inception- og Elaboration-fasene.

Flere system til en organisasjon
Det kan hende at man kan dra stordriftsfordeler hvis man skal utvikle flere IT-system i samme organisasjon. Da kan man ha et forretningsmodelleringsprosjekt som flere andre prosjekt kan dra nytte av.

Generisk forretningsmodell
Hvis man skal utvikle et IT-system til flere organisasjoner, så gir dette flere utfordringer. Hvor like er organisasjonene, og hvordan skal man prioritere hvis det er store forskjeller mellom de ulike organisasjonene?

Nytt foretak
Hovedutfordringen i forretningsmodellering er å finne systemets krav. Dette er derimot ikke nok hvis man skal lage et system for å støtte et nytt forretningsområde. Da må man i tillegg se om dette forretningsområdet er levedyktig.

Ombygging (BPR)
BPR står for "Buiness Process Reengineering". Begrepet brukes når man ønsker å reorganisere en bedrift eller et forretningsområde. Det er selvfølgelig det mest utfordrende scenarioet. Ombyggingen skjer i flere faser. Man må finne en visjon for den nye bedriften, bygge ned den gamle og bygge opp den nye.

Selve prosessen er beskrevet i arbeidsflytdiagrammet til forretningsmodelleringsdisiplinen. Hvilke steg som skal gjennomføres er avhengig av hvilke av de nevnte scenariene som er aktuelle. For å kunne dokumentere disse stegene har man i RUP en rekke artefakter. I RUP er det et mål å bruke "IT-språk" i forretningsmodellering. Hensikten er å prate det samme språk i forretningsmodellering som de andre disiplinene. Artefaktene som produseres i forretningsmodelleringsdisiplinen har derfor samme utforming som i de andre UML-modellene i RUP. De viktigste artefaktene som produseres i forretningsmodellering disiplinen er:

Business vision document
Her defineres hensikten og målene for forretningsmodelleringen.

Business Use-Case model
Denne modellen skal vise bedriftens påtenkte funksjoner. Disse modellene brukes videre for å finne aktører og leveranser som er relatert til systemet.

Business object model
Hvordan de forskjellige aktørene relaterer til hverandre og systemet.

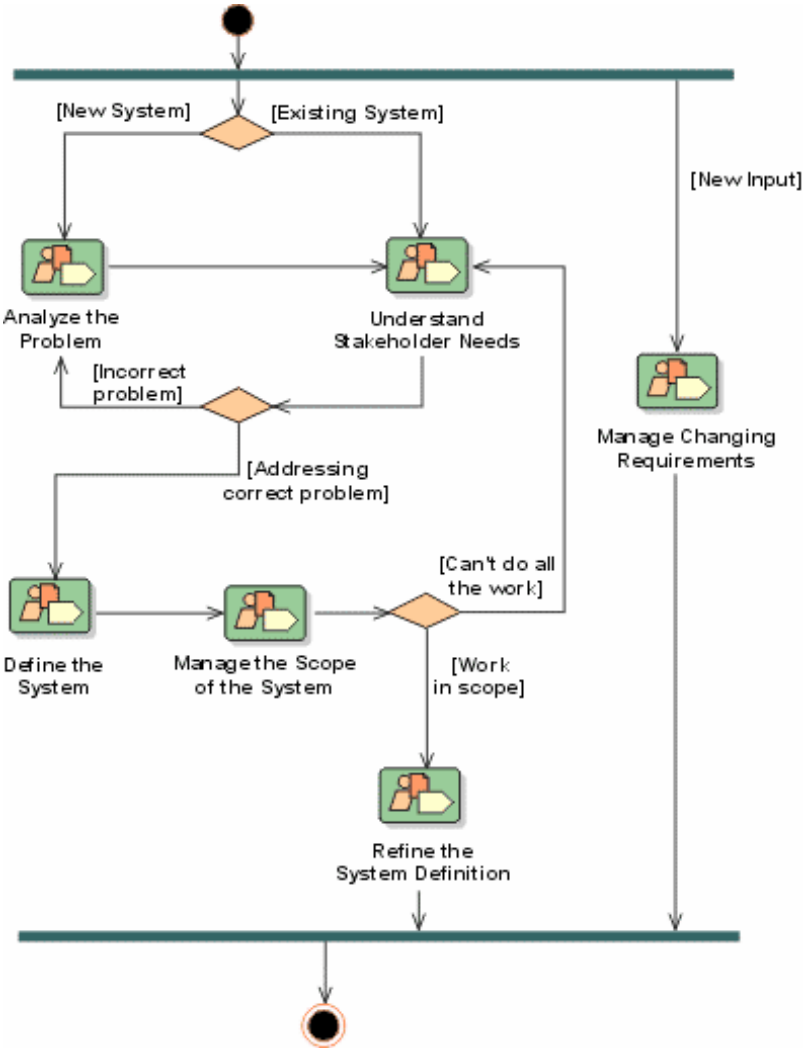
Disse artefaktene lages av prosjektdeltagere med rollene ”business process analyst” og ”business designer”

2.1.3 Kravhåndtering

Hvorfor
I systemutvikling er det ikke uvanlig å definere alle krav før utviklingen starter. Dette er vanlig i tradisjonell fossefallutvikling. Tilhengere av RUP [Jacobsen et al. 1999, Kruchten 2000, Leffingell & Widrig 2001] hevder på sin side at det er umulig å identifisere alle krav før utviklingen starter. [Kruchten 2000] hevder at kravene er dynamiske og de vil endre seg under prosjektets gang. RUPs løsning på dette er iterativ og evolusjonær utvikling. I RUP har man derfor en egen disiplin for å håndtere krav gjennom hele prosjektet.

Kruchten hevder at hensikten med kravhåndtering er:

- Å etablere og vedlikeholde enighet mellom kunder og andre interessenter i forhold til hva systemet skal gjøre og hvorfor.
- Å gi systemutviklerne en bedre forståelse for kravene.
- Å avgrense systemet.
- Gi et grunnlag for å planlegge innholdet i hver iterasjon
- Gi et grunnlag for å estimere utviklingstid og kostnad.



Figur 2.3 Arbeidsflytdiagram for kravhåndtering

Hvordan

I RUP defineres et krav som ”en betingelse eller mulighet som et system må være i overensstemmelse med”. Det finnes forskjellige typer krav. Vi har to hovedkategorier, funksjonelle og ikke-funksjonelle krav. Et funksjonelt krav er en handling som systemet må støtte. Kruchten beskriver det som ”de tingene som systemet gjør på vegne av brukerne.” Men funksjonelle krav er ikke nok for å lage et bra system. Et IT-system kan i tillegg ha en rekke ikke-funksjonelle krav. Kruchten deler de ikke-funksjonelle kravene opp i følgende kategorier: brukervennlighet, pålitelighet, ytelse og supportability (hvor enkelt det er å drive brukerstøtte).

Selve prosessen er beskrevet i arbeidsflytdiagrammet til kravhåndteringsdisiplinen. (Se figur 2.3). Prosessen består av fire hovedaktiviteter. Analysere problemet, fange krav, detaljere/raffinere og håndtere nye krav.

Analysere problemet
Her skal man skape enighet om hva som skal gjøres, identifisere interessenter og identifisere grensene og restriksjonene til systemet.

Fange krav
Ved å bruke forskjellige teknikker skal man finne kravene til systemet. Disse finner man hovedsakelig hos systemets interessenter. Systemets interessenter kan være brukere, kunden, utviklere og andre som har interesse av eller blir påvirket av systemet. Kravene som blir definert i forretningsmodelleringsdisiplinen er selvfølgelig en viktig input til denne aktiviteten.

Detaljere/raffinere
Basert på innspill fra alle interessenter skal man komme fram til de kravene systemet skal møte og hvilke rammer man skal forholde seg til. Videre skal disse detaljeres til et nivå som kan være utgangspunkt for design og implementering.

Håndtere nye krav
Krav er som nevnt i stadig endring. Det er viktig å holde orden på endringsønsker og endringer i systemkravene. Man bør holde så god oversikt at man kan spore de endringer som er gjort.

Det er i RUP definert tre roller i kravhåndteringsdisiplinen. Disse er System Analyst, Use-Case Specifier og User-Interface Designer.

System analyst
System analyst er ansvarlig for arbeidet med å fange systemets krav og å dokumentere disse. Hun har også ansvaret for å skrive overordnede Use-Case modellene basert på disse kravene.

Use-Case specifier
Use-Case Specifier har så ansvaret for å detaljere Use-Case diagrammene som blir produsert av System Analyst.

User interface designer
Med hjelp fra Use-Case diagrammene skal User-Interface Designer presentere systemet for brukerne, og sammen med brukerne så skal User-Interface Designer lage Use-Case Storyboards og en User-Interface Prototyp. Du kan lese mer om brukergrensesnittdesign i kapittel 2.1.4.

Resultat av kravhåndteringsdisiplinen er følgende artefakter:

- Interessenters ønskeliste
- Use-Case modeller
- Supplementære spesifikasjoner
- Ordbok/definisjoner
- Use-Case storyboard
- Brukergrensesnitt prototyp

2.1.4 Design av brukergrensesnitt i RUP

Brukergrensesnittdesign er ikke en egen disiplin, men er en del av kravhåndteringsdisiplinen. Jeg vi i det følgende se litt nærmere på brukergrensesnittdesign i RUP. Som nevnt tidligere er det User Interface Designer som er ansvarlig for å designe brukergrensesnittet. User Interface Designer er involvert i to aktiviteter. Disse er User-Interface Modeling og User-Interface Prototyping. De to viktigste artefaktene som User-Interface Designer produserer i disse aktivitetene er Use-Case Storyboard og User Interface Prototype. Design av brukergrensesnitt er plassert i kravdisiplinen fordi arbeidet med dette er så tett knyttet mot de andre artefaktene som er utviklet i denne disiplinen. De detaljerte Use-Case diagrammene som utvikles av Use-Case Specifier er de viktigste innspillene i denne aktiviteten. Brukergrensesnittdesign i RUP deles inn i fire hoveddeler. Modellering, prototypdesign, prototypimplementering og evaluering

Modellering
Her skal utviklerne lage en rekke artefakter som dokumenterer hvem som bruker systemet og hva slags funksjonalitet det skal ha.

Prototypdesign
På bakgrunn av disse modellene skal User-Interface Designer spesifisere brukergrensesnittet til applikasjonen. [Kruchten et al. 2001] foreslår en seks-stegs prosess som kan hjelpe User-Interface Designer med dette arbeidet. Disse seks stegene er:

1. *Identify the primary windows*
2. *Design the visualization of the primary windows*
3. *Design the operations of the primary windows*
4. *Design the property windows*
5. *Design the operations involving multiple objects*
6. *Design miscellaneous features*

Prototypimplementering
På bakgrunn av designet skal man så lage en eller flere prototyper. Det er flere måter å gjøre dette på. Kruchten gir tre alternativer: Man kan lage papirbaserte prototyper som er utviklet

med pen og papir. Disse er billige og raske å lage, men ikke like detaljerte. Disse egner seg best tidlig i et prosjekt. Senere kan man lage mer detaljerte skjermbilder ved hjelp av et tegneprogram. Disse gir mer detalj informasjon og gjør det lettere å se hvordan det ferdige systemet blir. Og etter hvert kan man lage interaktive prototyper som gir et ganske detaljerte bilde på hvordan den ferdige versjonen skal se ut.

Evaluering

Det er viktig å teste disse prototypene på andre personer. Da kan man finne feil som man selv ikke ser. Prototypen kan testes / vises fram til andre prosjektdeltagere, brukervennlighetseksperter og brukere.

2.1.5 RUPs User Experince plug-in

RUP har en egen ”User Experince plug-in”. Denne skal gjøre det lettere å utvikle brukervennlige IT-system. Her introduserer de nye aktiviteter, artefakter og roller.

De viktigste nye aktivitetene er:

- Develop the User-Experience Guidelines
  Utvikling av retningslinjer som er førende for systemutviklingsprosessen.

- Model the User-Experience
  Her bruker man kilder som kravene og Use-Case modellene for å modellere brukeropplevelsen (se liste over artefakter).

- Review the User-Experience
  Verifisere at brukeropplevelsemodellen møter funksjonelle og brukervennlighetskrav. At systemet er i henhold til relevante retningslinjer og aksepteres av interessentene.

Disse aktivitetene fører til artefakter som:

- Screen
  En abstrakt representasjon av de ulike brukergrensesnittene i systemet.

- Navigation Map
  Systemets overordnede navigasjon.

- Use-Case Storyboard
  En visuell beskrivelse av hvordan man tror systemet vil bli brukt av sluttbrukeren.

- User-Experience Guidelines
  Retningsliner for hvordan design prosessen skal gjennomføres.

- User-Experience Model
  Beskriver hvordan systemet oppleves for sluttbrukeren av systemet. Hvordan hun navigerer og hvordan de ulike elementene oppfører seg.

Det blir i tillegg introdusert to nye roller:

- User-Experience Designer
  ”User-Experience Designer” skal designe systemet basert på kravene, Use-Case modellene, storyboards og andre artefakter. Hun har også ansvaret for å utvikle prototyper av systemet,

- User-experience reviewer
  ”User-experience reviewer” er ansvarlig for å kvalitetssikre de ulike artefaktene som produseres.

Utvidelsen introduserer flere konsept og begreper som benyttes innen brukersentrert design. Derimot adresser ikke utvidelsen viktige brukersentrerte konsepter som brukermedvirkning, brukertesting, brukervennlighetskrav i dybden. Disse temaene berøres kun i korte trekk. Det eksister kun korte beskrivelser av de ulike aktivitetene og artefaktene (mellom 1 og 2 A4 sider). I tillegg til å fokusere på ”user-experience”, så er forretningsmodellering også et viktig tema i RUPs ”User Experince plug-in”. Ifølge Göransson, Lif og Guliksen [Göransson 2003] går ikke denne RUP-utvidelsen nok i dybden til å kunne si noe om brukersentrert design. De skriver følgende:

> *”The User Experience plug-in to RUP is developed and maintained by Rational. Primarily, it is focused on website projects, and does not cover the whole user-centred design process. There is a strong focus on creative design and business development, not so much on usability. Therefore, we see a need for a more general plug-in with a strong usability focus.” [side 4,* Göransson 2003*]*

Jeg skriver mer om Göransson, Lif og Guliksens kritikk av RUP i kapittel 4.4.1.

2.1.6 Unified Modeling Language

Unified Modeling Language (UML) er et standardisert metodespråk med forskjellige teknikker egnet til å visuelt modellere et IT-system. UMLs ulike diagrammer brukes til å beskrive et IT-system fra ulike ståsted. De forskjellige diagrammer brukes blant annet til å beskrive kravene fra brukerne, modellering av kildekode og databasemodellering. Flere verktøy kan brukes til å tegne UML-diagrammer. Det mest utbredte og kjente er Rational Rose.

2.1.7 Use-Case modeller

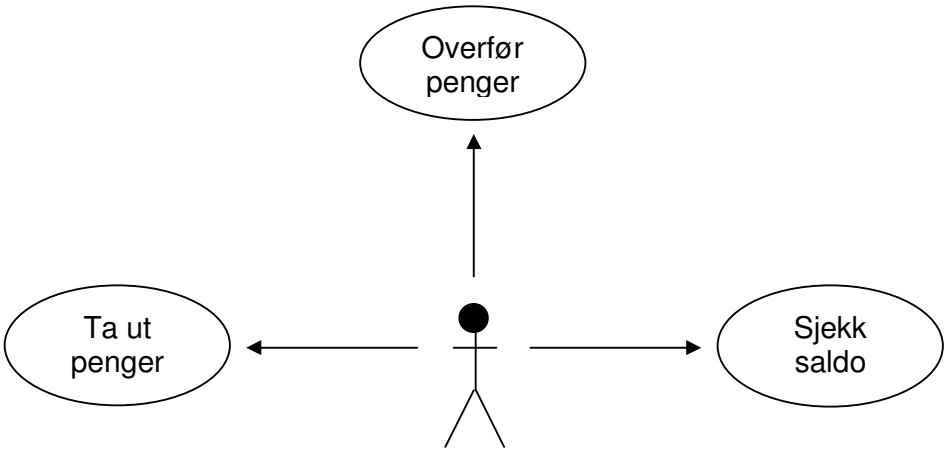
En av de mest brukte modellene fra UML er Use-Case modeller. Kruchten definerer en Use-Case på følgende måte:

> *"A Use-Case is a sequence of actions a system performs that yields an observable result of value to a particular actor."* [side 98, Kruchten 2000]

Videre definerer han en aktør(actor) på følgende måte.

> *"An actor is someone or something outside the system that interacts with the system."* [side 98, Kruchten 2000]

En Use-Case modell er alle Use-Casene for et system eller deler av et system, og alle aktørene som jobber med systemet. Use-Case modeller gjøres ofte visuelle ved hjelp av et Use-Case diagram. Eksempel på et Use-Case diagram ser du i figur 2.4.



Figur 2.4 Eksempel på Use-Case diagram

Use-Case-modeller har en rekke funksjoner i RUP. Use-Case-modellene blir til i kravdisiplinen og brukes deretter i alle de ulike disiplinene i RUP. I kravdisiplinen brukes Use-Case-modellene blant annet som innspill til brukergrensesnittdesignet; i analyse og design disiplinen brukes de som bro mellom krav og design(teknisk); i implementasjonsfasen brukes Use-Case-modellene blant annet til å finne områder av applikasjonen som trenger optimalisering; og som grunnlag for testplaner i testdisiplinen. De skal i tillegg fungere som et felles språk for kunder, brukere og systemutviklere. Use-Case-modellene er en svært sentral del av RUP.

2.2 Brukersentrerte metoder

Jeg har i kapittel 1 definerte begrepet brukervennlighet. Jeg vil her se nærmere på hvordan man kan utvikle brukervennlige IT-system.

2.2.1 ISO 13407

Det finnes en rekke ulike brukersentrerte metoder. Jeg vil ikke forsøke å gi oversikt over alle disse. Jeg vil derimot fokusere på ISO 13407. ISO 13407 er en internasjonal standard for hvordan man skal gå fram for å utvikle brukervennlige IT-system. ISO 13407 tar sikte på å være en guide for ledere og systemutviklere. Denne guiden er prosessbasert og forteller hvilke aktiviteter som bør være del i et utviklingsprosjekt som har behov for brukersentrert design.

2.2.2 ISO 13407, suksesskriterier

ISO 13407 er ment som en støtte til eksisterende metoder for systemutvikling. Standarden kan integreres inn i forskjellige typer utviklingsprosesser. På samme måte som RUP, er ISO 13407 basert på suksesskriterier. ISO 13407 suksesskriterier er:

- Aktiv involvering av brukere og en klar forståelse av brukskontekst.
- En passende fordeling av funksjon mellom brukere og teknologi.
- Iterasjon av designløsninger.
- Tverrfaglig design.

Aktiv involvering av brukere og en klar forståelse av brukskontekst
Ifølge ISO 13407 er det viktig å involvere brukere i systemutviklingsprosjekt. Dette begrunnes med at brukere gir verdifull informasjon om systemets brukskontekst, sine oppgaver og hvordan de vil jobbe med det framtidige systemet. Med brukermedvirkning er det også lettere å få aksept for det framtidige systemet hos brukerne og de vil kunne føle et eierskap til systemet. Hva slags brukerinvolvering man bør ha, avhenger av hva man ønsker å oppnå, og hva man ønsker å utvikle. Ved utvikling av skreddersydde løsninger, kan man involvere brukere direkte i utviklingsprosessen. Ved utvikling av mer generelle produkter eller hyllevaresystemer er ofte brukermassen delt, og ikke alltid like lett å få tak i. Det er likevel viktig at brukere deltar i utviklingsprosessen. Da kan man identifisere krav, få tilbakemelding på løsninger og få testet designet fortløpende.

En passende fordeling av funksjon mellom brukere og teknologi
Et viktig element i utviklingen av IT-system er fordeling av funksjon mellom bruker og applikasjon. Avgjørelsen bør ikke bare tas på grunnlag av hvilke funksjoner teknologien faktisk kan utføre, og overlate resten av funksjonene til brukerne. Avgjørelsene bør baseres på flere faktorer. For eksempel: hvilke muligheter og begrensninger mennesker har i forhold til teknologien? Man bør ha fokus på å utvikle meningsfulle oppgaver for de ulike brukerne. Representative brukere bør være involvert i disse avgjørelsene. ISO 9241-2 og ISO 10075 går dypere i forhold til dette temaet.

Iterasjon av designløsninger
Som RUP vektlegger også ISO 13407 iterativ utvikling. Å få tilbakemelding på løsninger og få testet designet fortløpende (iterativt), reduserer ifølge ISO 13407 risikoen for at systemet ikke skal møte de kravene som brukeren og organisasjonen har. Iterasjon tillater tidlige designløsninger å bli testet i realistiske scenarier, hvor resultatet fra disse undersøkelsene igjen blir input til kontinuerlig forbedring av løsningen.

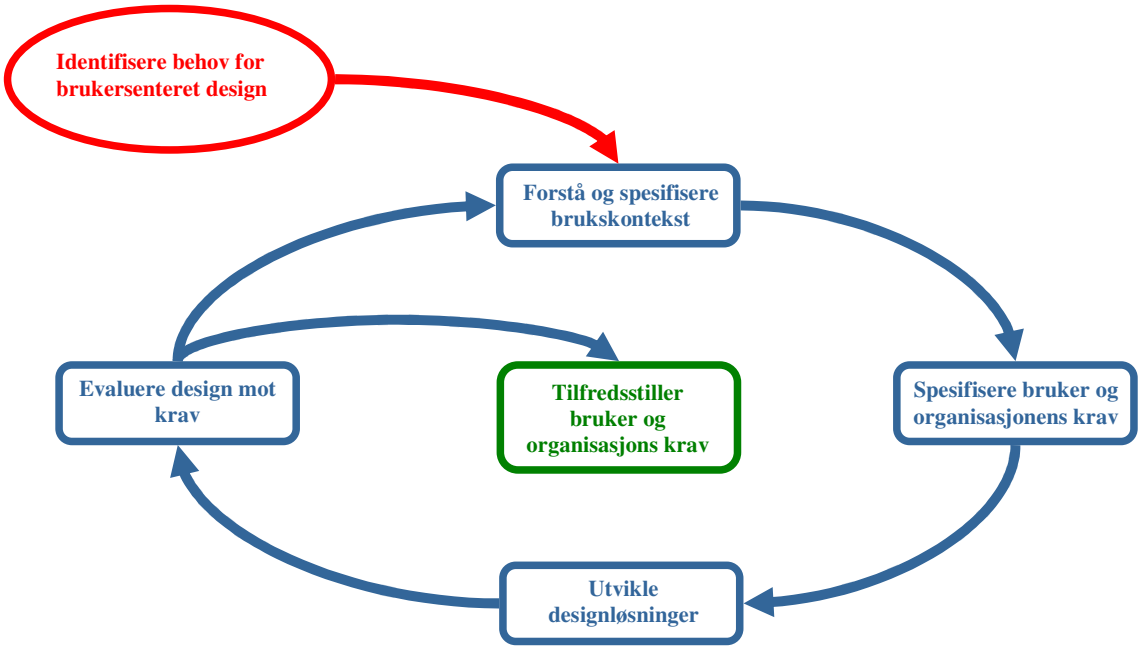
Tverrfaglig design
Brukersentrert design er avhengig av mange ulike kompetansetyper. Det er derfor viktig å sette sammen en tverrfaglig gruppe som inneholder de kompetanseområdene som er nødvendige. Rollene i en tverrfaglig gruppe kan være:

- Sluttbruker
- Kjøper av produkt, eller sluttbrukers sjef
- Applikasjonsdomenespesialist, business analytiker
- Systemanalytiker, systemingeniør, programmerer
- Markedsfører, salgsperson
- Brukergrensesnittdesigner, grafisk designer
- Ekspert på menneskelige faktorer og ergonomi, spesialist på HCI
- Teknisk forfatter, opplærings- og brukerstøttepersonale

2.2.3 ISO 13407, Prosess

Det bør utvikles en plan for å spesifisere hvordan de brukersentrerte aktivitetene passer inn i den totale utviklingsprosessen. Planen bør ifølge ISO 13407 identifisere:

- De brukersentrerte designprosess-aktivitetene. Å forstå og identifisere brukerkontekst, spesifisere brukeres og organisasjonens krav, produsere prototyper og evaluere design opp mot brukerens kriterier
- Hvordan disse aktivitetene skal integreres med andre systemutviklingsaktiviteter.
- Hvem som har ansvaret for brukersentrert utvikling og bredden på kunnskapen og synspunktene de har.
- Effektive kommunikasjonskanaler mellom designaktiviteter og andre aktiviteter, og metoder for å dokumentere disse aktivitetene.
- Integrere milepæler for brukersentrerte aktiviteter inn i det overordnede designet og utviklingsprosessen
- Tidsrammer som tar høyde for brukersentrert design.



Figur 2.5: Prosessdiagram ISO 13407

ISO 13407, Forstå og spesifisere brukskontekst
Det er viktig å forstå systemets brukskontekst for å kunne guide design avgjørelser og som et grunnlag for å evaluere systemet. Man bør alltid samle inn informasjon om brukskontekst når man skal utvikle nye IT-system. En beskrivelse av brukskonteksten bør i følge ISO 13407 inneholde:

- Karakteristikker av potensielle brukerne: kunnskap, erfaring, utdanning, opplæring, fysiske attributter, vaner, preferanser og evner.

- Oppgavene brukerne skal utføre: beskrivelsen bør inkludere det overordnede målet for systemet. Oppgaver som kan påvirke brukbarhet bør også beskrives sammen med bruksfrekvens og varighet.

- Miljøet systemet skal brukes i: dette inkluderer maskinvare, programvare og materialene som brukes. Beskrivelser av det fysiske og sosiale miljøet bør beskrives. Disse kan inkludere relevante standarder, det omliggende tekniske miljøet (LAN/WAN), kontorlokaler, møbler, temperatur, lys, støynivå m.m. Organisasjonens lover og regler bør i tillegg være med, samt sosiale og kulturelle aspekter som arbeidspraksis, organisasjonsstruktur og holdninger.

Etter å ha gjennomført aktiviteten ” Forstå og spesifisere brukskontekst” bør man sitte igjen med en fyldig beskrivelse av brukerne, oppgavene til brukerne og miljøet til brukerne.

Spesifisere brukerens og organisasjonenes krav
I tillegg til å spesifisere funksjonelle og ikke-funksjonelle krav, bør man i følge ISO 13407 spesifisere bruker- og organisasjonskrav. De følgende aspekter bør evalueres for å kunne identifisere relevante krav:

- Krav til ytelse. Disse kravene må sess i forhold til operasjonelle og finansielle mål
- Relevante lovfestede krav, inkludert sikkerhet og helse
- Samarbeid og kommunikasjon mellom brukere og andre relevante parter
- Brukerens arbeidsoppgaver
- Oppgaveutførelse
- Arbeidsdesign og organisasjon
- Håndtering av forandring, inkludert opplæring og ansatte som er involvert i forandringen
- Gjennomførbarhet i forhold til bruk og vedlikehold
- Brukergrensesnittet og design på arbeidsstasjon

Produsere designløsninger

For å produsere designløsninger bør man ifølge ISO 13407 benytte seg av etablert "state of the art"-kunnskap, erfaringen til prosjektdeltagerne og resultatet fra brukskontekst analysen. I ISO 13407 er denne aktiviteten delt opp i følgende seks steg:

- *Bruke eksisterende kunnskap til å utvikle designforslag med tverrfaglig input*
  For å produsere designløsninger kan man hente hjelp fra en rekke ulike faggrener. Dette kan være kunnskap og teori fra ergonomi, psykologi, kognitiv vitenskap, produktdesign og andre relevante disipliner.
- *Mer konkrete designforslag, ved hjelp av simuleringer, modeller og mock-ups*
  Ved å bruke ulike typer simuleringer vil man kunne kommunisere mer effektivt med sluttbrukerne. Dette vil føre til at designere, sluttbrukere og andre interessenter på et tidlig tidspunkt vil kunne se hvordan en løsning kan bli, og komme med innpill i forhold til dette.
- *Presentere designforslaget til brukere og la dem utføre reelle eller simulerte oppgaver*
  Ved hjelp av papirbaserte prototyper vil man kunne involvere sluttbrukere tidlig i prosessen. Ved å gjøre dette vil man tidlig kunne få svar på spørsmål knyttet til ulike designaspekter. For eksempel, hvor lett det er å manøvrere seg gjennom IT-systemets navigasjon.
- *Endre designet i henhold til tilbakemelding fra brukerne og å ha en iterativ prosess fram til de brukersentrerte målene er nådd*
  Etter hvert som man får tilbakemeldinger for brukeren kan utviklerne redesigne systemet. Man kan begynne med papirprototyper i et tidlig tidspunkt når det er mange ukjente aspekter. Senere kan man lage mer realistiske prototyper og man kan teste løsningene i mer realistiske miljø.
- *Håndtere iterasjonene*
  Ifølge ISO 13407 bør aktivitetene nevnt ovenfor loggføres. Dette fordi det vil hjelpe med å styre den iterative prosessen.

Evaluere design opp mot krav

Evaluering er en viktig del av brukersenteret design og bør være en del av alle fasene i systemutviklingsprosessen. Evaluering er viktig for å forbedre eksisterende design, finne om bruker og organisasjonskrav er møtt og å overvåke langtidsbruk av systemet. Det er viktig å starte evalueringsprosessen så tidlig som mulig. Jo senere en feil blir oppdaget, jo dyrere blir det å rette den.

2.2.4 Artefakter

Mens RUP benytter seg av UML, er det i brukersentrert design vanlig å benytte seg av andre type artefakter. Jeg vil her kort forklare tre av disse: personas, scenarios og work models.

Scenarioer

Scenarioer er ifølge [Carrol & Go 2004] beskrivelser som inneholder aktører, bakgrunnsinformasjon om en aktør og antagelser om deres miljø, deres mål og en sekvens av handlinger eller hendelser. Scenarioer kan bli presentert på ulike måter. Det kan være for eksempel være som tekst, dreiebok eller som video. Hensikten med scenarioer kan være å analysere brukernes oppgaver, forutse framtid arbeid, lage mocups og protoyper, evaluering av ferdig system, lage brukerveileding og for å begrunne designvalg.

For mer om scenarioer se: Making Use av John M. Carroll [Carroll 2000].

Personas

Personas er karakteristikker av brukere. Disse karakteristikkene inneholder ulik type innformasjon om en tenkt bruker av et IT-system. Dette kan være informasjon om hvilke mål de har, hva deres holdninger og følelser er i forhold til arbeidsoppgaver og andre viktige spørsmål. Hensikten med personas er å guide designet av et IT-system og er ofte et supplement til scenarioer. Vanligvis er personas laget ved hjelp av data fra en studie av brukere.

For mer om personas se: About Face 2 av Alan Cooper og Robert Reimann[Cooper & Reiman 2003].

Work models

Work models er visuelle modeller på samme måte som UML, men work modeles har et noe annet fokus enn UML. Work models tar sikte på å visualisere arbeid fra ulike ståsted. De er mindre tekniske en UML-modeller og fokuserer i større grad på å modellere ikketekniske aspekter som kulturelle faktorer og brukskontekst.

For mer om work models se: Contextual Design av Hugh Beyer og Karen Holtzblatt [Beyer & Holtzblatt 1998].

2.3 Hva er brukermedvirkning

ISO 13407 legger stor vekt på brukermedvirkning, men hva som legges i brukermedvirkning varierer avhengig av hvem du spør. Pelle Ehn mener at brukere må involveres når nye IT-system skal utvikles [Ehn 1993]. Han mener at ferdigheter hos arbeidere ikke kan karakteriseres med en formell beskrivelse, noe som ofte er praksis i utviklingsprosjekter. Ehn bruker UTOPIA-prosjektet som eksempel. Prosjektet startet i 1981 og jobbet med layout og bildebehandling i avisindustrien. I dette prosjektet blir det utviklet en designmetode som fikk navnet ”Tool Perspective”. Denne tilnærmingen var inspirert av hvordan verktøy blir designet innen tradisjonell håndverk. Her skal designere jobbe sammen med brukerne av systemet. Disse to gruppene skal lære av hverandre. Designere skal lære brukerne om teknologiske muligheter, mens brukerne skal lære designerne om sitt arbeid. Det er viktig at systemet blir designet av erfarne brukere og dyktige designere. Det er også viktig at prosessen er iterativ med prototyper og skjermbildetegninger. Tradisjonelle artefakter som datastruktur- og informasjonsflytdiagram fungerer ikke ifølge Ehn.

Pelle Ehn representerer et syn på hvordan designere skal bringes nærmere til brukerne, men det er naturlig nok ikke enighet om hvordan dette skal gjøres. Et noe annet syn blir fremsatt av Jakob Nielsen. Nielsen har blitt veldig kjent for sin kamp for brukervennlige IT-system og websider. Hans webside useit.com er blant de mest besøkte innen emnet brukervennlighet. I artikkelen “First Rule of Usability? Don't Listen to Users” [Nielsen 2001] gir Jakob Nielsen utrykk for en annen filosofi om hvordan designere bør forholde seg til brukere. Han mener at designere ikke skal være så opptatt av hva brukere sier, men heller fokusere hva de gjør. IT systemer skal ikke utvikles i samarbeid med brukere, men designere må bruke mye ressurser på å forstå dem. Han sier blant annet følgende:

> *“But ultimately, the way to get user data boils down to the basic rules of usability:*
>
> - *Watch what people actually do.*
> - *Do not believe what people say they do.*
> - *Definitely don't believe what people predict they may do in the future.” [Nielsen 2001]*

Nielsen har liten tro på brukeres evne til å selv beskrive sine problemer. Han mener at det er viktig å forstå brukere og kontekst, men man må stille de riktige spørsmålene og ikke ta noe for god fisk. Nielsen sier videre:

> *”However, when collecting preference data, you must take human nature into account. When talking about past behavior, users self-reported data is typically three steps removed from the truth:*

- *In answering questions (particularly in a focus group), people bend the truth to be closer to what they think you want to hear or what's socially acceptable.*

- *In telling you what they do, people are really telling you what they remember doing. Human memory is very fallible, especially regarding the small details that are crucial for interface design. Users cannot remember some details at all, such as interface elements that they didn't see.*

- *In reporting what they do remember, people rationalize their behavior. Countless times I have heard statements like "I would have seen the button if it had been bigger." Maybe. All we know is that the user didn't see the button." [Nielsen 2001]*

Nilsen og Ehn prøver å oppnå det samme. Å skape en bro mellom de som designer IT-system og de som bruker dem (Ehn er også opptatt av demokrati på arbeidsplassen). De har derimot svært ulike tilnærminger. I tillegg til Nilsens og Ehns tilnærminger finnes det et hav av metoder som prøver å oppnå det samme.

[Svanæs og Seland 2003] foreslår bruk av rollespill i designprosessen. Hensikten med rollespill er å kunne utforske forskjellige bruks scenarioer ved hjelp av skuespill mellom brukere og designere.

En annen tilnærming er å integrere etnografisk metode i systemutviklingsprosessen [Huges et al. 1994, Cooper & Reimann 2003]. Bruk av etnografisk metode i systemutviklingsprosessen blir begrunnet med et ønske om å kunne forstå den "virkelige verden". Faktiske arbeidsprosesser følger sjeldent bedriftens offisielle arbeidsinstrukser [Suchman 1987, Miller 1992] og etnografi kan være en egnet metode for å kartlegge de faktiske forhold. Denne kunnskapen kan så brukes til å informere designet av et nytt IT-system. Vanligvis er etnografiske studier tidkrevende og de som gjennomfører de tilbringer ofte måneder eller år sammen med sine informanter. Tradisjonell etnografi tar heller ikke hensyn til at kunnskapen de dokumenterer skal brukes til å designe noe nytt. Etnografiske studier som tar så lang tid vil ikke være praktisk i et utviklingsprosjekt. Man må derfor tilpasse etnografi til utviklingsprosessen, og [Huges et al. 1994] foreslår flere modeller for dette. Et av disse er "Quick and Dirty Ethnography". Her utfører en etnograf korte feltstudier(2-3 uker) hos potensielle brukere av systemet. Deretter blir funnene i disse feltstudiene rapportert til en designer som da kan ta hensyn til disse dataene. Dette skjer iterativt.

En metode som har fått mye oppmerksomhet er Contextual Inquiry[Beyer & Holtzblatt 1998]. Contextual Inquiry er en feltstudiemetode tilpasset de behovene man har i et systemutviklingsprosjekt. Contextual Inquiry er basert på tre hovedprinsipper:

- Å forstå brukskonteksten et system benyttes i er essensielt for å designe elegante løsninger.
- Brukeren er en partner i designprosessen.
- Metoder som benyttes i brukersentrert design (for eksempel contextual inquiry og brukertesting) må ha et fokus.

Metoden bygger på tett samarbeid mellom designer og bruker, og har et høyt fokus på å forstå brukskonteksten til et system. Vi kan se på Contextual Inquiry som en hybrid mellom intervjuer og observasjon. Brukere blir intervjuet i sin kontekst mens de uførerer relevante oppgaver.

Brukermedvirkning er ikke et entydig begrep. Det er mange måter å involvere brukeren på. Vi må bør istedenfor prate om type brukermedvirkning. Enid Mumford foreslår tre begreper: Deltagelsens innhold, deltagelsens struktur og deltagelsens prosess[Mumford 1984].

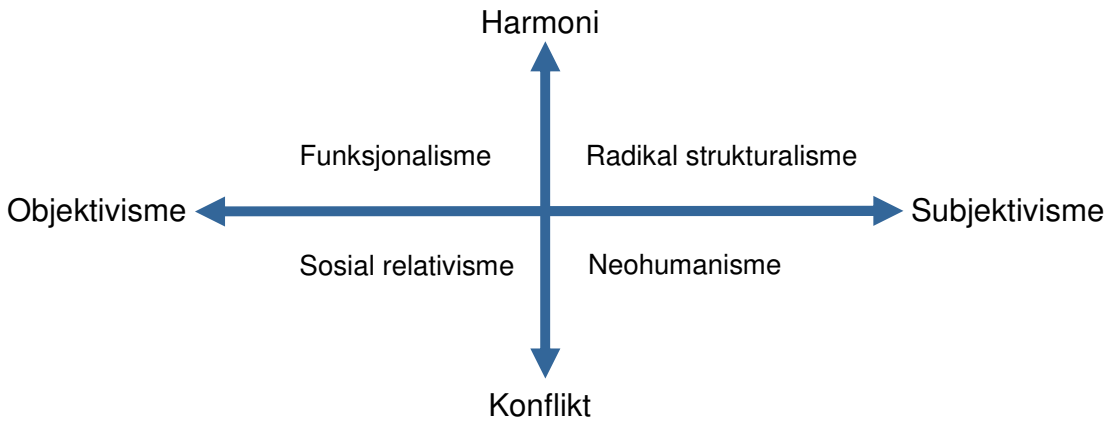
Deltagelsens innhold er hvilke tema skal man ta avgjørelser om; hva som er innenfor ansvarsområdet til gruppen og hva som er utenfor. *Deltagelsens struktur* er hvordan deltagelse finner sted i prosjektet. Her er det mange alternativ. Man kan ha deltagere som er aktivt del av prosjektet og som har en faktisk maktposisjon i prosjektet. Et annet alternativ er at man kan ha eksterne grupperinger som på en eller annen måte kan påvirke prosjektet. Disse kan påvirke ved å utøve press på en eller annen måte. *Deltagelsen prosess* er hvordan man kommer fram til kunnskapen som skal påvirke uformingen av systemet, og hvordan man kommer fram til effektive arbeidsforhold mellom de forskjellige partene. Disse begrepene gjør oss i stant til å være mer konkret når vi omtaler forskjellige teknikker innen brukermedvirkning.

2.4 Paradigmer i systemutvikling

Det eksisterer et hav av ulike utviklingsmetoder. Det kan derfor være interessant å kartlegge de ulike tilnærmingene. Hirschheim & Klein gjør et forsøk på å dele forskjellige tilnærminger til systemutvikling inn i fire forskjellige paradigmer[Hirschheim 1989]. Disse paradigmene blir presentert ved hjelp av hjelp av to akser. Disser aksene representerer antagelser gjort innen forskjellige paradigmer. Aksene er objektivisme mot subjektivisme og harmoni mot konflikt (se figur 2.6).

Hva mener de med objektivisme? På den siden hører de ortodokse utviklingsmetodikkene hjemme. Dette er de "tradisjonelle" utviklingsmetodikkene. Dette er metodikker som strukturert analyse og objektorientert modellering. Dette er utviklingsmetoder som benytter seg av metoder fra realfagene for å studere menneskelige anliggender. Her har man fokus på det matematiske og ingeniøraktige i systemutvikling. I den subjektiviske aksen er man i større grad opptatt av sosiale forhold og individets subjektive oppfating. Verden anses som vanskelig å forstå og man tar i bruk metoder som etnografi og iterativ utvikling.

På harmonisiden har vi de utviklingsmetodikkene som tar for gitt at utvikling skjer i en harmonisk verden hvor alle jobber mot et delt mål. På den andre siden har vi utviklingsmetodikker som tar høyde for konflikt. For eksempel: ulike grupperinger har forskjellige mål i forhold til systemet som skal utvikles. Hirschheim og Kleins argument er at utfordringen ikke bare ligger i å forstå brukere og organisasjon. Man må innse at dette skjer i en verden hvor ulike grupper har forskjellige interesser. Disse grupperingene har forkjelig agendaer som kan være i konflikt med hverandre. Dette er en utfordring som ikke er et tema i mange utviklingsmetodikker.



Figur 2.6: Paradigmer i systemutvikling

Hirschheim og Kleins gir navn til de fire paradigmene. Disse er funksjonalisme, sosial relativisme, radikal strukturalisme og neohumanisme.

2.5 Rasjonalitet

Hvordan kan vi ta høyde for konflikter og skjev maktfordeling i en utviklingsmetodikk? For å se nærmere på dette kan vi bruke John Forester's kategorisering av begrenset rasjonalitet [Forester 1989]. Forester har laget en oversikt over grader av kompleksitet en leder vil møte i en organisasjon. Ved å kjenne hvor komplekst et problem er vil man kunne finne ulike strategier som passer for hver situasjon. De fire typene er: kognitive begrensninger hos prosjektleder, sosial differensiering –arbeidsdeling, pluralistisk konflikt og reproduserbare strukturelle skjevheter.

Begrenset rasjonalitet 1: Kognitive begrensninger

Dette er Foresters enkleste scenario. Vi har kun en planlegger som ikke er i konflikt med den eksterne verden. Utfordringene for denne planleggeren ligger i å ta de riktige avgjørelsene. Hun har tvetydige problemstillinger og må forholde seg til imperfekt informasjon. Selv om planleggeren er oppmerksom på feilkilder så har hun ikke nødvendigvis tid og ressurser til å korrigere.

Begrenset rasjonalitet 2: Sosial differensiering –arbeidsdeling

Alle utforingene i begrenset rasjonalitet en gjelder i dette scenarioet. Men nå er ikke vår aktør lenger alene. Nå må hun forholde seg til andre aktører. Alle disse aktørene er samarbeidsvillige, men kan ha andre perspektiv og en annen virkelighetsforståelse. Selv om de er samarbeidsvillige aktører, med samme mål, kan de utilsiktet komme til å motarbeide hverandre

Begrenset rasjonalitet 3: Pluralistisk konflikt

Inkluderer utforingene i begrenset rasjonalitet en og to. I tillegg forutsettes at ikke alle aktører er fullt ut samarbeidsvillige. Her kan det være interessemotsetninger mellom de ulike aktørene. Disse interessemotsetningene kan være både åpne og hemmelige.

Begrenset rasjonalitet 4: Reproduserbare strukturelle skjevheter

Her inkluderes begrenset rasjonalitet en til tre. I tillegg til inntressemotsetningene i begrenset rasjonalitet tre, så er aktørenes evne til å handle og investere ulikt fordelt, noe som setter sterke føringer på hvordan aktørene handler. Disse skjevhetene er også reproduserbare.

3 Forskningsmetode

Forskning er en organisert og systematisk måte å finne svar på spørsmål. I dette kapittelet vil jeg se nærmere på hva mine forskningsspørsmål var og hvordan jeg har gått fram for å finne svar.

3.1 Forskningsspørsmål

3.1.1 Utgangspunktet

Temaet i denne oppgaven er forholdet mellom RUP og brukersentrert systemutvikling.

> **Hvis vi tar som utgangspunkt at brukersentrerte teknikker fører til økt brukervennlighet: I hvor stor grad er RUP brukersentrert, og hvilke grep kan vi eventuelt gjøre for å få en mer brukersentrert utviklingsprosess?**

En måte å angripe denne problemstillingen er å se på de teoretiske forskjellene og likhetene mellom RUP og brukersentrert systemutvikling. På hvilke områder er de like og hvordan skiller de seg fra hverandre. En måte å gjøre dette på er å analysere beskrivelsene av RUP og brukersentrert systemutvikling, og å sammenligne disse.

Det kan være stor forskjell på teori og praksis i hvordan systemutviklingsprosjekt blir gjennomført. Noen ganger blir utviklingsmetodene brukt på helt andre måter enn det som var intensjonen. Det kan for eksempel være organisatoriske faktorer som legger føringer for hvordan en utviklingsmetode kan brukes [Grudin 1991, Gruding & Poltrock 1994]. Det er derfor svært interessant å se på hvordan RUP og brukersentrerte metoder praktiseres. En empirisk studie vil forhåpentligvis gi oss større kunnskap om utfordringene ved å integrere brukersentrerte metoder i RUP.

3.1.2 Forskningsspørsmål

På bakgrunn av forrige avsnitt har jeg kommet fram til følgende forskningsspørsmål:

- Teori: Hvordan er RUPs tilnærming til systemutvikling sammenlignet med brukersentrerte metoder?

- Praksis: Hvilke faktorer påvirker den faktiske anvendelsen av brukersentrert metode i RUP-prosjekter?

- Anbefalinger: Hvordan kan brukersentrerte metoder integreres i RUP?

Svaret på det første spørsmålet er basert på beskrivelser av RUP. For å svare på spørsmålet vil jeg se på noen av de mest sentrale RUP-tekstene og sammenligne disse med brukersentrert design. For å svare på det andre spørsmålet, har jeg gjennomført en case-studie av et RUP prosjekt med ambisjoner om å benytte seg av brukersentrerte metoder. Og svaret på det tredje forskningsspørsmålet mitt baserer seg på funnene i de to første.

3.2 Forskningsmetoder - teori

Hvilken strategi er best egnet for å svare på spørsmålene som er gitt i forrige avsnitt? Det finnes mange vitenskapelige metoder å velge mellom. For eksempel, hva er best egnet av kvalitativ og kvantitativ metode? Jeg vil her se nærmere på noen forskjellige alternativ.

3.2.1 Kvalitativ versus kvantitativ metode

Når man skal lage et forskningsdesign er det mange aspekter man må ta hensyn til. En dimensjon er forskjellene på kvalitativ og kvantitativ metode.

Kvantitativ metode

I kvantitativ metode bruker man metoder som spørreskjema, strukturerte intervju og strukturert observasjon. Slike studier har ofte mange deltagere og man streber etter å finne et representativt utvalg av dem man studerer. For eksempel, når man skal gjennomføre en meningsmåling over politiske partier vil man strebe etter å finne et representativt utvalg av stemmeberettigede i det aktuelle landet. Resultatet av en kvantitativ studie er tall og statistikk. Kvantitativ metode brukes når man ønsker breddekunnskap. Eksempler på forskningsspørsmål i kvantitativ forskning kan være:

- Hvem hadde fått regjeringsmakt hvis det hadde vært norsk valg i dag?
- Hvor vanlig er RUP som systemutviklingsmetodikk i prosent av norske IT-selskap?

Kvalitativ metode

I kvalitativ metode bruker man metoder som observasjon, intervju og dokumentanalyse. Slike studier har få deltagere sammenlignet med kvantitative studier og foregår ofte over lenger tid. I kvalitative studier ønsker man å få dybdekunnskap om ett eller flere tema. Resultatene av en kvalitativ studie kan være kunnskap om adferd, erfaringer, opplevelser, tanker, forventninger eller holdninger hos enkeltindivid og grupper. Eksempler på forskningsspørsmål i kvalitativ forskning kan være:

- Hvordan tolker norske velgere sitt partis politiske budskap?
- Hvordan tolkes RUP av systemutviklere?

Forskeren har en svært aktiv rolle i kvalitative studier og må være svært bevisst på hvordan hun opptrer. Ved å være tilstede påvirker forskeren også utfallet av det hun studerer. Dette er en viktig faktor når man skal evaluere validiteten til en kvalitativ studie.

Det er ikke uvanlig å kombinere disse to teknikkene. For å få et bra resultat i en kvantitativ undersøkelse er det viktig å stille de riktige spørsmålene. Dette kan være vanskelig hvis man ikke kjenner til det miljøet man ønsker å undersøke. Da kan det være lurt å starte med en kvalitativ undersøkelse. Ved å gjennomføre denne undersøkelsen får man større kunnskap om

det aktuelle miljøet. Noe som igjen kan gjøre det lettere å formulere spørsmålene i en kvantitativ undersøkelse. Kvalitativ og kvantitativ forskning kan kombineres på mange ulike måter. Man kan bruke kvalitative undersøkelser for å forklare et mønster man finner i kvantitative undersøkelser. Det er også mulig å bruke kvantitative undersøkelser for å generalisere funn i en kvalitativ undersøkelse.

3.2.2 Intervju

Intervju er en teknikk som blir brukt innen de fleste samfunnsvitenskapelige forskningstradisjonene. De forskjellige tradisjonene har forskjellige varianter, men det er vanlig å skille mellom tre typer basert på struktur og standardisering [side 270, Robson 1993].

Strukturerte intervju
Et strukturert intervju består av forhåndsdefinerte spørsmål. Disse spørsmålene skal gjengis med korrekt ordlyd av intervjuer og blir vanligvis opplest i bestemt rekkefølge. Skillet mellom strukturerte intervju og vanlige spørreundersøkelser er informantens mulighet for åpne svar. I tradisjonelle kvalitative spørreundersøkelser må informanten velge mellom ulike svaralternativ.

Semistrukturerte intervju
I semistrukturerte intervju har man forhåndsdefinerte spørsmål, men intervjueren har større frihet til å gjøre endringer. Intervjueren kan endre spørsmålene underveis, komme med tilleggsopplysninger og fjerne spørsmål som ikke passer inn i den gitte konteksten. Intervjueren har her en friere rolle, og kan gjøre egne tolkninger underveis.

Ustrukturerte intervju
I ustrukturerte intervju har man et overordnet tema, men ingen eller svært få forhåndsdefinerte spørsmål. Mye er opp til informanten, men intervjueren må sørge for at man holder seg på det opprinnelige temaet. Samtalen kan være fullstendig uformell.

Semistrukturerte og ustrukturerte intervjuer blir ofte brukt i kvalitative studier. De er godt egnet til dette fordi de er fleksible. Intervjueren kan endre og korrigere på spørsmålene underveis og kan stille oppklaringsspørsmål. Over tid kan man skape et forhold mellom intervjuer og informant. Hvis dette blir gjort riktig kan det føre til at informanten får en større forståelse for hva intervjueren er ute etter. På grunn av denne dynamikken kan man gå mer i dybden og man kan håndtere mer komplekse tema.

Men det finnes en rekke problemer med intervjuer som man må være oppmerksom på når man skal lage et forskningsdesign. Det tar tid å gjennomføre intervjuer. I tillegg til selve intervjuet som er på en halv til en time kommer forberedelser og transkribering. Det er

vanskelig å gjennomføre gode intervjuer og det krever erfaring fra den som gjennomfører intervjuene. Man må i tillegg være kritisk til validiteten på svarene man får. Det er et velkjent fenomen at intervjuobjekt svarer i forhold til hva de tror intervjueren vil høre eller hva de selv ønsker at svaret burde være. Man må også se på intervjuet i forhold til kognitive aktiviteter hos mennesker. Det er for eksempel ikke alle detaljer som er like lette å huske i en intervjusituasjon. På grunn av disse problemene er det svært vanlig å supplere intervju med andre kvalitative teknikker. En slik teknikk er observasjon.

3.2.3 Observasjon

Det er ikke bestandig nok å intervjue informanter når man skal gjøre en kvalitativ studie. Det kan være stor forskjell på hva informanter sier de gjorde og hva de faktisk gjorde eller vil gjøre. For å kompensere for dette er det vanlig å benytte seg av observasjon i kvalitative studier.

Det er vanlig å kategorisere ulike observasjonsteknikker i forhold til to motpoler. Disse er deltagende observasjon og strukturert observasjon. Deltagende observasjon er en kvalitativ tilnærming som stammer fra antropologi, mens strukturert observasjon er en kvantitativ tilnærming. Det er vanlig å skille mellom skjult og åpen observasjon.

Åpen og skjult observasjon

Det er vanlig at informantene i en kvalitativ studie er klar over at de blir observert og analysert. Men dette er ikke uproblematisk. Hvordan vet man at de man observerer oppfører seg slik de vil gjør når forskeren ikke er tilstede? I åpen observasjon er det en fare for at aktørene vil ”spille for galleriet”. Fremfor å oppføre seg som normalt vil de gjøre det de tror er forventet av dem og det som er sosialt eller etisk riktig.

For å unngå ”spill for galleriet” kan man gjennomføre observasjonene i skjul. Forskeren kan delta i en gruppe uten å informere om sin rolle som forsker. Forskeren kan også observere gruppen fra utsiden. Det er derimot tvilsomt om skjult observasjon er etisk forsvarlig. I de fleste tilfellene vil det være uforsvarlig å forske på noen som ikke har gitt sitt samtykke til dette.

En annen tilnærming for å unngå ”spill for galleriet” er å eksponere informantene for observatører over så lang tid at de blir vant til deres tilstedeværelse. Heller ikke denne tilnærmingen er problemfri. Hvis man observerer en gruppe over lengre tid vil man påvirke denne gruppen. Det kan også være vanskeligere å forbli objektiv hvis man observerer en gruppe over lang tid. Klassisk observasjon slik den er kjent fra antropologi krever ofte at man følger et miljø over svært lang tid, gjerne to til tre år. I moderne forskning er det derimot vanlig med betraktelig kortere tidsrammer, men det er uansett en tidkrevende aktivitet.

3.2.4 Dokumentanalyse

I kvalitative studier er det ikke uvanlig å analysere dokumenter som er produsert av dem man studerer. Slike artefakter kan være møtereferat, brev, spesifikasjoner, kontrakter og reklamemateriell. Dokumentanalyse bruks mest som et supplement til andre metoder. En fordel med dokumentanalyse er at det ikke er like tidkrevende som intervju og observasjon.

3.2.5 Etikk

Det er viktig å være bevisst på de etiske utfordringene når man skal gjennomføre en empirisk studie. Åpen og skjult observasjon er et eksempel på en etisk utfordring. Det finnes også mange andre utfordringer. [side 67, Robson 1993] nevner praksis som er etisk tvilsom. Noen av disse er:

- Å involvere deltagere uten at de er klar over det eller har godkjent sin deltagelse.
- Å holde tilbake informasjon knyttet til prosjektets sanne natur.
- Eksponere deltagere for fysisk eller mentalt stress.

3.2.6 Validitet

Uansett hvilke metoder som blir valgt, bør forskeren vurdere gyldigheten og påliteligheten til forskningen som er gjennomført. Hvordan kan vi vurdere gyldigheten og påliteligheten til forskningsarbeid innen informasjonssystemer? En måte å vurdere gyldighet og pålitelighet innen slik forskning er utviklet av Klein og Myers. I artikkelen ”A set of principles for Conducting and Evaluating Interpretive Field Studies in Information systems [1999]” presenterer Klein og Myers syv prinsipper for å utføre og evaluere kvalitet innen fortolkende feltstudier:

Den hermeneutiske sirkel

Ifølge prinsippet om den hermeneutiske sirkel oppnår mennesker forståelse ved å se på både enkeltelementene og helheten som disse former. Dette prinsippet må ses i sammenheng med de andre prinsippene til Klein og Myers.

Kontekstualisering

Det er viktig å reflektere rundt den sosiale og historiske bakgrunnen, for å forstå hvordan den nåværende situasjonen oppsto.

Interaksjon mellom forsker og forskningsobjektene

Det er viktig med refleksjon rundt hvordan forholdet mellom forsker og forskningsobjekt påvirker datainnsamlingen.

Abstraksjon og generalisering

Hvordan kan man relatere forskningsresultatene til teoretiske og generelle konsepter?

Veksling mellom teoretisering og lesing av data
Hva er forskjellen mellom de oppfatningene man hadde da man utviklet forskningsdesignet og de man fikk fra de faktiske funnene? Hadde forskeren forutinntatte holdninger? I hvor stor grad passer funnene? Hvor lydhør var man for å endre oppfatning?

Flere fortolkninger
Ut fra et sett med data er det mulig å gjøre ulike tolkninger. Det øker kvaliteten å synliggjøre at ulike tolkninger finnes.

Mistenkeliggjøring
Det er viktig å være mistenkelig overfor mulig fordreining av virkeligheten. Forskningsobjektet kan ha motiver for å skjule eller gi falske opplysninger. Slike fordreininger kan skje ubevisst. I tillegg til å forstå selve dataene må forskeren kunne lese og tolke det som skjuler seg bak de uttalte ordene.

Jeg vil i kapittel 8 bruke disse prinsippene for å evaluere gjennomføringen av case-studiet.

3.3 Metodevalg

Det er mye å ta hensyn til hvis man skal finne et godt forskningsdesign. Ulike metoder har ulike egenskaper, og man må prøve å veie de ulike styrkene og svakhetene mot hverandre. Jeg vil her se på mine forskningsspørsmål og på bakgrunn av disse komme fram til et forskningsdesign.

3.3.1 Forskningsdesign

Jeg har tre forskningsspørsmål og tre forskjellige tilnærminger til å finne svar på disse spørsmålene:

Teori
Mitt første forskningsspørsmål er følgende:

- Hvordan er RUPs tilnærming til systemutvikling sammenlignet med brukersentrerte metoder?

For å svare på dette spørsmålet har jeg tenkt å se nærmere på noen av de mest sentrale RUP tekstene og sammenligne disse med brukersentrert design. Jeg vil her se på flere aspekter knyttet til de ulike tilnærmingene. Jeg vil se på historien bak de ulike tilnærmingene. Hvorfor har de blitt slik de er? Jeg vil i tillegg se på hvilken retorikk de to tilnærmingene bruker for å rettferdiggjøre sin verdi. Jeg vil også se nærmere på forskjellene mellom Use-Case og scenarioer som teknikk. Til slutt vil jeg se på noe av kritikken som er kommet mot RUP fra tilhengere av brukersentrert design.

Erfaring
Mitt andre forskningsspørsmål er følgende:

- Hvilke faktorer påvirker den faktiske anvendelsen av brukersentrert metode i RUP-prosjekter?

For å finne svar på dette spørsmålet har jeg valgt å satse på kvalitativ metode. Dette er ikke et spørsmål som lar seg beskrive av tall og statistikk. Det er mange faktorer som kan påvirke bruken av RUP og brukersentrerte metoder. Dette kan være holdningene til systemutviklerne, holdninger hos kunde, de organisatoriske rammebetingelsene eller kunnskapsnivået om RUP og brukersentrerte metoder.

Det er få studier som ser nærmere på bruken av brukersentrerte designmetoder i RUP. Derfor kan det være hensiktsmessig å gjennomføre en kvalitativ studie av RUP og brukersentrert design i praksis. En slik studie vil forhåpentligvis gi oss en pekepinn på hvor skoen trykker i forhold til dette temaet. Siden jeg ikke helt vet hva jeg leter etter vil jeg ikke benytte meg av

noen kvantitative metoder. Derimot kan svarene vi får fra denne studien være et utgangspunkt for en kvantitativ studie (for eksempel en spørreundersøkelse).

Et systemutviklingsprosjekt kan studeres på mange måter med forskjellige briller. For eksempel, hvilket detaljnivå skal studeres. På den ene siden kan jeg ta for meg et veldig spesifikt tema som Use-Case modeller. Jeg kan studere hvordan Use-Case modellene blir til og hvordan de blir brukt. En interessant ting å se på kan være hvordan Use-Case modeller fungerer i forhold til å kommunisere med sluttbrukere. En annen tilnærming er å prøve å se det store bildet. Hva er prosjektets rammebetingelser, hvem har makten og hvorfor ble prosessen slik den ble? Jeg har i dette studiet valgt å se å på de store linjene.

Jeg har valgt metoder som observasjon, intervju og dokumentanalyse for å finne et svar på mitt forskningsspørsmål. Mitt forskningsspørsmål er veldig åpent og krever derfor en åpen tilnærming. Det kan være mange faktorer som påvirker faktisk bruk av RUP. Jeg har derfor valgt en tilnærming hvor jeg først prøver å få et overblikk over prosjektet. Hovedvirkemiddelet blir observasjon supplert med intervjuer og dokumentanalyse. Mitt hovedfokus er bruken av brukersentrerte metoder i RUP. Jeg er derimot åpen for overraskelser.

Intervjuene
Intervju kan være en nyttig teknikk for å få en rask forståelse av hvordan prosjektet fungerer. Tidlige intervju kan gi en pekepinn på hvilke aktiviteter jeg bør observere og legger føringer for utformingen av mitt forskningsdesign. Intervju kan gi et innblikk i hva som har skjedd i prosjektet frem til nå. Intervju gir meg også en mulighet til å se hvilke holdninger prosjektgruppen har til brukersentrert design og hvilke holdninger som eksisterer i bedriften for øvrig. Hvis studiet tar en bestemt retning kan jeg bruke intervju for å utdype et bestemt tema. Jeg vil hovedsakelig benytte meg av semistrukturerte intervju.

Observasjonene
Jeg vil benytte meg av observasjon i tillegg til intervju. Ved hjelp av observasjon vil jeg forhåpentligvis få større innsikt i arbeidsmetodene i prosjektet. Et fokusområde vil være å se på forskjeller mellom den offisielle versjonen av hvordan de jobber, og de faktiske arbeidsprosessene. Kartlegging av hvilke aktiviteter som blir gjennomført og hva som er resultatet av de ulike aktivitetene er også noe jeg vil se på. Jeg vil i tillegg ha fokus på kommunikasjon. Hvordan kommuniserer systemutviklerne med kunden og brukerne?

I utgangspunktet skal jeg bruke resultatene fra de første intervjuene til å finne hvilke deler av prosjektet jeg skal observere. De mest aktuelle kandidatene er arbeid knyttet til design av brukergrensesnitt (Dette kan være interne møter, møter med kunder og med brukere) og prosjektmøter. Jeg vil følge prosjektmøter fordi de vil gi meg en mulighet til å få et bedre oversiktsbilde over utviklingsprosessen.

Dokumentanalyse
Hvis jeg får tilgang vil jeg se på ulike dokumenter som blir produsert i forbindelse med prosjektet. De viktigste dokumentene som jeg vil se på er artefakter knyttet til design og dokumenter som beskriver eller dokumenterer arbeidsprosessen til gruppen. Artefakter knyttet til systemets design kan for eksempel være ulike UML-modeller eller bruksscenarioer. Artefakter som beskriver eller dokumenterer arbeidsprosessen til gruppen kan for eksempel være prosjektplanen eller møtereferat. Jeg vil i tillegg prøve å få tilgang til prosjektets visjonsdokument, hvis dette er tilgjengelig. Dokumenter som er produsert tidlig i prosjektet kan også gi meg en mulighet til å forstå arbeidet fram til min deltagelse.

Anbefaling
Mitt tredje forskningsspørsmål er følgende:

- Hvordan kan brukersentrerte metoder integreres i RUP?

Basert på funnene i de to andre forskningsspørsmålene skal jeg gi anbefalninger på hvordan brukersentrerte metoder kan integreres i RUP. Jeg vil benytte meg av mine og andre relevante funn. Hvordan dette løses, er avhengig av svarene jeg får på de to første forskningsspørsmålene.

3.3.2 Gjennomføring av case-studie
Via min veileder kom jeg i kontakt med et konsulentfirma som har RUP som primære utviklingsmetode. De har som mål å benytte seg av brukersentrerte designmetoder i sine prosjekt. Etter et møte mellom meg og representanter for dette firmaet ble vi enige om at jeg skulle følge et pågående utviklingsprosjekt som de hadde fått via en anbudskonkurranse. Jeg fikk tilgang til å observere og intervjue prosjektdeltagerne i dette prosjektet. Jeg fikk også tilgang til webbasert prosjektverktøy hvor prosjektdeltagerne delte dokumenter relatert til prosjektet.

Jeg hadde i første omgang fokus på selve arbeidsprosessen i prosjektet. I en slik situasjon kan det være lurt å begynne med og intervjue noen av prosjektdeltagerne slik at man kan få en følelse av hvordan prosjektet er organisert.

For å undersøke arbeidsprosessen observerte jeg prosjektdeltagerne i forskjellige situasjoner. Jeg supplerte dette med å gjøre intervjuer og å lese dokumenter som prosjektdeltagerne har produsert. Mine intervju og observasjoner ble dokumentert ved hjelp av videoopptak og notater.

Prosjektet startet fjerde kvartal 2002. Jeg startet mine observasjoner januar 2003 og fulgte prosjektet fram til Construction-fasen som startet i mai samme år. Tabell 3.1 gir en oversikt over mine observasjoner.

I artikkelen ”A set of principles for Conducting and Evaluating Interpretive Field Studies in Information Systems” presenterer Klein og Myers [1999] syv prinsipper for å utføre og evaluere kvalitet innen fortolkende feltstudier. Jeg vil i kapittel 8.2 se på mit studie i lys av disse prinsippene.

Tid	***Aktivitet***	***Dokumentasjon***
Uke 1	Prosjektmøte	Notater
Uke 1	Intervju med Kari	Notater
Uke 1	Intervju med Susanne	Notater
Uke 1	Brukergrensesnittdesign ”brainstorm”	Notater
Uke 2	Prosjektmøte	Video Notater
Uke 3	Prosjektmøte	Notater
Uke 4	Prosjektmøte	Video Notater
Uke 4	Brukergrensesnittdesign workshop	Notater
Uke 4	Kundemøte	Video Notater
Uke 5	Møte med potensielle brukere	Video Notater
Uke 6	Prosjektmøte	Notater
Uke 7	Prosjektmøte	Video Notater
Uke 8	Prosjektmøte	Video Notater
Uke 8	Intervju med Susanne	Video Notater
Uke 8	Intervju med Kari	Video Notater
Uke 15	Presentasjon av mine observasjoner og mulige prosessforbedringer.	Video

Tabell 3.1: Oversikt over mine observasjoner

4 RUP og brukersentrert design fra et teoretisk perspektiv

Jeg vil i dette kapittelet se nærmere på forholdet mellom RUP og brukersentrert design fra et teoretisk perspektiv. Jeg vil forsøke å belyse ulike aspekter knyttet til de to tilnærmingene. Jeg vil se på historien til de to tilnærmingene. Dette vil forhåpentligvis kunne si noe om hvorfor de har den formen de har i dag. Jeg vil også se på hvilken retorikk de to tilnærmingene bruker for å rettferdiggjøre sin verdi. Hvilke metoder bruker de, og hva er forskjellene og likhetene mellom metoder som tas i bruk? For å belyse dette vil jeg se nærmere på forskjellene mellom Use-Case og scenarioer som teknikk. Til slutt vil jeg se på noe av kritikken som er kommet mot RUP fra tilhengere av brukersentrert design.

4.1 Historien bak

Historien bak ulike utviklingsmetodikker kan være interessant. De kan gi oss innblikk i ulike hendelser og filosofi som har gitt utviklingsmetodikkene den formen de har i dag.

4.1.1 Utviklingen av RUP

Mange forskjellige personer og organisasjoner har bidratt til å gi RUP den formen den har i dag. Utviklingen av RUP har røtter så langt tilbake som slutten av 60-tallet. Tidlige versjoner av det som i dag er RUP ble utviklet i telekommunikasjonsselskapet Ericsson fra slutten av 60-tallet og fram til slutten av 80-tallet. En viktig hendelse i denne perioden var utviklingen av standarden ”Specification and Description Language (SDL)”. Denne standarden var sterkt inspirert av Ericssons tilnærming og ble akseptert av CCITT (internasjonal standardiseringsorganisasjon innen telekommunikasjon) som en standard. SDL inneholdt en rekke elementer som ligner på dagens UML-diagrammer. Blant annet class diagrams, activity diagrams, collaboration diagrams og sequence diagrams.

En av de fremtredene personene på utviklingsmetodikk i Ericsson i denne perioden var Ivar Jacobsen. I 1987 sluttet Ivar Jacobsen i Ericsson og etablerte Objectory. Med sin erfaring fra Ericsson utviklet Jacobsen og de andre ansatte i firmaet utviklingsprosessen Objectory. Prosessen ble produktifisert og markedsført mot organisasjoner som drev med systemutvikling. Metoden ble tatt i bruk av mange forskjellige systemutviklingsmiljø i flere land. Det var under utviklingen av Objectory at Use-Case modeller ble en sentral del i utviklingsmetodikken.

Et annet firma som jobbet med systemutviklingsmetoder var Rational Software Corporation. Det hadde utviklet et sett med metoder som var komplementære til Objectorys tilnærming. Blant annet jobbet de mye med objektorientert design, abstraksjon, gjenbruk og prototyping. I 1995 ble Objectory kjøpt av firmaet Rational Software Corporation. Det nye firmaet fikk navnet Rational. Dette firmaet hadde som hovedoppgave å markedsføre og selge RUP som et produkt. Rational ble igjen kjøpt av IBM i 2003.

Det er først i de siste årene at brukergrensesnitt og brukervennlighet har blitt et tema i utviklingen av RUP. Brukergrensesnitt har lenge vært en egen aktivitet i RUP, men det var først i 2001 at denne aktiviteten ble beskrevet på detaljnivå. I "User Interface Design in the Rational Unified Process" er det beskrevet hvordan denne aktiviteten kan gjennomføres [Kruchten et al. 2001]. Det er i tillegg utviklet en "user experience plug-in" til RUP.

I sin beskrivelse av RUPs historie fremhever Jacobson, Booch og Rumbaugh at man i denne perioden har gjort en rekke verdifulle bidrag til systemutvikling. De legger mest vekt på:

- De har vært med på utvikling av modelleringsspråket UML
- Arkitekturbasert utvikling
- Komponentbasert utvikling
- Inkrementell og iterativ utvikling

4.1.2 Utviklingen av brukersentrert systemutvikling

Historien til brukersentrert design er noe mer kompleks enn historien til RUP. Mens RUP er en tilnærming, så er det et stort antall tilnærminger som går under paraplyen brukersentrert design. Det er derfor vanskeligere å gi et korrekt og helhetlig framstilling til utviklingen av brukersentrert systemutvikling. Ambisjonen her er ikke å gi en korrekt og helhetlig framstilling av historien til brukersentrert systemutvikling, men jeg vil likevel presentere noen av de hendelsene som har påvirket utviklingen.

En faggren sterkt knyttet til brukersentrert design er menneske-maskin-interaksjon. Denne faggrenen startet på 70 tallet og hentet mye inspirasjon fra eksperimentell og kognitiv psykologi. Faggrenen baserte seg mye på laboratoriumbaserte eksperimenter. Et viktig bidrag fra denne perioden var Card, Moran og Newells bok "The psychology of human-computer interaction"[Cart et al. 1983]. Boken gir en rekke analytiske modeller som kan brukes til å si noe om ytelsen til brukerne. Et annet fokusområde i denne perioden var brukertesting.

Mye av bidraget fra denne perioden var laboratoriumbaserte. Å gjøre laboratoriumeksperimenter alene er ikke nødvendigvis nok. IT-system benyttes ofte i komplekse miljø hvor arbeidsoppgavene og brukskonteksten stadig er i endring. Mot slutten av 80 tallet ble det i større grad fokusert på brukskontekst som en viktig faktor i forhold til brukervennlighet. Det ble blant annet argumentert for å hente teknikker fra etnografi. Ved å gjøre etnografiske studier av brukere og deres omgivelser ville man være i stand til å ta mer informerte design avgjørelser.

Andre bidrag til brukersentrert design har kommet fra mange hold. En bidragsyter har vært den Europeiske Union, som har finansiert en rekke prosjekt. Et av resultatene fra disse prosjektene er ISO 13407. Det finnes mange ulike tilnærminger som vi kan kalle

brukersentrert design. Et felles trekk for disse tilnærmingene er fokus på menneskelige aspekter.

4.1.3 Rettferdiggjøring

Både RUP og ulike brukersentrerte metoder (for eksempel ISO 13407) bruker mye tid på å rettferdiggjøre sin eksistens. Argumentene som benyttes har ulik karakter og jeg vil derfor se litt nærmere på dette. Dette har vi allerede sett på i kapittel 1. En av de viktigste begrunnelsene til RUP er ønsket om en solid og repeterbar prosess. Vi kan ikke bare stole på de enkelte ansatte. Vi må ha en prosess som kan produsere kvalitetssystemer på en forutsigbar måte. Klarer vi dette, vil det bety mindre risiko for de som tar i bruk prosessen.

> *"The software problem boils down to the difficulty developers face in pulling together the many strands of a large software undertaking. The software development community needs a controlled way of working. It needs a process that integrates the many facets of software development. It needs a common approach, a process that*
>
> - *Provides guidance to the order of a team's activities.*
> - *Directs the tasks of individual developers and the team as a whole.*
> - *Specifies what artefacts should be developed.*
> - *Offers criteria for monitoring and measuring a project's products and activities "*
>
> [Side 3-4, Jacobson et al. 1999]

Brukersentrerte metoder har en noe annen tilnærming for å rettferdiggjøre sin eksistens. I artikelen "Representing the user: Notes on the disciplinary rhetoric of HCI" ser [Cooper & Bowers 1995] nærmere på hvordan brukersentrert design rettferdiggjøres. Her er brukervennlighet et sentralt begrep. Brukeren står i sentrum og for de fleste aktiviteter knyttet til brukersentrert systemutvikling. Mye av utviklingen kommer fra faggrener som kognitiv psykologi og etnografi. I brukersentrert utvikling ønsker man i større grad å ta menneskelig natur med i betraktningen når man skal lage IT-system. Dette rettferdiggjøres både økonomisk og moralsk. Det argumenteres for at fokus på brukervennlighet gir brukerne bedre arbeidsbetingelser, færre ulykker og at man sparer penger.

Det er slik jeg ser det ingen motsetninger mellom RUP og brukersentrert design i den overordnede filosofien til de to utviklingsmetodikkene. Er det mulig å kombinere det beste fra begge verdener? Stabilitet på den ene siden, og brukervennlighet på den andre. Jeg vil her se nærmere på disse to metodene, for å se om vi finner hindringer til en integrering.

4.2 Organisasjonssyn

Hvis vi ser på artefaktene som blir produsert i et RUP-prosjekt, så ser vi at organisasjoner og arbeidsprosesser blir beskrevet på en abstrakt og maskinaktig måte. Modeller som i RUP har navnene Use-Case models, Buissness object models og Domain models er eksempler på artefakter som beskriver organisasjoner og arbeidsprosesser på en abstrakt og maskinaktig måte. Disse modellene er hentet fra UML. RUP anbefaler at man benytter seg av disse modellene. Disse artefaktene er en viktig del av RUP og de legger sterke føringer på hvordan IT-system blir designet. I disse beskrivelsene er det i liten grad skille mellom mennesker og maskiner. Mennesker er ofte beskrevet som objekter som er en del av systemet på samme måte som programvare og maskinvare. Det er liten grad tatt høyde for skillet mellom hvordan maskiner fungerer og hvordan mennesker oppfører seg. Hvis man ser en organisasjon som en mekanisk enhet hvor arbeid blir utført i forhold til regler og formler, så har man et funksjonalistisk organisasjonssyn. Den funksjonalistiske retningen blir ofte referert til som Taylorisme, etter Frederick Winslow Taylor som i 1911 skrev Principels of Scientific Managment[Taylor 1911]. Det funksjonalistiske organisasjonssynet var et resultat av den industrielle revolusjon. Å si at RUP er Taylorisme vil være å sette det på spissen, men funksjonalistiske trekk finner vi i RUP.

I RUP er det lite fokus på kompleksiteten ved å forstå arbeid. Ifølge[Bansler & Bødker 1993] er det ikke uproblematisk å forstå arbeid. Det er ofte stor forskjell på bedriftens offisielle versjon og de faktiske arbeidsoppgavene. Dette må man ta hensyn til hvis man ønsker å utvikle IT-system som vil fungere i praksis. Å ha et for objektivistisk syn på arbeid kan ifølge Bansler og Bødker ha mange konsekvenser:

- Det er lett å undervurdere ferdighetene til de ansatte. Mange arbeidsoppgaver involverer en stor mengde problemløsning. Dette gjelder også rutinemessige arbeidsoppgaver.

- Den uformelle kommunikasjonen mellom ansatte har en stor påvirkning på hvordan organisasjoner blir drevet. Mange arbeidsprosesser blir drevet av uformell kommunikasjon. Dette bør man ta hensyn til når man skal utvikle IT-system.

- Når man planlegger IT-system må man ta hensyn til muligheten for menneskelige feil. Menneskelige feil kan blant annet føre til lavere produktivitet og ulykker. Ved å ta hensyn til menneskelige aspekter og brukskonteksten under designprosessen kan man lage system som fører til færre menneskelige feil. Man bør i tillegg lage system som gjør det lettere å korrigere / rette opp eventuelle feil som oppstår.

- Makt er skjevt fordelt i organisasjoner og man har ofte flere grupperinger med ulike agendaer. Dette kan påvirke et IT-system. Mange IT-system har feilet fordi de går på

tvers av eksisterende maktstrukturer. Dette er et viktig aspekt som man må ta hensyn til.

I hvilke grad disse punktene gjelder RUP og UML kan diskuteres. Det er avhengig av mange faktorer. Rational mener at RUP er en metaprosess og at prosessen skal tilpasses hvert enkelt prosjekt. Hvilke deler man tar i bruk er avhengig av det aktuelle prosjektet og hvordan man tolker prosessen. Selv om flere RUP kilder[Jacobsen et al. 1999, Kruchten 2000] ikke har fokus på å forstå kompleksiteten i arbeid, så er det ikke umulig å integrere aktiviteter som egnet til å forstå arbeid. Den RUP-vennlige boken "Managing Software Requirements, A Unified Approach"[Leffingwell 2001] beskriver ulike tilnærminger som kan være egnet til dette. På en annen side, så er mange av artefaktene som RUP benytter seg av ikke egnet til å beskrive slike aspekter. Skal man beskrive ulike aspekter knyttet til arbeid, så er ikke abstrakte artefakter som Use-Case modeller nødvendigvis det mest hensiktsmessige. Dette kommer jeg tilbake til i avsnitt 4.4.

I brukersentrert design ser man ikke på arbeid som en mekanisk enhet hvor oppgaver blir utført i forhold til regler og formler. Hovedpoenget i brukersentrert design er å ta høyde for menneskelige faktorer. I brukersentrert design er man opptatt av aspekter som ferdighetene til de ansatte, fysiske og psykiske behov, uformell kommunikasjon og menneskelige feil. For å få større forståelse for disse aspektene, gjennomfører man ulike aktiviteter. Dette kan være aktiviteter som brukermedvirkning, etnografisk studier og brukertester. Kunnskapen man opparbeider seg blir så dokumentert og kommunisert på hensiktsmessig måte.

Som nevnt ovenfor, så er makt skjevt fordelt i organisasjoner og man har ofte flere gruppering med ulik agenda. Dette må man ta hensyn til når man skal utvikle IT-system. Makt og konflikt er sjeldent et tema i brukersentrert design. Fra et organisasjonsmessig perspektiv kan man hevde at dette er en svakhet ved brukersentrert design. Dette skal jeg komme tilbake til i kapittel 6.2.7.

I kapittel 2.4 ser vi på Hirschheim og Klein sin paradigmeinndeling av systemutviklingsmetoder. Hirschheim og Klein kategoriserer utviklingsmetoder i to akser. Disse aksene er objektivisme mot subjektivisme og harmoni mot konflikt. Utviklingsmetoder som benytter seg av metoder fra realfagene for å studere menneskelige anliggender kategoriseres under objektivisme. I den subjektiviske aksen er man i større grad opptatt av sosiale forhold og individets subjektive oppfatning. Verden anses som vankelig å forstå og man tar i bruk metoder som etnografi og iterativ utvikling. I forhold til Hirschheim og Kleins modell vil jeg sette RUP i kategorien objektivisme. Dette skal jeg også komme tilbake til i kapittel 6.2.6.

4.3 Use-Case og scenarioer

Det er utviklet en rekke artefakter både innen RUP og brukersentrert design. Jeg vil her sammenligne de to mest populære, Use-Case og scenarioer.

Use-Case og scenarioer

Mens RUP er Use-Case drevet, så er scenarioer ofte den foretrukne teknikken i brukersentrert design. Disse to teknikkene har flere likhetstrekk, men også noen sentrale forskjeller. Det er ikke mulig å komme fram til noe fasitsvar på forskjellene mellom Use-Case modeller og scenarioer. Blant annet fordi det er vanskelig å vite nøyaktig hvordan en Use-Case modell eller et scenario skal være. En som har kritisert Use-Case modellene er Larry L. Constantine. Han mener at Use-Case som konsept er for dårlig definert, og at de inneholder for mange detaljer[Constantine & Lockwood 2001]. Også innen ulike scenariotekniker er det mange varianter[Carroll & Go 2004]. Noen av egenskapene som de skiller de ulike tilnærmingene er miljøet de skildrer, hensikten, innholdet og formen.

Hensikt

Hva ønsker man å oppnå med modellen? Er det for å informere strategisk planlegging, er det et innspill til de som designer brukergrensesnittet, er det for å kommunisere designløsninger til kunder og brukere, eller er det en måte å dokumentere kravene til systemet? Ulike varianter av Use-Case modeller og scenarioer bidrar til disse og andre oppgaver.

RUP er ifølge Kruchten Use-Case drevet[Kruchten 2000]. Use-Case modeller har derfor en rekke funksjoner i RUP. Use-Case modellene blir til i kravdisiplinen og brukes deretter i alle de ulike disiplinene i RUP. I kravdisiplinen brukes Use-Case modellene blant annet som innspill til brukergrensesnittdesignet, i analyse og designdisiplinen brukes de som bro mellom krav og (teknisk) design, i implementasjonsfasen brukes Use-Case modellene blant annet til å finne områder av applikasjonen som trenger optimalisering, og som grunnlag for testplaner i testdisiplinen. De skal i tillegg fungere som et felles språk for kunder, brukere og systemutviklere.

Ifølge Carrol[Carrol 2000] er hensikten med scenarioer å analysere brukernes oppgaver, forutse framtidig arbeid, lage mockups og prototyper, evaluering av ferdig system, lage brukerveileding og for å begrunne designvalg.

Miljø

Det er vanlig at Use-Case modeller og scenarioer skildrer et eksisterende eller framtidig system. Use-Case modeller skildrer normalt kun det aktuelle systemet og brukerne av systemet. I scenarioer er det også vanlig å skildre brukskonteksten til systemet. Det er i tillegg vanlig å ha mer utfyllende informasjon om systemets brukere i scenarioene.

Innhold
Carroll poengterer at scenarioer er historier, og at de har en setting, aktører og et plot. Use-Case og scenarioer prøver begge å si noe om hvilke aktører et informasjonssystem har, og hvordan disse jobber eller vil jobbe med et informasjonssystem. Den store forskjellen ligger i hva slags informasjon man legger vekt på i forhold til aktørene, funksjonaliteten og brukskonteksten. De vanligste scenario typene som er foreslått innen brukersentrert design legger i større grad vekt på systemets brukskontekst og på hvordan systemet oppfattes av brukeren[Carroll & Go 2004]. Scenarioer tar i større grad sikte på å forklare hvorfor brukerne ønsker å utføre de ulike operasjonene og hvilke eksterne faktorer som spiller inn. Denne typen informasjonen kan være svært nyttig å ha når man skal designe et brukergrensesnitt.

Form
Use-Case modeller er ofte tekstlige beskrivelser eller diagram. Innen UML er det klare retningslinjer for notasjonen til et Use-Case diagram. Use-Case modeller er vanligvis mangfoldige. Det vil si at en Use-Case modell vil kunne inneholde alle mulige kombinasjoner av oppgaver en bruker vil kunne utføre. Scenarioer benytter seg av ulike medieformer som video, lyd, historier og tegninger. Formen varier også, de kan være formelle, semi-formelle og uformelle. Scenarioer er vanligvis ikke mangfoldige. Vi kan se på scenarioer som en instans av en Use-Case modell. Scenaroier er konkrete, mens Use-Case er abstrakte.

De finnes ulike typer Use-Case modeller og det finnes ulike typer scenarioer. På den ene siden har vi de erketypiske Use-Case modellene, disse er formelle beskrivelser av et system og de aktørene (mennesker og andre eksterne aktører) som kommuniserer med systemet. Når man utvikler slike Use-Case modeller har man fokus på aspekter som systemet, arkitekturen og utviklingen. På den andre siden har vi scenarioer som er formet som uformelle historier. De forteller om det framtidige systemet, om brukerne (blant annet mål, holdninger og følelser) av systemet og brukskonteksten rundt. Når man utvikler disse scenarioene har man fokus på aspekter som mennesker, brukervennlighet, følelser og kognisjon. Hvor man skal plassere seg på denne skalaen avhenger av situasjonen man er i og hva man skal utvikle.

Det er bred enighet blant tilhengere av brukersentrerte metoder om at man trenger tilgang til ”soft” informasjon for å lage brukervennlige IT-system. Dette inkluderer blant annet utfyllende informasjon om systemets brukere og brukskontekst. Ved å bruke scenarioer med slik informasjon er det lettere for systemutviklere å ha empati for sluttbrukeren og det er lettere å reflektere rundt ulike bruksscenarioer. Hvis man vil designe brukervennlige system, så vil systemutviklere kunne dra større nytte av scenarioer enn Use-Case modellene.

Et annet aspekt er kommunikasjon med kunde og bruker. Use-Case modeller og andre UML-modeller har blitt kritisert for å være for abstrakte og tekniske[Göransson 2003], noe som fører til at ikke-tekniske personer har vanskelig for å bruke Use-Case modeller som et verktøy. Scenarioer kan hjelpe mot dette fordi de er mer konkrete og har en uformell form.

4.4 Kritikk av RUP

Til nå her jeg sammenlignet noen av egenskapene til RUP og brukersentrert design. I dette avsnittet vil jeg se på noe av kritikken som er kommet mot RUP fra tilhengere av brukersentrert design.

4.4.1 Göransson, Lif, Gulliksen

Bengt Göransson, Magnus Lif og Jan Gulliksen er vitenskaplig ansatte ved institutt for informasjonsteknologi ved Uppsala Universitet i Sverige. De har deltatt i en rekke utviklingsprosjekt hvor RUP har vært den valgte utviklingsmetodikken. De anerkjenner at RUP har mye å bidra med i forhold til programvare utvikling, og at RUP har blitt en de facto standard for programvareutvikling i Sverige. Men de har observert en rekke problemer i forhold til utvikling av brukervennlige IT-system[Göransson et al. 2003].

Hva er feil

Göransson, Lif og Guliksen fulgte flere RUP prosjekt som hadde som intensjon å integrere brukersentrert design. De rapporterte følgende problem:

- Manglende fokus på brukersentrert design gjennom hele prosjekt. Kortsiktige mål.
- De som jobbet med brukersentrert design ble oversett.
- Det var ofte for stort fokus på Use-Case modeller. Noe de kaller ”Use-Case mani”.
- Brukere har problemer med å forstå UML.
- Dårlige holdninger i forhold til brukersentrert design.

Dette var problemer i bestemte prosjekt, Göransson, Lif og Guliksen hevder videre at RUP har en utforming som hindrer brukersentrert design.

- For stort fokus på Use-Case modeller og systemarkitektur kommer i veien for å være brukersentrert.
- Iterativ utvikling i RUP er ikke det samme som iterativ utvikling i brukersentrert design.
- Aktiviteter relatert til brukersentrert utvikling skjer kun i kravhåndteringsdisiplinen.
- For stort fokus på artefakter. De hevder at man risikerer å ikke se det store bildet hvis man er for opptatt med å utvikle ulike type artefakter. RUP er i for stor grad en

papirmølle hvor de ulike prosjektdeltagerne er mest opptatt av å fullføre sine dokumenter framfor å samarbeide med de andre medlemmene.

- Ingen støtte for grensesnittdesign. Use-Case modellene er det viktigste innpillet til brukergrensesnittdesign i RUP. Dette er ifølge Göransson, Lif og Guliksen ikke nok. Use-Case og andre UML varianter skiller ikke sterkt nok mellom mennesker og andre systemelementer. Mennesker blir ofte beskrevet som en del av maskineriet.

Göransson, Lif og Guliksen kommenterer ulike tilnærminger fra Rational for å gjøre RUP mer brukersentrert. Disse er:

- User Experience plug-in: Rational har en egen "User Experience plug-in" som kan benyttes i RUP-prosjekter. Denne fokuserer primært på web-utvikling. Göransson, Lif og Guliksen hevder at denne fokuserer på kreativ design og foretningsmessige aspekter, men ikke på brukersentrert design.

- Usability Engineering Roadmap og User-Centered Design Concept Paper: Rational har også gjort tilgjengelig to dokumenter som viser hvordan RUP adresserer brukersentrert design. Disse dokumentene legger ikke til noen nye metoder, roller eller artefakter. Göransson, Lif og Guliksen hevder de nåværende metodene ikke er tilstrekkelige og at man må legge til nye metoder, roller og artefakter før man kan si at prosessen er brukersentrert.

- User Interface Design i RUP: Göransson, Lif og Gulliksen stiller seg skeptiske til RUPs prosess for brukergrensesnitt design(se kapittel 2.1.4). De mener at brukergrensesnittdesign er en kreativ prosess, som ikke kan beskrives som en steg-for-steg prosedyre. Mer konkret hevder de også at det vil være uhensiktsmessig å følge prosedyren steg for sted, fordi det vil føre til et skjermbilde per Use-Case. Noe som igjen vil føre til fragmenterte systemer.

Göransson, Lif og Guliksen ser i tillegg nærmere på problemer knyttet til bruk av Use-Case modeller:

- Vanskelig for brukere å forstå: Use-Case modeller blir for abstrakte og det er derfor vanskelig å se hvordan de relaterer seg til brukerens arbeidsoppgaver og brukergrensesnittet.
- Ulike behov: Göransson, Lif og Guliksen hevder derimot at Use-Case modeller egner seg bra til kommunikasjon mellom designerer og utviklere. Men også her er det problemer, designere og utviklere har ofte ulike behov i forhold til Use-Case diagrammenes størrelse. Utviklere foretrekker ofte små Use-Case modeller når de spesifiserer funksjonalitet. Designere derimot trenger ofte store Use-Case modeller som står til brukenes faktiske arbeidsoppgaver.

- Ikke nok relevant informasjon for design: Use-Case alene er ikke tilstrekkelig som innpill til brukergrensesnittdesign. For å designe et brukergrensesnitt trenger man mye informasjon som man ikke finner i Use-Case diagram.
- Ikke nok fokus på sluttbrukere: Use-Case diagram fokuserer for mye på systemet som en teknisk enhet. De bør i istedenfor fokusere på brukerne av systemet.

Hvordan forbedre

I tillegg til kritikk, kommer Göransson, Lif og Guliksen med forslag til endringer. De innser at RUP er en de facto standard og utrykker at ”RUP—If You Can’t Beat Them, Join Them!” Riktig tilnærming er derfor å finne ut hvordan RUP kan endres for å bedre støtte brukersentrert utvikling.

Göransson, Lif og Guliksen ønsker å utvide RUP med en brukersentrert designdisiplin. Denne kommer i tillegg til de disiplinene som allerede eksisterer i RUP. RUP er i stor grad basert på ”best practices” (se kapittel 1). Den nye disiplinen er basert på tolv ”best practices”. Disse er:

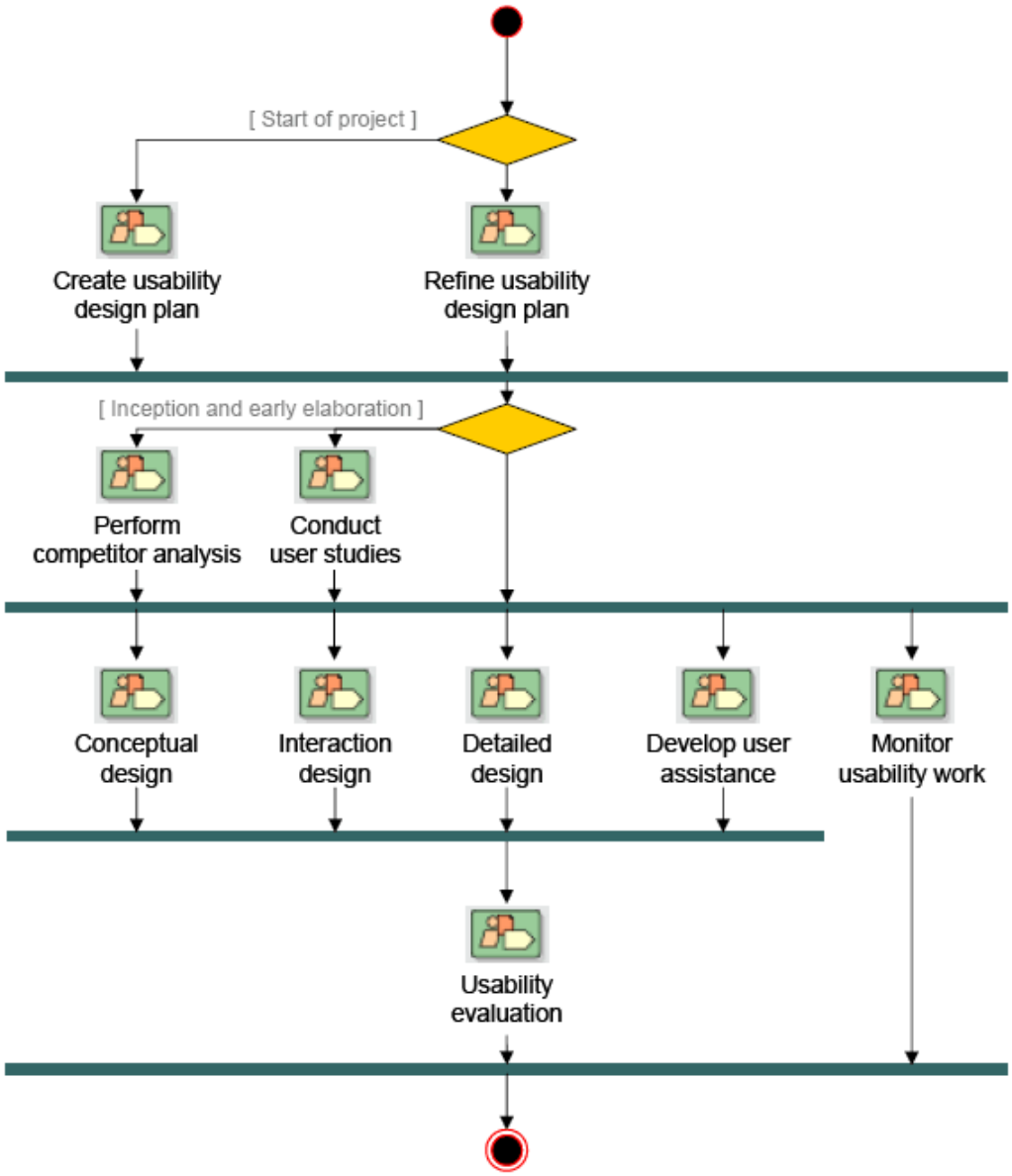
- Fokus på brukere
- Høy grad av brukermedvirkning
- Evolusjoner utvikling
- Enkle designframstillinger
- Prototyping
- Evaluering i forhold til kontekst
- Tydelige og bevisste designaktiviteter
- Profesjonelle holdninger
- Alle prosjekt må ha en brukervennlighetsforkjemper
- Helhetlig design
- Prosessen må kunne tilpasses i hvert tilfelle
- Brukersentrerte holdning i hele organisasjonen

Som en del av denne nye disiplinen foreslår Göransson, Lif og Guliksen nye roller, arbeidsflyt og artefakter.

Roller

De foreslår tilsammen fem nye roller. Disse er:

- Usability designer.
- Field Study Specialist
- Interaction Designer
- Graphic Designer
- Usability Evaluation Specialist.



Figur 4.1 Ny disiplin for brukersentrert design i RUP

Arbeidsflytt
Arbeidsflyten til de nye aktivitetene kan du se i figur 4.1 Arbeidsflyten består av fire deler. Planlegging, undersøkelser, design og testing. Første steg er å lage en plan for hvordan man skal gjennomføre brukersentrert design. Så skal man gjøre studier av brukere og konkurrenter. Videre skal man designe løsningen for så å teste den. Disse stegene skal gjennomføres iterativt.

Artefakter
Det blir foreslått nye artefakter. Disse er brukerprofiler, liste over kvalitative og kvantitative brukervennlighetsmål, beskrivelse av brukernes nåværende oppgaver og oppgavene slik de er når systemet har blitt tatt i bruk, konseptuelle design skisser og en beskrivelse av systemets brukskontekst.

4.4.2 Dave Cronin
Dave Cronin er ansatt i konsulentfirmaet Cooper. Firmaet er veldig kjent i det brukersentrerte designmiljøet. Mest på grunn av grunder Alan Cooper som er kjent som en ”guru” innen design av brukergrensesnitt. Gjennom firmaets nyhetsbrev publiserte Cronin en kritikk av RUP[Cronin 2003].

Cronin er tilhenger av metodikken som er utviklet av de ansatte ved Cooper. Denne kalles Goal Directed Design (GDD). Cronin hevder at GDD og RUP ikke nødvendigvis er i konflikt, men at man må prøve å finne det beste fra begge verdener. Han hevder at RUPs styrke ligger i dens evnen til å håndtere risiko under konstruksjonen (programmering) i store og komplekse prosjekt. Han er derimot ikke like positiv til RUP i forhold til kravhåndtering og brukergrensesnittdesign. Hva er det da som er feil med RUP? Cronin hevder at stadige endringer i krav er et problem i utvikling av IT-system. Kruchten skriver også om dette[Kruchten 2000]. Begge refererer til en større studie gjennomført av Casper Jones [Caspers 1995] hvor det hevdes at 40 % av krav oppstår etter at utvikling har begynt.

Hvordan løses man dette problemet? Cronin sier følgende:

> *“The way RUP (and other “agile” development philosophies) deals with this disturbing fact is to accept it (however unfortunate it may be) and to optimize development practices around managing and responding to change. This is an understandable approach, and makes a lot of sense when you think about what the development team has to work with, but I also think the attitude is defeatist. It is little different than throwing out your roadmaps before your cross-country trip because there’s really no telling how many flat tires, rain storms, and rude drivers you may encounter along the way.”*

Cronin ønsker en mer proaktiv tilnærming til problemet og kaller RUPs tilnærming ”bottom up”. Basert på egen erfaring hevder Cronin at det fult mulig å finne en stor del av kravene før man begynner å programmere. Deres metode baserer seg sterkt på kvalitative undersøkelser. Disse gjennomføres før design, og er viktige i forhold til utformingen av produktet. Cronin poengterer at det er forskjell på høynivåkrav og lavnivåkrav. Høynivåkravene kartlegges tidlig i prosjektet ved hjelp av blant annet kvalitative undersøkelser. Lavnivåkravene kartlegges blant annet ved hjelp av iterativ utvikling. Cronin hevder at man i GDD i mindre grad enn RUP jobber iterativt ved hjelp av prototyper. Han anerkjenner at prototyper har en verdi, men poengterer samtidig at det er problemer knyttet til bruken av dem. Prototyper er ifølge Cronin dyre. Det koster å programmere og han mener at man kan finne mange av de samme feilene ved hjelp av papirprototyper. Han mener at prototyper fører til økt treghet i designprosessen siden det tar tid å programmere disse. Man bør derfor unngå å programmere prototyper tidlig i utviklingsprosessen.

Cronin ønsker å bytte ut Use-Case og aktører med scenarioer og personas. GDD benytter seg i større grad av artefakter som er mer konkrete og har en uformell form framfor de abstrakte og tekniske Use-Case diagrammene (se kapittel 4.3.).

4.5 Oppsummering

Mitt første forskningsspørsmål i denne oppgaven er: Hvordan er RUPs tilnærming til systemutvikling sammenlignet med brukersentrerte metoder?

Når vi sammenligner RUP og brukersentrert design, så finner vi mange forskjeller på mange ulike nivå. Blant annet har de to metodene ulik bakgrunnshistorie. Mens RUP har blitt utviklet i et teknologisk orientert miljø, så har brukersentrert design hentet inspirasjon fra faggrener som kognitiv psykologi og antropologi. De bruker ulik retorikk for å rettferdiggjøre sin eksistens. RUP markedsfører ønsket om en solid og repeterbar prosess. I brukersentrert design er det derimot brukeren som står i sentrum. Her er brukervennlighet et sentralt begrep.

Göransson, Lif og Guliksen etterlyser flere brukersentrerte metoder i RUP. De hevder at verken RUP eller de ulike utvidelsene til RUP inneholder de metodene som er nødvendige for å kalle RUP en brukersentrert utviklingsmetodikk. RUP og brukersenteret design representerer to ulike paradigmer innen utviklingen av IT-system. RUP har sitt utspring i et teknisk orientert miljø og har sin styrke i metodens evne til å håndtere de tekniske utfordringene man finner i store utviklingsprosjekt. Brukersenteret design har derimot et fokus på menneskelige aspekter i forhold til teknologi og er derfor mer egnet til å utvikle brukervennlige løsninger.

Kan man så kombinere det beste fra to verdener. Både Göransson, Lif og Guliksen og Dave Cronin mener at dette er mulig. De poengterer at det finnes ulike hindringer, men er positive i forhold til mulighetene for en integrasjon. Prosessene er ulike på mange områder og man trenger derfor å gjøre strukturelle endringer. Blant annet kan ikke Use-Case ha en så sentral rolle.

Min sammenligning av RUP og brukersentrert design viser noen av forskjellene mellom metodene. For å få et klarere bilde av utfordringene knyttet til å integrere RUP og brukersentrerte metoder, har jeg gjennomført en case studie av et utviklingsprosjekt. Dette prosjektet hadde som mål å kombinere bruken av RUP og brukersentrerte metoder. Jeg vil i neste kapittel se nærme på dette prosjektet.

5 Empiri

5.1 Mitt fokus

Jeg vil i dette kapittelet forsøke å gi en oversikt over hvordan prosjektet var organisert. Jeg vil fokusere på de aktivitetene som bidro til brukergrensesnitt-designet. Jeg har hovedsakelig fulgt Elaboration fasen i dette prosjektet og alle observasjoner ble gjennomført i denne fasen. Jeg har vært observerende deltager på en rekke møter og workshops, og i tillegg har jeg intervjuet noen av prosjektdeltagerne. Jeg har også sett nærmere på en rekke av artefaktene som ble produsert i forbindelse med prosjektet. Jeg vil ikke referere til noe litteratur i dette kapittelet. Derimot vil jeg forsøke å gi best mulig innblikk hvordan dette prosjektet fungerte, hvem gjorde hva, hvordan gjennomførte de arbeidet og hvilke problem hadde de. Casestudiet er anonymisert og alle navn er oppdiktet.

5.2 Kontekst

Cairo er et konsulentfirma med cirka 60 ansatte. De viktigste aktivitetene er utvikling og integrasjon av informasjons systemer. RUP er valgt som utviklingsmetodikk i firmaet og Cairo ser på seg selv som et ledende kompetansemiljø på RUP. Cairo har som ambisjon å bruke RUP i alle store systemutviklingsprosjekt.

Cairo fikk det omtalte prosjektet via en anbudskonkurranse sammen med to andre firma. Disse var underleverandører for Cairo. Disse firmaene stilte med spesialkompetanse som Cairo ikke hadde, i tillegg kom prosjektlederen fra et av disse firmaene. Alle prosjektdeltagere jobbet heltid med prosjektet med unntak av prosjektlederen som hadde 20 prosents stilling i prosjektet. Alle prosjektdeltagerne jobbet i samme kontorlokale.

Kunden er et statlig direktorat. Systemet som skulle utvikles hadde et stort antall potensielle brukere både i og utenfor direktoratet. De viktigste brukerne er direktoratet selv og fylkeskommunene, men det finnes også en rekke andre brukere av systemet. Jeg kan dessverre ikke nevne disse på grunn av anonymitet i forhold til prosjektet. Systemet måtte i tillegg integreres med andre informasjonssystemer både i og utenfor direktoratet. Systemet ble implementert som en webapplikasjon. Jeg vil ikke gi flere detaljer i forhold til systemet på grunn av anonymitet. Selve systemet er ikke så viktig i denne case studien. Jeg har her et fokus på selve utviklingsprosessen. Jeg ble først involvert i dette prosjektet etter at Inception fasen var ferdig. Men via intervju og ved å studere artefaktene som ble produsert i denne fasen har jeg likevel et utgangspunkt for å fortelle litt om hva som skjedde i Inception. Jeg avsluttet mine observasjoner etter Elaboration fasen i prosjektet, og har derfor ikke så mye data om det som skjedde etter Elaboration fasen.

Hva som er RUPs definisjon av et begrep og prosjektdeltagernes definisjon av et begrep, trenger ikke være det samme. De begrepene som omtales i dette kapittelet er hentet fra

datamaterialet til studiet. For eksempel: Når jeg omtaler Inception fasen i prosjektet, så er det perioden som prosjektdeltagerne definerer som Inception.

5.3 Prosjektet og deltagerne

5.3.1 Prosjektet

Dette var et relativt stort prosjekt og var estimert til å vare i overkant av et år. Prosjektet utgjorde cirka 7 årsverk hos leverandøren.

5.3.2 Utgangspunktet

Kunden hadde før Cairo ble involvert gjort en del forarbeid. Resultatet av dette arbeidet var en kravspesifikasjon og noen enkle UML-modeller. Kunden ønsket at disse dokumentene skulle være utgangspunktet for alt videre arbeid. Denne kravspesifikasjonen var viktig for prosjektets videre framdrift og jeg vil derfor i det følgende beskrive hvordan denne var bygd opp.

Kravspesifikasjonen

Kravspesifikasjonen er delt i fem deler. Disse er introduksjon, funksjonelle krav, ikke-funksjonelle krav, krav til gjennomføring og hva som skal leveres.

Introduksjon

Introduksjonen er delt i to deler. Den første delen er en introduksjon til selve dokumentet og inneholder hensikten med dokumentet, leserveiledning, målgruppe, definisjoner og referanser. Om hensikten med dokumentet står det følgende:

> *"Hensikten med denne kravspesifikasjonen er å beskrive kundens krav til leverandør på en slik måte at aktuelle tilbydere kan forstå omfanget av de etterspurte tjenestene i forbindelse med utviklingen av systemet. Videre er det fra kunden ønskelig at denne kravspesifikasjonen brukes som utgangspunkt for håndtering av detaljering og endringer iht. krav i prosjektet, slik at den kan brukes som grunnlag for akseptansetest."*

Funksjonelle krav
Dokumentet har 60 funksjonelle krav, disse kravene er beskrevet med en kort tekst. I tillegg er det for hvert krav et estimat på hvor ofte denne funksjonaliteten vil bli brukt og hvem som har tilgang til denne funksjonaliteten. Et av kravene ser slik ut (anonymisert):

> *"6.2.14 Nytt <objectA>*
> *Det skal registreres et eller flere nye X tilknyttet en allerede eksisterende Y. Dersom endringen gjøres av instans uten godkjenningsmyndighet lagres det nye X kun midlertidig, gå videre til hendelse 6.2.16.*
> *Hvem: Alle med skrivetilgang.*
> *Hvor ofte: Ca. 100 ganger årlig." Hentet fra kravspesifikasjonen.*

Denne formen er brukt på alle de funksjonelle kravene i kravspesifikasjonen.

Ikke-funksjonelle krav
De ikke-funksjonelle kravene er hovedsakelig tekniske krav. Disse kravene er delt opp i kategorier med navn som skalerbarhet, sikkerhet og valg av plattform. De ikke-funksjonelle kravene inneholder også noen ikke-tekniske krav. Disse er blant annet et krav til brukergrensesnittet. Kravet har følgende form:

> *"Brukergrensesnittet skal være enkelt å forstå, og ikke være vanskelig å bruke enn vanlig Internett-side. Den skal kreve minimalt med spesialopplæring, kompetanse og maskinvare." Hentet fra kravspesifikasjonen.*

Krav til gjennomføring
Kravspesifikasjonen har en rekke krav til hvordan prosjektet skal gjennomføres. Dette gjelder områder som samarbeidsform, prosjektorganisering og delleveranser. På samarbeidsform skriver de følgende:

> *"Leverandøren skal sørge for at løsningen utvikles i tett samarbeid med prosjektdeltakere fra kunden. Kunden skal stille med deltakere i form av brukere. Disse brukerne skal ikke brukes til å gi innspill og revidere resultat, men delta aktivt i utformingen av løsningen. Kunden skal sørge for at leverandøren har tilgang til brukere i tilstrekkelig antall og med riktig kompetanse." Hentet fra kravspesifikasjonen*

Og på delleveranser skriver de følgende:

> *"Leverandøren skal legge opp en prosjektplan som fokuserer på hyppige leveranser. Kunden ønsker hyppige leveranser for å sikre at løsningen utvikles i samsvar med krav og behov innenfor rammene av kravspesifikasjonen." Hentet fra kravspesifikasjonen*

Det skrives følgende om utviklingsform:

> *"Kunden ønsker en utvikling av systemet basert på iterativ og inkrementell utvikling. En slik tilnærming kan bidra til å sikre en løsning som er tilpasset behovene innenfor rammene skissert i kravspesifikasjonen. Gjennom utvikling, uttesting og evaluering av deler av løsningen, kan en tidlig avdekke avvik mellom løsning og de definerte krav til systemet." Hentet fra kravspesifikasjonen*

Hva som skal leveres
Disse kravene inneholder en beskrivelse av de artefaktene som skal leveres ved slutten av prosjektet. De viktigste er selve applikasjonen, dokumentasjon og opplæringsdokumentasjon.

UML-modeller
Før leverandøren ble involvert hadde kunden lagd noen UML-modeller. De viktigste var noen Use-Case modeller og et ER-diagram.

5.3.3 Prosjektdeltagerne

De som utgjorde kjernen av dette prosjektet var syv ansatte fra leverandøren og tre representanter fra kunden.

Leverandøren

Prosjektlederen
Mikael var prosjektleder og hadde en 20 prosents stilling i prosjektet. Han hadde det overordnede ansvaret for prosjektgjennomføringen. Han hadde i tillegg ansvaret for de ukentlige prosjektmøtene, ukentlige samtaler med alle prosjektdeltagerne, risikohåndtering, finne prosjekt deltagere og å skrive visjons dokumentet i samarbeid med kunden.

Kari - Kravhåndtering
Kari var ansvarlig for kravhåndtering. Hennes ansvar var å opprettholde enighet mellom de forskjellige interessentene til systemet. Viktige arbeidsoppgaver for henne var å kommunisere endringer som har blitt gjort og å motta endringsforlag fra kunden. Hun dokumenterte sitt arbeid i kravspesifikasjonen og Use-Case modeller. Hun var også ansvarlig for planlegging og gjennomføring av forskjellige systemtester.

Susanne - Interaksjonsdesign
Susanne var ansvarlig for systemets grafiske design og systemets interaksjonsdesign. Hennes offisielle tittel i firmaet er interaksjonsdesigner. Susanne mente selv at hun hadde to forskjellige, men relaterte, arbeidsoppgaver. På den ene siden var hun ansvarlig for systemets grafiske profil, altså systemets estetiske utforming, og på den andre siden var hun ansvarlig for systemets interaksjonsdesign. Susanne har bakgrunn som grafisk designer.

Tom – Ansvarlig konstruksjon
Tom var sammen med Mikael den mest erfarne av utviklerne. Han hadde det overordnede ansvaret for systemets konstruksjon, og for systemets tekniske arkitektur. Dette inkluderer aktiviteter som å finne ut hvilken plattform man skal bruke (database, webserver, applikasjonsserver og lignende), hvordan de forskjellige komponentene skal kommunisere, hvordan håndtere sikkerhet og andre tekniske spørsmål.

Daniel - Databasemodellering
Daniel var ansvarlig for å oppdatere databasemodellen. Han var også ansvarlig for å sette opp den fysiske databasen.

Ole - Webteknolog
Ole var ansvarlig for GUI-implementasjon. Han var den i gruppa med mest kunnskap om GUI implementasjon og web teknologi. Det vil si teknologier som HTML, JavaScript og Flash. I Inception og Elaboration fasen jobbet han med å finne passende teknologier for å implementere brukergrensesnittet. For eksempel, skal vi bruke Flash? Hvilke fordeler/ulemper har vi ved å bruke Flash for å implementere brukergrensesnittet? Ole var i tillegg ansvarlig for å utvikle en prototyp av systemet.

Pål – Ekspert
Prosjektet trengte teknologikompetanse på et spesifikt område. Pål jobbet med de delene av system-arkitekturen som berørte dette spesialfeltet.

Kunden
Kunden hadde allerede før prosjektet startet gjort et grundig forarbeid. Noen av de som gjorde dette arbeidet var også involvert i fortsettelsen.

Ethan – Prosjekt leder
Ethan var prosjektleder på kundesiden. Han jobber hos kundens forsknings og utviklings avdeling og han er utviklernes primære kontaktperson. Han påtok seg også en rolle han selv ga navnet brukerambassadør. Ethan mente at han hadde god kunnskap om brukerne og deres behov. Det var derfor ikke nødvendig å involvere brukere i prosjektet. Han kunne prate på vegne av brukerne.

Karl – domeneekspert
Jobbet blant annet med den originale databasemodellen. Hadde mye relevant domenekunnskap.

Lise – System maintenance
Jobbet med vedlikehold av system hos kunden. Jobbet i tillegg med den opprinnelige database modellen.

5.4 Opprinnelig prosjektplan

I forbindelse med Cairos tilbud til kunden ble det laget en prosjektplan. Denne prosjektplanen forholder seg til RUPs faseinndeling og har fire faser hvor hver fase har x antall iterasjoner. Disse fasene er Inception, Elaboration, Construction og Transition. Inception- og Elaboration fasene er planlagt med to iterasjoner hver. Det skulle være flere iterasjoner i Construction, men antallet var ikke planlagt. Det var ikke planlagt noen iterasjoner i Transition-fasen. Det var ifølge prosjektplanen viktig å ha en iterativ og inkrementell prosess, hensikten er å redusere risiko.

Prosjektplanen beskriver iterativ og inkrementell på følgende måte:

> *"I begrepene "iterativ" og "inkrementell" legger vi følgende:*
> *• At prosessen er iterativ betyr at vi leverer flere delresultater underveis*
> *• At prosessen er inkrementell betyr at hvert delresultat utgjør en delmengde av den totale funksjonaliteten i systemet*
> *• Risikoen i prosjektet reduseres ved at løsningen tidlig testes ut. Dette skjer på flere plan: både ifm. integrasjonstest ved del-leveranse og brukertesting.*
>
> *Dette innebærer:*
> *• At det er mulig for kunden å måle progresjonen underveis i prosjektet*
> *• At det er mulig å avsjekke at den leverte funksjonaliteten er i henhold til krav og spesifikasjoner etter hvert som funksjonaliteten leveres." Fra gjennomføringsplanen levert i forbindelse med tilbud.*

Jeg vil i det følgende gi en oversikt over de aktivitetene som tok sted i Inception og Elaboration. Jeg vil her konsentrere meg om helheten. I seksjon 4.5 vil jeg se nærmere på noen utvalgte tema.

5.4.1 Inception

> *" **Inception** (innledningsfasen): Avgrensning, definere ambisjoner, avdekke faktiske behov og krav." Definisjon av Inception hentet fra gjennomføringsplanen til prosjektet.*

De viktigste aktivitetene i Inception fasen slik jeg så det var å skrive visjonsdokumentet, forstå og detaljere kravspesifikasjonen, identifisere brukererroller, skrive Use-Case, utvikle informasjonsarkitektur og å avklare en rekke tekniske momenter. Jeg vil gi en enkel beskrivelse av disse aktivitetene, hvem som deltok og hva de produserte. Da jeg ikke fulgte prosjektet på egen hånd før etter Inception fasen, har jeg lite data på hvordan de kom fram til selve resultatet. Data om hva som skjedde i Inceptionfasen er hentet fra intervju og artefakter som prosjektdeltagerne har produsert. Tabell 5.1 viser aktivitetene fra Inception i GANTT-form. Aktivitetene til Susanne og Kari er mer detaljerte enn aktivitetene til de andre deltagerne. Det er aktivitetene til disse to som er de som er mest relevante for denne oppgaven.

Visjonsdokument
Dette dokumentet ble ansett som viktig fordi det satte sterke føringer for hvordan prosjektet ble gjennomført. Dokumentet skulle sammen med anbudsdokumenter, tilbud og kontrakt være et viktig innspill til planleggingen av selve prosjektet og skulle fryses når Inception var ferdig. Dokumentet skulle også være et bidrag til å få bedre oversikt over prosjektets omfang. Prosjektleder Mikael hadde hovedansvaret for dette dokumentet og gjorde mesteparten av skrivejobben.

Forstå og detaljere kravspesifikasjon
Prosjektdeltagerne brukte store deler av første uke til å sette seg inn i kundens kravspesifikasjon. Kunden hadde på et tidlig tidspunkt i prosjektet hevdet at kravspesifikasjonen var gjennomarbeidet og et bra utgangspunkt for videre arbeid. Utviklerne var uenige i dette, de mente at kravspesifikasjonen var uferdig, og at det var alt for mange åpne spørsmål. Dette kommer jeg tilbake til i avsnitt 5.5.1. Det ble i løpet av Inceptionfasen gjennomført workshops og møter for å fylle disse hullene.

Skrive de viktigste brukstilfellene
Kunden hadde i kravspesifikasjonen identifisert en rekke brukerroller. Med brukerrolle menes ulike roller som har ulik tilgang til systemet basert hva deres interesse er. Ulike brukerroller har for eksempel tilgang til ulik funksjonalitet i systemet. Det ble i begynnelsen av prosjektet satt av en del tid til å jobbe videre med disse rollene, og det ble brukt mye tid på å forstå og forenkle disse. Videre ble det jobbet mye med å finne de brukstilfellene som passet til de forskjellige rollene. Ifølge RUP skal man identifisere 20% av de viktigste brukstilfellene og

skrive disse i Inceptionfasen. Kari hadde hovedansvaret for dette, men de fleste prosjektdeltagerne hadde en eller flere brukstilfeller som de var involvert i.

Informasjonsarkitektur
Parallelt med dette jobbet Susanne og Ole med systemets informasjonsarkitektur. Visjonsdokumentet beskriver informasjonsarkitektur på følgende måte.

> *"Systematisk strukturering av den informasjonen som er tilgjengelig for brukere av systemet, slik at brukergrensesnittet gir enklest mulig tilgang til denne informasjonen" Definisjon av informasjonsarkitektur ifølge visjonsdokumentet.*

Informasjonsarkitekturen ble utformet i en kontinuerlig dialog med representanter fra kunden. Resultatet av dette arbeidet var et diagram som viste de forskjellige komponentene som løsningen besto av og hvordan disse hang sammen.

Grafisk design
Etter at Susanne var ferdig med å utvikle systemets informasjonsarkitektur begynte hun å utforme systemets grafiske design.

Kundemøter
Data som ble brukt til alle disse aktivitetene var kravspesifikasjonen og UML modellene som kunden hadde utviklet før Cairo ble engasjert. I tillegg jobbet de tett mot kundenes representanter i prosjektet. Dette skjedde både via formelle workshops og uformel kommunikasjon.

Annet
Det ble også gjennomført en rekke andre aktiviteter. En del av disse var knyttet til prosjektledelse, for eksempel planlegging og medarbeideroppfølging. Prosjektgruppa begynte allerede i Inception å se en del på de tekniske sidene ved systemet. Dette var aktiviteter som valg av utviklingsplattform og valg av database. Man begynte også å jobbe noe med testplaner og databasemodellering.

	Inception								
	40	41	42	43	44	45	46	47	48
	Iterasjon 1				Iterasjon 2				
Susanne									
-Informasjons arkitektur									
-Grafisk design									
-Kunde møte									
Kari									
-Kravhåndtering									
-Kartlegge brukergrupper/brukstilfeller									
-Skive de viktigste brukstilfeller									
-Kunde møte									
Mikael									
-Prosjektledelse									
-Med. Arb. Oppfølging									
-Visjons dokument									
-Testplan									
-Kunde møte									
Tom									
-Valg av utv. plattform									
-Skriv system arkitektur dokument									
-Kunde møte									
Ole									
-Kunde møte									
Daniel									
-Database modellering									
-Kunde møte									
Pål									
-Teknologi evaluation									
-Kunde møte									

Tabell 5.1: GANTT diagram over aktiviteter i Inception fasen

5.4.2 Elaboration

> ***"Elaboration*** *(utdypingsfasen): Arkitektur, detaljering av krav (brukstilfeller), informasjonsstruktur, grafisk design." Definisjon av Elaboration hentet fra gjennomføringsplanen til prosjektet.*

I Inceptionfasen ble det gjort en del forarbeid, blant annet skrev man visjonsdokument, de viktigste brukstilfellene og gjorde en del planlegging. I Elaboration var fokus på å gjøre ting mer konkret. De viktigste aktivitetene i Elaboration fasen slik jeg så det, var å skrive alle brukstilfellene, grensesnittdesign, en brukergrensesnittprototyp og en arkitekturprototyp. Brukergrensesnittprototypen skulle brukes til å gjennomføre brukertester mot potensielle brukere. Arkitekturprototypen skulle brukes til å verifisere teknologivalgene og finne en passende teknisk arkitektur.

Brukstilfeller (Use-Case)
I Inception ble de viktigste brukstilfellene påbegynt. I løpet av Elaboration skulle de fleste brukstilfellene detaljeres. Alle prosjektmedlemmer var med og skrev brukstilfeller i en aller annen form, men det var Kari som hadde hovedansvaret for, og som var redaktør for, dokumentene som inneholdt alle brukstilfellene. Brukstilfellene ble skrevet med innspill fra kunden, de andre utviklerne, kravspesifikasjonen, databasemodeller og diverse UML modeller som kunden hadde utviklet. Det var ingen brukere involvert i arbeidet med brukstilfellene.

GUI-Design
Parallelt med dette begynte Susanne å designe systemets brukergrensesnitt. I tillegg til de ferdige brukstilfellene brukte Susanne det samme informasjonsgrunnlaget som Kari gjorde. Det vil si innspill fra kunden, de andre utviklerne, kravspesifikasjonen, databasemodeller og diverse UML-modeller som kunden hadde utviklet. Det første hun begynte med var å identifisere de forskjellige skjermbildene og å lage layout for disse. Deretter la hun til nye detaljer gradvis. Susanne brukte mesteparten av sin tid i Elaboration fasen til å jobbe med skjermbildene til systemet. Selv om Susanne hadde hovedansvaret for GUI-design, så var også de andre prosjektmedlemmene engasjerte i denne aktiviteten.

Alle hadde sine meninger om hvordan skjermbildene skulle utformes, og skjermbildedesign var et stadig tema i kontorlokalet. Etter hvert som skjermbildene ble ferdige, ble de presentert for Ethan og de andre prosjektdeltagerne hos kunden. Disse hadde også en rekke meninger om hva som var viktig og det ble brukt mye tid på diskusjoner rundt grensesnittdesign.

Utvikle brukergrensesnitt prototyp
Det var aldri meningen å lage en prototyp av hele systemet da dette ville bli for omfatende. Et av grensesnittene som de hadde størst problem med å designe var avansert søk. Det hadde vært mye fram og tilbake og diskusjoner rundt dette grensesnittet. Det ble derfor besluttet at

dette skulle være hovedfokus i brukergrensesnitt prototypen, sammen med den mest sentrale funksjonaliteten. Formålet med prototypen var å få tilbakemelding på brukervennlighet og opplevd nytteverdi av systemet. Prototypen ble utviklet av Susanne og Ole. Susanne gjorde GUI design og lagde all grafikk til prototypen, mens Ole gjorde selve programmeringen.

Gjennomføre brukertest

Når prototypen var ferdig, ble det gjennomført en brukertest av systemet. Leverandøren hadde en egen usability-lab som ble benyttet. Systemet ble testet på fem mulige sluttbrukere. De fikk en rekke oppgaver som de måte løse mens de ble observert. På bakgrunn av denne testen fant de en rekke brukervennlighetsproblemer. Selv om det ble funnet noen problemer, så var de fornøyd med de overordende prinsippene i grensesnittet. Det var derfor kun nødvendig med mindre justeringer. Kari hadde hovedansvaret for å gjennomføre brukertesten.

Tekniske aspekter

Det ble i denne perioden også jobbet med en rekke tekniske aspekter. Det viktigste var ferdigstillingen av arkitekturprototypen.

	Elaboration															
	49	50	51	52	1	2	3	4	5	6	7	8	9	10	11	12
	Iteration 1								Iteration 2							
Susanne																
-Prototyp design																
-Informasjons arkitektur																
-Grafisk design																
-GUI design																
-Kunde møte																
-Møte med brukere																
Kari																
-Kravhåndtering																
-Use Case diagrammer																
-Skriv testplan																
-Planleging/analyse brukertest																
-Brukertest																
-Kunde møte																
-Møte med brukere																
Mikael																
-Prosjektledelse																
-Med. Arb. Oppfølging																
-Kunde møte																
-Skrive testplan																
Ole																
-Use Case diagramer																
-Teknisk arkitektur																
-Utv. GUI prototyp																
-Kunde møte																
Tom																
-Use Case diagramer																
-System arkitektur dok.																
-Arkitektur prototyp																
-Kunde møte																
Daniel																
-Database modeling																
-Database implementation																
-Kunde møte																
-Arkitektur prototyp																
Pål																
-Teknologi evaluering																
-System arkitektur																
-Kunde møte																
-Use Case diagramer																

Tabell 5.2: GANTT diagram over aktiviteter i Elaboration fasen

5.5 Grensesnitt design

Til nå har jeg prøvd å gi et oversiktsbilde over de aktivitetene som prosjektgruppen gjennomførte i Inception og Elaboration. Her vil jeg se nærmere på noen detaljer og jeg vil fokusere på hvordan grensesnittdesign ble gjennomført. For å gjøre dette temaet mer håndterlig har jeg delt det opp i tre deler:

Tilgang til informasjon
Jeg vil først se nærmere på hvordan prosjektmedlemmene fikk tak i informasjon om hvordan denne løsningen skulle se ut.

Dokumentasjon
Hvordan ble denne informasjonen dokumentert? Det vil si, hvilke artefakter ble produsert, og hva slags utforming hadde de?

Design
Hvordan ble brukergrensesnittet utformet? Hva påvirket utformingen av brukergrensesnittet? Var det kunden, artefaktene eller potensielle brukere?

5.5.1 Tilgang til informasjon

Utgangspunktet for all utvikling var som nevnt kundens opprinnelige kravspesifikasjon og UML-diagrammer. I forbindelse med et intervju jeg gjennomførte, hevdet Kari at kunden hadde stor tro på sin egen kravspesifikasjon. Ifølge Kari hadde de så stor tro på kravspesifikasjonen, at en egen kravhåndterer ikke ville være nødvendig. Cairo valgte likevel å ta med en egen kravhåndterer i prosjektet. Cario definerte dette som en egen rolle i tilbudsdokumentet de ga til kunden i forbindelse med anbudsrunden. Rutiner for endringshåndtering var også en del av tilbudet Cairo ga til kunden. Selv om kunden angivelig var svært fornøyd med sin opprinnelige kravspesifikasjon, så skjønte de at det gjensto en del arbeid før de kunne begynne å programmere løsningen.

Det ble tidlig i Inception avtalt at man skulle gjennomføre en rekke workshops for å gi utviklerne innspill til videre arbeid. Det ble opprettet en intern referansegruppe med representanter fra kunden, og sammen med denne gruppen gjennomførte utviklerne en rekke workshops. De fleste utviklerne deltok i disse workshopene, da de handlet om en rekke tema, både tekniske og funksjonelle. Det var hovedsakelig Kari og Susanne som jobbet med de funksjonelle workshopene. Hensikten med workshopene var blant annet å definere brukergrupper, kartlegge informasjonsstruktur, detaljere de viktigste bruksområdene og å revidere forslag til brukergrensesnittdesign.

Det skulle i tillegg opprettes en ekstern referansegruppe med potensielle brukere av systemet, og i denne referansegruppen skulle det være brukere fra flere forskjellige organisasjoner. I et

møte mellom kunde og leverandør ble det bestemt at denne referansegruppen skulle utsettes til man var ferdig med en prototyp av systemet. Jeg var ikke til stede på møtet, men ifølge utviklerne var det kunden som var den største pådriveren for å avvikle denne referansegruppen. Et av argumentene var at kunden selv hadde ansatte med erfaring fra de miljøene hvor produktet skulle brukes. At det er lettere å kommentere/forholde seg til noe som er konkret, ble også brukt som et argument for å vente til prototypen ble fullført. Følgende tekst er hentet fra referatet til dette møtet.

> *"Opprettelse av referansegrupper ble diskutert. Det vil bli opprettet en intern referansegruppe hos kunden. Ekstern referansegruppe vil ikke bli opprettet før prosjektet har en prototyp til utprøving." fra møtereferat Inception.*

> *"De viktigste brukergruppene i tillegg til Kunden sentralt er XX og YY. Kundens egne ansatte kommer selv fra disse miljøene og kjenner godt til hvilke krav og forventninger som stilles." fra møtereferat Inception.*

Ethan påtok seg i dette møtet mye av ansvaret for å gi utviklerne den informasjonen de trengte for å kunne jobbe videre. Ethan mente han kunne ta på seg rollen som brukerambassadør i dette prosjektet. Utviklerne var skeptiske til dette, men godtok løsningen. Det ble aldri opprettet noen ekstern referansegruppe etter at prototypen var ferdig. Det ble derimot gjennomført en brukertest mot fem potensielle sluttbrukere og to møter med potensielle sluttbrukere.

Selv om utviklerne godtok denne løsningen var de ikke helt fornøyde. Spesielt i Elaboration begynte de å bli bevisste på konsekvensene av denne tilnærmingen. I et intervju hadde Susanne følgende kommentar:

> *"For noe av grensesnittdesignet, kan jeg bruke mine tidligere erfaringer og kunnskap om grensesnittdesign. Men for andre avgjørelser har jeg ganske enkelt ikke nok data. Jeg kjenner ikke til brukerne eller deres arbeidssituasjon." Susanne i intervju under Elaboration fasen*

Tom var frustrert over manglende kunnskap om sluttbrukerne.

> *"Dette er håpløst, hvordan kan vi vite noe om dette uten innspill fra brukere? Vi kan jo ingenting om deres arbeidssituasjon." Tom i en design workshop under Elaboration fasen*

Også andre prosjektmedlemmer var bekymret. Manglende kontakt med sluttbrukere ble oppfattet som et risikoelement. Følgende ble skrevet i et møtereferat.

> *"Risiko-lista ble gjennomgått. En del av risiko-momentene fra forrige gjennomgang er avklart, og kan nedgraderes. Viktigste nye punkt (som også har vært oppe tidligere) er prosjektgruppas bekymring for at* Ethan *snakker på vegne av alle brukere. Selv om vi har stor tillit til* Ethan*, vil brukbarheten av det endelige resultatet i stor grad stå og falle på om* Ethan *har korrekt oppfatning av brukernes behov." Fra møtereferat Elaboration*

Utviklerne følte at de hadde for lite informasjon til å ta de riktige designavgjørelsene. De kunne si noe basert på sin erfaring, men var usikre på om de kunne designe en bra løsning.

Det ble derimot gjennomført to enkle møter med potensielle brukere i slutten av Elaboration fasen. Det første møtet ble gjennomført i Cairos lokaler hvor to potensielle brukere var invitert. Disse to personene var ikke direkte knyttet til kunden, men det var kunden som hadde formidlet kontakt. Kari og Susanne presenterte det de hadde gjort til nå. Dette gjorde de ved å presentere både skjermbilder og UML-diagrammer. Det andre møtet ble gjennomført ute hos de potensielle sluttbrukerne. Kari dro dit for å se litt på hvordan de jobbet og hvordan eksisterende system ble brukt. Slik jeg så det, førte ikke disse møtene til noen nevneverdige endringer i systemets design. Følgende kommentar ble derimot rapportert:

> *"Brukerne stusser litt på vektleggingen av eier. De ser et behov for å kunne registrere X direkte (uten at det tilknyttes en Y). De mener det stort sett er det de gjør hele tida i sitt daglige registreringsarbeid." fra rapport møte med brukere Elaboration*

Selv om dette ble et tema, så ønsket kunden at det opprinnelige kravet skulle stå.

5.5.2 Artefaktene

Jeg har allerede beskrevet kundens opprinnelig kravspesifikasjon og UML-diagram. Jeg vil her beskrive noen av de viktigste artefaktene som ble produsert av prosjektteamet, slik jeg ser det. Disse er Brukstilfellebeskrivelser, databasemodeller, informasjonsarkitektur og brukergrensesnittprototyp.

Brukstilfellebeskrivelser

Brukstilfelledokumentet som sto klart ved slutten av Elaboration fasen inneholdt 21 brukstilfellebeskrivelser. Disse brukstilfellene hadde alle samme oppbygging. Hvert brukstilfelle besto av:

Kort beskrivelse

Hva denne delen av systemet skal bruks til av de forskjellige brukerne.

Aktører

De forksjellige brukergruppene som skal bruke denne delen av systemet. Det skilles for eksempel mellom innsynsbruker som kan se men ikke registrere data og innholdasansvarlig som kan redigere og endre data. I tillegg til disse to finnes det en rekke andre brukertyper/aktører.

Startbetingelse

En eventuell startbetingelse for brukstilfelet.

Normalt hendelsesforløp

Her beskrives det antatt mest sannsynlige hendelsesforløpet for dette brukstilfellet steg for steg. De fleste brukstilfellene består av mellom 4 og 10 steg.

Alternative hendelsesforløp

De fleste brukstilfellene har et eller flere alternative hendelsesforløp.

Spesielle krav/kommentarer

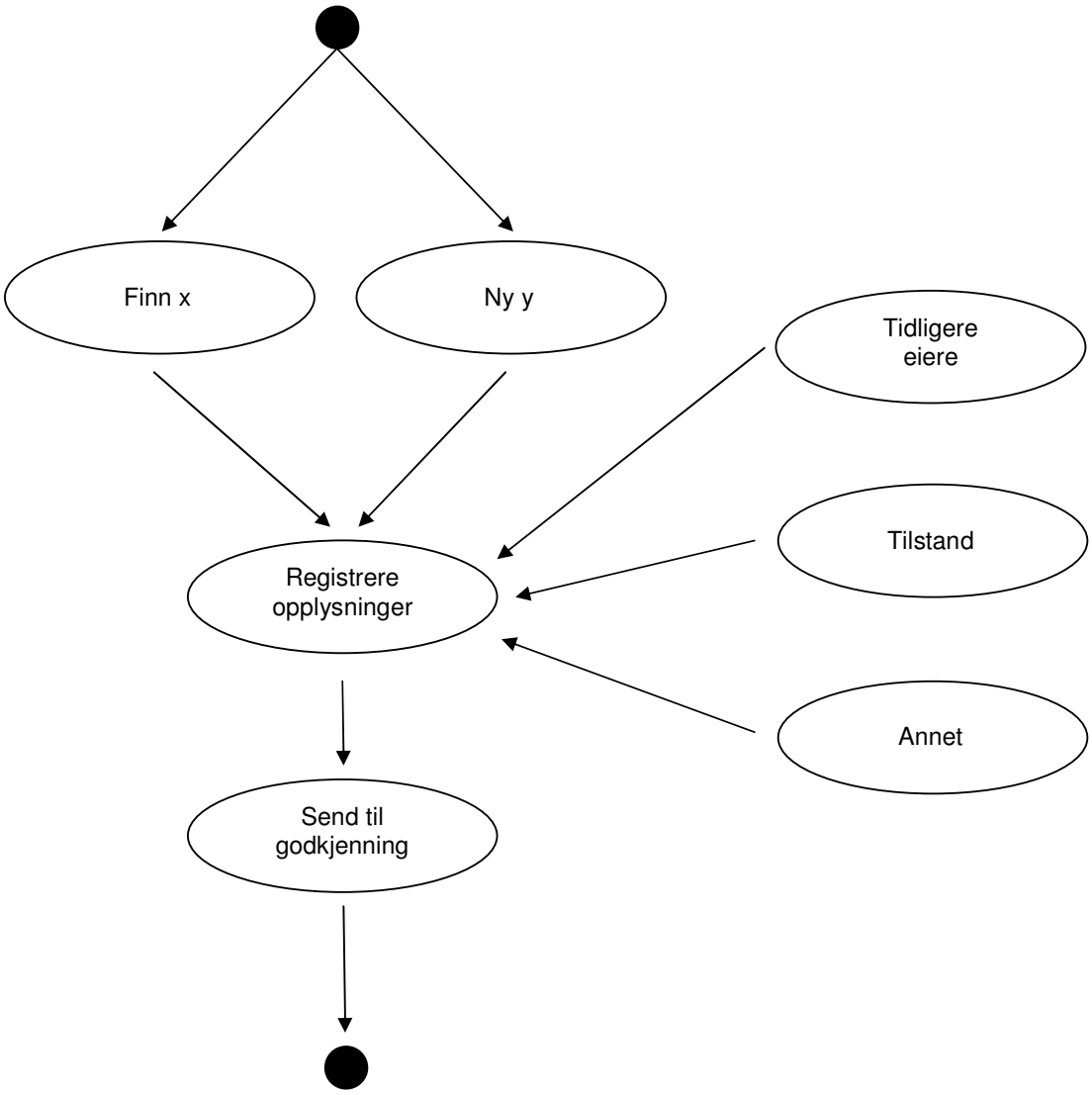
Hvis det er noen spesielle krav eller kommentarer til brukstilfellet.

Henvisninger til kravspesifikasjon

Dette er en liste over de kravene i kravspesifikasjonen som dette brukstilfellet realiserer.

Diagram

Et visuelt diagram over stegene i brukstilfellet. Figur 5.1 er en anonymisert versjon av et Use-Case diagram hentet fra brukstilfelledokumentet.



Figur 5.1 Anonymisert versjon av et Use-Case diagram.

Databasemodeller
Det ble utviklet en vanlig ER-modell. Denne ble brukt av de fleste prosjektdeltagerne. Den ble blant annet brukt av Susanne når hun designet brukergrensesnittet, og den ble brukt av Daniel for å lage den fysiske databasen. Daniel var ansvarlig for å holde databasemodellen oppdatert. Det ble diskutert om man skulle lage to sett med ER-modeller. En fysisk modell og en logisk modell. Dette for å unngå for mange implementasjonsdetaljer i modellen. Man kom fram til at dette ikke var nødvendig og at den var forståelig for alle i gruppen.

Informasjonsarkitektur
Informasjonsarkitekturen ble dokumentert ved hjelp av en enkel sitemap som viste de ulike elementene som systemet besto av.

Brukergrensesnittprototyp
Brukergrensesnittprototypen ble implementert ved hjelp av HTML og flash. Prototypen simulerte avansert søk og de mest sentrale funksjonene i systemet.

5.5.3 Forholdet mellom kunde og leverandør

Forholdet mellom utviklerne og kunden var stort sett bra, men det var ett problemområde. Utviklerne følte at kunden hadde for mange endringsforslag. Dette skapte problemer for utviklerne.

> *"Det er i ferd med å bli et problem at kunden stadig har nye, gode ideer til løsninger. Det er svært positivt med en engasjert kunde, men vi må selv stille krav til tidsfrister dersom diskusjoner med Karl og Lise hindrer oss i å nå egne tidsfrister. Det går nå mer tid enn forutsatt på slike diskusjoner. Endringer i skjermbilder har f.eks. ringvirkninger for design av XML-strukturer, og medfører ekstra arbeid for flere enn Susanne. Dette er i ferd med å føre til forsinkelser for Software Architect Document. Skjermbildene bør nå ikke endres flere ganger før brukertesten gjennomføres. Neste mulighet for kunden til å endre skjermbilder blir i forbindelse med brukertesten." fra møtereferat Elaboration*

Flere av utviklerne var misfornøyde og kundens vingling ble ansett som et risikoelement i prosjektet. I begynnelsen av Elaboration var ikke dette et så stort problem, men kunden fikk stadig flere endringsforslag etter hvert som ting ble mer konkret. Dette førte til at systemutviklerne måtte kjøre en strammere linje i forhold til endinger. I et møtereferat skriver de følgende:

> *"Vi skal heretter være mer påpasselig med endringer, og skille tydelig mellom avtalte løsninger og uforpliktende løsningsforslag. Endringer i avtalte løsninger dokumenteres vha. endringsskjema, dette håndteres av* Mikael *og* Ethan*. Prosjektdeltakerne skal ikke avtale endringer direkte med kunden uten* Mikaels *medvirkning." Møtereferat Elaboration*

Mot slutten av Elaboration hadde ikke leverandøren mulighet til å ta høyde for alle endringsforslagene fra kunden.

> *"Ikke alle endringene som er oppstått som følge av de siste ukers diskusjoner med kunden kommer med. Vi satser på å beskrive leveransen og tydeliggjøre hva som skal korrigeres i neste versjon." Møtereferat Elaboration*

Dette var et anbudsprosjekt, og det er derfor viktig for leverandøren å holde seg innenfor de økonomiske rammebetingelsene som anbudet gir. Hvis kunden ombestemmer seg for ofte, så kan dette føre til mindre overskudd for leverandøren.

5.6 Presentasjon av resultat

Etter at jeg hadde fullført case studiet, fikk jeg muligheten til å presentere mine observasjoner for prosjektdeltagerne hos leverandøren. Jeg presenterte her prosessen slik jeg så den. Jeg presenterte her to hovedproblem:

- Kunden ombestemmer seg for ofte.
- Ikke nok kunnskap om brukerens arbeidsoppgaver / utfordringer.

Prosjektdeltagerne sa seg enige i at dette var de to største ikke-tekniske utfordringene i prosjektet. Som en del av denne presentasjonen kom jeg også med forslag til forbedringer i utviklingsprosessen. Jeg foreslo at de i framtidige prosjekt skulle integrere ISO 13407. Jeg hevdet i tillegg at teknikker som contextual inquiry og personas kunne vært nyttig i et slikt prosjekt. I neste kapittel vil jeg diskutere dette case studiet. Jeg vil prøve å strukturere temaet, blant annet ved å bruke ulike teorier. På bakgrunn av dette vil jeg komme med forslag til forbedringer i kapittel 7.

5.7 Oppsummering av kapittelet

Hovedtrekkene i dette case-studiet, slik jeg ser det, er følgende:

- Systemutviklerne i dette prosjektet hadde et ønske om å utviklete et brukervennlig system og å drive brukersentrert systemutvikling.
- Noen av systemutviklerne hadde også gode kunnskaper om brukersentrerte metoder.
- Likevel ble det i liten grad praktisert brukersentrert design i dette prosjektet.
- Flere av systemutviklerne var frustrerte over manglende kontakt med mulige sluttbrukere. Manglende kontakt med sluttbrukere ble oppfattet som et risikoelement.
- Kunden ønsket ikke å involvere sluttbrukere. Dette var slik de så det, ikke nødvendig. De kom selv fra de aktuelle miljøene, og hadde derfor god kjennskap om krav og forventninger til systemet.
- Selv om systemutviklerne var frustrerte over manglende brukermedvirkning, så ble det ikke gjort nevneverdig forsøk på å overbevise kunden om at dette var viktig.

6 Empirisk diskusjon

Jeg vil her se nærmere på case-studiet presentert i forrige kapittel. Kapittelet er delt opp i tre forskjellige tema. Disse er:

1. Helhetlig vurdering av prosjektet.
Jeg vil prøve å gjøre en helhetlig vurdering av prosjektet. I hvilken grad var dette et RUP prosjekt? Brøt prosjektgruppen med noen av RUP's prinsipper eller fulgte de metodens filosofi til punkt og prikke? Og i hvilken grad var dette brukersentrert design? Mine funn var at prosjektet brøt med flere av prinsippene i både RUP og brukersentrert design. Grunnen til dette, var ikke kunnskapsnivået om RUP og brukersentrert design hos prosjektdeltagerne. Grunnen lå hovedsakelig i forholdet mellom de forskjellige aktørene.

2. Forholdet mellom de forskjellige aktørene
Jeg vil derfor se nærmere på forholdet mellom de ulike partene i dette prosjektet. Maktfordelingen mellom de ulike aktørene, hvordan de kommuniserte og de økonomiske rammebetingelsene påvirket i stor grad prosjektets utforming. Jeg vil i del to se nærmere på disse rammebetingelsene

3. Foreløpig svar på forskningsspørsmål.
På bakgrunn av dette vil jeg prøve å gi foreløpige svar på følgende forskningsspørsmål:

- Empiri: Hvilke faktorer påvirker den faktiske anvendelsen av brukersentrert metode i RUP-prosjekter?

I kapittel sju vil jeg bruke disse erfaringene og konklusjonen fra kapittel fire for å se på måter å integrere brukersentrerte metoder i RUP. Jeg vil i kapittel sju prøve å svare på mitt tredje forskningsspørsmål:

- Anbefalinger: Hvordan kan brukersentrerte metoder integreres i RUP?

6.1 Modenhet

Her vil jeg undersøke egenskapene til dette prosjektet. I hvilken grad var dette et RUP prosjekt, og hvor brukersentrert var utviklingen?

6.1.1 Grad av RUP bruk

Fulgte dette prosjektet RUPs filosofi? For å svare på dette spørsmålet skal jeg se nærmere på RUPs suksesskriterier [side 5, Kruchten 2000] og sammenligne disse med dette prosjektet. Det er ikke alle suksesskriteriene som er like relevante for den oppgaven. Noen omhandler tekniske utfordringer, noe som er utenfor temaet til denne oppgaven. Jeg vil forholde meg til de delene av suksesskriteriene som er knyttet til systemets brukervennlighet. Jeg vil hovedsakelig skrive om Inception- og Elaborationfasene.

Kriterium 1: Programvare skal utvikles iterativt

Den overordnede strukturen i prosjektet er organisert i forhold til prinsippene i RUP. De hadde en Inception og en Elaboration fase. Begge disse fasene hadde to iterasjoner hvor prosjektdeltageren skulle levere ulike RUP-artefakter. Resultatet fra de tre første iterasjonene var hovedsakelig en rekke dokumenter. De viktigste resultatene fra disse iterasjonene var visjonsdokument, Use-Case modellene, informasjonsarkitekturen og grafisk design. Disse dokumentene ble hyppig kommentert av prosjektdeltagerne fra kundens side. Ved slutten av hver iterasjon ble alle dokumentene sendt til kunden for godkjenning. Dette var også arbeidsmetoden i den siste iterasjonen i Elaboration. Her jobbet også utviklerne tett opp mot kunden og fikk stadige tilbakemeldinger. Den siste iterasjonen i Elaboration skiller seg derimot ut ved at det også ble gjennomført en brukertest mot potensielle sluttbrukere. De viktigste artefaktene i denne iterasjonen var: arkitekturprototyp, brukergrensesnittprototyp og brukertestrapporten.

Hovedhensikten med iterativ utvikling i RUP er å kontinuerlig verifisere produktet for å redusere risiko. I dette prosjektet gjøres det hovedsakelig ved hjelp av uformelle samtaler med kunden. Det er ingen brukere involvert i prosjektet, men det blir gjennomført en brukertest mot fem potensielle brukere og to møter med to potensielle brukere. Et annet viktig aspekt var at samarbeidet mellom utviklerne og kunden ikke bestandig var like bra. Utviklerne var misfornøyde med at kunden ombestemte seg så ofte. Ifølge utviklerne hadde kunden alt for mange endringsforslag og de ble oppfattet som vinglete. Utviklerne var bekymret for at prosjektet ville komme til å sprekke hvis kunden ville fortsette å være så vinglete. Dette fordi endringene ofte var tidkrevende og førte til mye ekstraarbeid. På en annen siden kan man jo si at, dette er en del av å drive iterativ utvikling. Man har lov til å prøve og feile for så å komme fram til et bra resultat gjennom å lære over tid.

Utviklerne ble i Elaborationfasen veldig opptatt av å begrense antall endringer i systemet. De var bekymret for at endringer ville føre til budsjettoverskridelser. Dette var noe de hadde tatt

høyde for i kontraktsforhandlingene. Da ble det lagt klare rammer for hvordan endringshåndtering skulle skje. Kan vi da si at dette var iterativ utvikling i henhold til RUP? Det var et tett samarbeid med kunden. Selv om dette samarbeidet ikke var problemfritt, var det i stor grad en iterativ prosess mellom utviklere og kunde. Det ble i tillegg gjennomført en brukertest på fem potensielle sluttbrukere. Men er dette nok for å si at man var iterativ i forhold til RUPs metode? Dette er avhengig av hvordan man tolker RUP. I RUP er man opptatt av at systemet skal verifiseres av systemets interessenter. I dette prosjektet ble systemet verifisert av representanter fra kunden. Dette vil i hovedsak si Ethan, Karl og Lise. Brukere og andre interessenter var i veldig liten grad involvert i prosjektet. Potensielle sluttbrukere var kun involvert ved noen få anledninger (to møter og en brukertest). Som sagt, så er dette avhengig av tolkning. Selv om man legger stor vekt på å verifisere systemet opp mot ulike interessenter, så skriver Kruchten også følgende:

> *"A iterative process lets you take into account changing requirements. The truth is that requirements normally change. Changing requirements and requirements creep have always been primary sources of project trouble, leading to late delivery, missed schedules, unsatisfied customers, and frustrated developers. But by exposing users (or representatives of the user) to an early version of the product, you can ensure a better fit of the product to the task."* [side 77, Kruchten 2000]

Her åpner Kruchten for at man kan involvere en representant istedenfor en sluttbruker. I Delta-prosjektet hadde Ethan en rolle han selv ga navnet brukerambassadør. RUP som beskrevet av [Jacobson et al. 1999, Kruchten 2000] har slik jeg ser det ikke noe klart og tydelig synspunkt på hvor mye og hvordan brukere skal involveres i et utviklingsprosjekt, og hvordan man ser dette i forhold til iterativ utvikling. Brukere var i liten grad en del av den iterative utviklingen i dette prosjektet.

En annen faktor er de økonomiske. Hvordan påvirker ønsket om profitt viljen til iterativ utvikling? Det kan godt være at man ikke kjører så mange iterasjoner som man behøver for å øke overskuddet på prosjektet eller unngå underskudd. I anbudssitvasjoner er man ofte presset på pris, og det er derfor viktig med effektiv drift. Prosjektdeltagerne var i dette prosjektet svært bevisste på å unngå prosjektoverskridelser og det kan ha påvirket graden av iterativ utvikling.

Grad av iterativ utvikling i dette prosjektet er avhengig av hvordan man tolker RUP. Hvor viktig er det å involvere andre interessenter enn kunden og hvordan bør denne involveringen skje? Et annet spørsmål er viktigheten av brukertesting. Hvor viktig er brukertesting og hvilke krav stilles det til gjennomføringen av slike tester? Har er ikke retningslinjene i RUP klare nok til å svare på dette spørsmålet. De er også viktig å forstå at de økonomiske rammene legger føringer for hvordan iterativ utvikling skjer i dette prosjektet.

Kriterium 2: Verifiser kvalitet kontinuerlig
Dette suksesskriteriet henger nøye sammen med ”Programvare skal utvikles iterativt”. Ved hver iterasjon skal men verifisere ulike deler av systemet. Man bør verifisere aspekter som systemets funksjonalitet, brukervennlighet, pålitelighet og ytelse. Kruchten legger vekt på at disse verifikasjonene skal være så objektive som mulig. Som nevnt ble produktet i dette prosjektet verifisert på to måter. Kontinuerlig godkjenning av representanter fra kunden og en brukertest. I hvilken grad Delta-prosjektet fulgte RUPs filosofi på dette suksesskriteriet er nok en gang avhengig av hvordan man tolker RUP. Hvis det er viktig å involvere faktiske brukere, så fulgte ikke prosjektet dette suksesskriteriet. Hvis det derimot er nok å involvere noen som har god kunnskap om brukerne, så kommer dette prosjektet bedre ut.

Kriterium 3: Fokus på kravhåndtering
Hvis man har en iterativ utvikling med kontinuerlig verifisering av kvaliteten, vil man også få en ny forståelse for kravene til systemet. Da vil man få nye krav. Det er ifølge Kruchten viktig å gjøre en grundig jobb i å finne, organisere og dokumentere disse kravene. Cairo hadde i forbindelse med tilbudet utformet detaljerte retningslinjer for kravhåndtering. Disse ble ikke tatt i bruk i prosjektets begynnelse, men etter hvert som kunden begynte å komme med flere endringsforlag ble disse retningslinjene implementert. Slik jeg ser det fulgte prosjektgruppen RUPs metode i forhold til kravhåndtering.

Kriterium 4: Kontroller endringer
I et utviklingsprosjekt er det ofte mange utviklere. I tillegg er de kanskje spredt over flere geografiske lokasjoner, mens de jobber med forskjellige produkter, i forskjellige utgivelser og mot flere plattformer. Da er det viktig med koordinering av disse aktivitetene. Med slik kompleksitet er det ifølge Kruchten viktig å holde oversikt over endringer som blir gjort. I dette prosjektet var alle utviklere plassert på samme kontor og de fleste jobbet heltid eller tilnærmet heltid på prosjektet. Dette suksesskriteriet er derfor ikke så viktig i dette prosjektet. Et unntak var avstanden mellom kunde og utvikler. Leverandøren og kunden holdt til på to ulike plasser. Kari jobbet mye med å holde oversikt over de endringene som ble gjort i prosjektet. Hun brukte i også mye tid på å kommunisere disse endringene til kunden. I tillegg hadde Ethan og Mikael hyppige møter for å holde oversikt over prosjektframdrift.

Kriterium 5: Modeller programvare visuelt
Flere av UML-modellene som RUP anbefaler ble utviklet i dette prosjektet. De viktigste UML-modellene i dette prosjektet var Use-Case-modellene og ER-modellene. Det ble brukt mye tid på å lage disse modellene. Begge disse ble hyppig brukt av Susanne i begynnelsen av prosjektet. Disse modellene ble derimot ikke så hyppig brukt i kundemøter. Her var det hovedsakelig skjermbilder som ble brukt som støttende artefakter. Jeg studerte ikke bruken av disse modellene og kan derfor ikke si noe om de på detalj nivå. Derimot kan jeg si at visuell modellering var et viktig aspekt i dette prosjektet.

Kriterium 6: Bruk komponentbasert arkitektur

Dette suksesskriteriet er teknisk motivert og er derfor ikke viktig i denne oppgaven.

Konklusjon

Slik jeg ser det var dette et prosjekt som i stor grad var inspirert av RUPs filosofi. De var fokuserte på RUPs suksesskriterier og planla i forhold til disse. De fulgte også en del av RUPs anbefalte prosesser og utviklet en del av de artefaktene som er sentrale i RUP. Det var noen viktige faktorer som man kan sette spørsmålstegn ved. Prosjektet kunne vært noe mer iterativt. Det ble kun gjennomført en brukertest og to møter med potensielle brukere. Utover dette ble systemet aldri verifisert ved hjelp fra potensielle brukere. Prosjektet tok ikke i bruk alle de metodene og artefaktene som RUP anbefaler. Men dette er heller ikke nødvendig. RUP oppfordrer utviklingsprosjekt til å velge de metodene som er best egnet i et bestemt prosjekt.

6.1.2 Grad av brukersentrert systemutvikling i prosjektet

Det finnes en rekke metoder for å evaluere hvor moden en organisasjon eller et prosjekt er i forhold til brukersentrert design. Metoden jeg skal bruke i denne oppgaven er ”Usability Maturity Model” [Earthy 1998]. J. Earthy har utviklet en ”Human Centerdness Scale” som består av seks nivåer av modenhet en orgransiasjon kan ha. Se tabell 6.1.

ID	Nivå	Egenskaper
X	Ikke anerkjent	(ingen indikatorer)
A	Anerkjent	Anerkjenner problem - medlemmene i organisasjonen skjønner at det er et problem med brukskvaliteten på systemene de lager. Utfører prosesser - man utfører prosesser for å få informasjon som kan brukes til å gjøre systemet mer brukersentrert.
B	Vurdert	Bevisst på brukskvalitet - være klar over at brukskvalitet er en egenskap ved systemet Brukerfokus - de som jobber med brukerrelaterte elementer i systemet tar i betraktning at et menneske skal bruke det.
C	Realisert	Brukeren er involvert - man henter informasjon fra representative brukere ved hjelp av egnede teknikker gjennom hele livssyklusen Teknologi med menneskelige faktorer - metoder og teknikker med fokus på menneskelige faktorer brukes i eller med brukersentrerte prosesser Menneskekunnskap - kunnskaper om menneskelige faktorer brukes i brukersentrerte prosesser
D	Integrert	Integrasjon - brukersentrerte prosesser er integrert med andre prosesser Forbedring - brukersentrerte prosesser brukes til å forbedre arbeidsprodukter fra andre prosesser Iterasjon - utviklingsprosessens livssyklus er iterativ
E	Institusjonalisert	Brukersentrert ledelse - brukersentrertheten har innflytelse på håndteringen av alle systemenes livssyklusprosesser Brukersentrert organisasjon - brukersentrertheten har innflytelse på organisasjonens holdning

Tabell 6.1. Grader av brukersentrert systemutvikling. Oversettelse av tabellen er hentet fra [Eriksen & Mustaparta 2003]

Jeg studerte kun Delta-prosjektet og vil derfor ikke kunne si noe om modenheten hos noen av organisasjonene som var involverte i prosjektet. Min analyse vil derfor kun gjelde dette prosjektet.

Hvert nivå har mellom to og tre delnivå. Nivå B har to delnivå. Jeg vil plassere dette prosjektet på nivå B.2. Nivå B betegnes ved at utviklerne er bevisste på at brukskvalitet ved systemet, og at de som jobber med brukerrelaterte elementer i systemet tar i betraktning at mennesker skal bruke det. Tabell 6.2 viser kravene til nivå B1 og B2.

ID	Krav
B1.1	Ansatte er gjort oppmerksomme på at brukskvalitet er en attributt til et system og at det er mulig å forbedre denne.
B1.2	Ansatte er gjort oppmerksomme på at brukskvalitet kan oppnås via en rekke brukersentrerte aktiviteter.
B1.3	Ansatte er gjort oppmerksomme på at hele systemet må være brukersentrert. Ikke bare brukergrensesnittet eller den fysiske ergonomien.
B2.1	Ansatte er gjort oppmerksomme på at behovene til sluttbrukerne må tas hensyn til under utvikling og ved brukerstøtte.
B2.2	Ansatte er gjort oppmerksom på at sluttbrukeres ferdigheter, bakgrunn og motivasjon er ulik den til utviklerne og de som har ansvaret for brukerstøtte.

Tabell 6.2. Krav nivå B1 og B2

Utviklerne i dette prosjektet var bevisste på systemets brukskvalitet og ga uttrykk for at det var viktig å tilrettelegge for sluttbrukere. Selv om jeg ikke har detaljkunnskaper om deres holdninger til brukersentrert design, så vil jeg på bakgrunn av min erfaring i prosjektet hevde at de oppfyller kravene til B1 og B2. Dette er derimot ikke nok til å bli plassert på nivå C. Nivå C stiller flere krav som ikke var oppfylt i dette prosjektet. Kravene til nivå C1 kan du se i tabell 6.3.

ID	Krav
C1.1	Nivå C krever at brukere er aktivt involverte i prosjektet og at man henter informasjon fra disse brukerne med egnede teknikker gjennom hele prosjektet.
C1.2	Systemets brukeropplevelse skal testes opp mot målbare krav som er utviklet på grunnlag av data som er hentet fra brukere.
C1.3	Designløsningen skal vises fram til systemets interessenter. De skal også kunne gjennomføre ulike oppgaver ved hjelp av en prototyp eller ved en simulering.
C1.4	Tidlig og kontinuerlig testing skal være er en essensiell del av utviklingsmetodikken.

Tabell 6.3. Krav nivå C1

I nivå C1 er det flere krav som ikke er oppfylt. Den viktigste årsaken til dette er den lave graden av brukerinvolvering i dette prosjektet. Det var noe brukermedvirkning i dette prosjektet, men ikke nok til å tilfredstille kravene i nivå C. I nivå B, er det holdningene og kunnskapen til systemutviklerne som står i fokus. I hvilken grad er de klar over viktigheten av å være brukersentrerte? Det er utviklerne i dette prosjektet klar over. Etter min oppfatning hadde flere av utviklerne høy kunnskap om brukersentrert systemutvikling. Hvorfor praktiserte de ikke brukersentrert utvikling til tross for høyt kunnskapsnivå? Det finnes en rekke rammebetingelser som setter føringer for hvordan utviklingsprosjekt gjennomføres. I dette prosjektet la forholdet mellom de ulike aktørene store føringer for hvordan dette prosjektet ble gjennomført. Jeg vil i de neste avsnittene se nærmere på forholdet mellom de ulike aktørene i dette prosjektet.

6.2 Forholdet mellom de forskjellige aktørene

Mangel på kontakt med sluttbrukere er ikke unikt for dette prosjektet. Lignende problemer knyttet til forholdet mellom bruker og designer er et kjent problem. Jeg vil her se nærmere på årsakene til manglende brukermedvirkning..

6.2.1 Manglende brukermedvirkning

En av de som har sett nærmere på dette er Jonathan Grudin [Grudin 1991, Grudin & Poltrock 1994]. Grudin legger vekt på at ulike kontekster gir ulike utfordringer. Han presenterer tre typiske utviklingskontekster[Grudin 1991]. Disse er: produktutvikling, egenutvikling og prosjekter vunnet på anbud. Grudin hevder at det i alle disse scenarioene er problemer knyttet til brukermedvirking, men at årsakene varierer.

Produktutvikling

En type utviklingsmiljø er bedrifter som lager IT-system som skal selges som hyllevare. Bedriften lager i dette paradigmet et produkt som de ønsker å få høyest mulig marked for til høyeste mulige pris. Grudin rapporterer om rekke barrierer for brukermedvirkning i en slik kontekst:

- *Vanskelig å identifisere brukere:* I en slik kontekst kan det være vanskelig å identifisere brukere. For å identifisere potensielle sluttbrukere må man se på produktets marked. Men strategiske markedsavgjørelser er ikke nødvendigvis tilgjengelig for alle i organisasjonen. Ledelsen i bedriften vil ofte holde kortene tett til brystet for å unngå lekkasjer til konkurrentene.
- *Vanskelig å få tilgang:* Det kan være vanskelig å få tilgang til sluttbrukere. De jobber som oftest i andre organisasjoner, og kan også jobbe i andre land.
- *Mellommenn:* Det er ikke bare de som skal designe systemet som har en inntresse i å kunne noe om brukerne av systemet. Ledelsen, markedsavdelingen og brukerstøtte vil også være interesserte i å ha kontakt med brukere. Representanter fra disse avdelingene kan også ta ansvaret for brukermedvirkningen. Problemet med dette er at deres fokus ikke nødvendigvis stemmer overens med hva som er viktig for å utvikle brukervennlige IT-system. Konflikter mellom markedsavdelingen og utviklingsavdelingen er ikke uvanlig i slike organisasjoner.
- *Korte tidsperspektiv:* Det er ikke uvanlig med korte tidsperspektiv i slike utviklingsmiljø. Tradisjonell markedsteori oppfordrer til hurtig utvikling fordi dette reduserer risiko. Men innen programvareutvikling fører dette til små forbedringer framfor faktisk innovasjon. Noe som igjen fører til mindre brukermedvirkning.

Egenutvikling
Hvis en organisasjon ønsker å utvikle et IT-system på egen hånd, så er dette egenutvikling. I et slikt scenario er det lettere å vite hvem brukeren er siden de jobber i samme organisasjon. Men også i denne konteksten er den er rekke aspekter som kan hindre brukermedvirkning.

- *Avstand til brukere:* Å identifisere brukere er lettere ved egenutvikling enn ved produktutvikling, men det trenger ikke å bety at det er så mye lettere å få tilgang til brukerne. I store organisasjoner kan det være både geografiske og organisasjonsmessige skiller mellom utviklere og brukere.
- *Interessekonflikter:* Ledelsen og utviklerne selv kan ha store interesser i systemet som blir utviklet. Noe som kan føre til interessekonflikter. Også mellom ulike arbeidsgrupper kan det være interessekonflikter. Slike interessekonflikter kan føre til politisk polarisering og gjøre brukermedvirking vanskelig. Det kan i tillegg være inntressekonflikt mellom brukerens politiske rolle og rollen som bruker av systemet.
- *Vanskelig å rekruttere:* Å bidra til et utviklingsprosjekt kan kreve både tid og energi. Enkelte brukere vil få problemer med å sette av nok tid til å kunne bidra på en optimal måte.
- *Manglende ressurser:* Egenutvikling preges ofte av mindre resursser enn hva som er vanlig i andre typer systemutvikling.
- *Urealistiske forventninger:* Grudin poengterer at man må være forsiktig hvis man skal bruke prototyping som metodikk. Hyppig bruk av prototyping kan gi brukerne urealistiske forventninger til sluttproduktet. Prototyping kan i tillegg gi et urealistisk bilde av hvor ferdig systemet er.

Kontraktprosjekter.
Prosjektet fra casestudien i denne oppgaven er et eksempel på et kontraktprosjekt. Grudin hevder at det er her vi finner de sterkeste hindringene til brukermedvirkning. Dette gjelder spesielt anbudsprosjekter med fast pris. Delta-prosjektet er et eksempel på et anbuds prosjekt med fast pris. Eventuelle store endringer i kravspesifikasjonen i dette prosjektet ville føre til reforhandlinger.

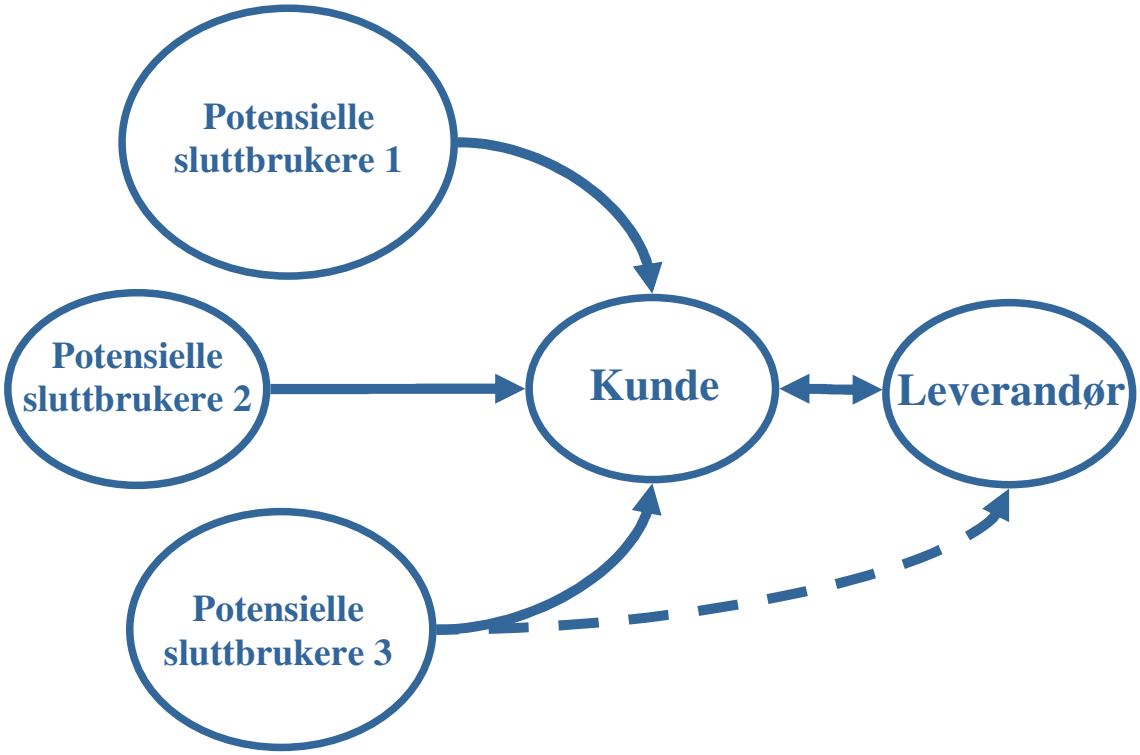
- *Interessekonflikt:* I denne konteksten er det er ikke uvanlig å skille mellom å skrive kravspesifikasjon og utvikling av selve systemet. En slik kravspesifikasjon er nødvendig fordi den gir leverandøren et innblikk i hva som skal utvikles. Den er også nødvendig fordi den gir testbare mål slik at man kan ha en enighet mellom kunde og leverandør om hva som skal utvikles. Å spesifisere krav er derfor ofte et eget prosjekt, med egen anbudsrunde. Da er det ikke uvanlig at firmaet som får i oppdrag å utforme kontrakten i tillegg vil ha utviklingskontrakten Det er her pengene ligger og er derfor mer lukrativt. Dette fører til en interessekonflikt. På den ene siden skal leverandøren designe et best mulig system, på den andre siden skal prosjektet møte en kravspesifikasjon. Å involvere brukere i denne situasjonen utgjør en risko for

leverandøren. Fra et økonomisk perspektiv, så kan det være bedre å forholde seg til en kravspesifikasjon, enn uforutsigbare brukere. Ved å involvere brukere risikerer utviklerne å gjøre prosjektet mer komplekst. Dette er en risiko som kan være vanskelig å akseptere sett fra et økonomisk perspektiv. Å ikke involvere brukere vil gjøre det lettere for et firma som jobber med design og programmering.

- *Prototyping:* Grudin hevder at prototyputvikling ikke er uvanlig i slike prosjekt. Utviklingen av prototypene er ofte kontraktsfestet. Ifølge Grudin prøver leverandørene å legge opp til minimalt med tilbakemeldinger fra brukere for å unngå merarbeid. Dette kan for eksempel gjøres ved å presentere prototypen på mest mulig flatende måte.

6.2.2 Aktørene i Delta-prosjektet

Jeg har valgt å dele de forskjellige aktørene i dette utviklingsprosjektet inn i tre grupper. Disse er leverandøren, kunden og sluttbrukerne. Alle disse befant seg på forskjellige geografiske steder. Leverandøren og kunden hadde kontorer i to forskjellige byer mens framtidige sluttbrukere var spredt over hele landet. De som representerte kunden var også framtidige sluttbrukere av systemet. Leverandøren hadde mye kontakt med representanter for kunden gjennom hele prosjektet. De hadde i tillegg kontakt med potensielle sluttbrukere ved noen få anledninger. Men de mente at de hadde for lite kontakt.



Figur 6.1 Forholdet mellom de ulike aktørene

6.2.3 Gapet mellom designer og bruker

Flere av prosjektmedlemmene i dette prosjektet ga utrykk for at manglende kontakt med sluttbrukere var et problem. Susanne mente det var svært vanskelig å designe brukergrensesnitt med det utgangspunktet hun hadde. Hun følte at hun ikke hadde nok spesifikk informasjon om de virkelige behovene til sluttbrukerne.

> *"For noe av grensesnittdesignet, kan jeg bruke mine tidligere erfaringer og kunnskap om grensesnittdesign. Men for andre avgjørelser har jeg ganske enkelt ikke nok data. Jeg kjenner ikke til brukerne eller deres arbeidssituasjon." Intervju med Susanne i Elaboration*

Også de andre utviklerne så dette som et problem, og flere av prosjektdeltagerne var skeptiske til at kundens prosjektleder Ethan, hadde rolle som brukerambassadør i prosjektet. Selv om prosjektgruppen hadde noe kontakt med potensielle sluttbrukere var det hovedsakelig kunden som la alle føringer på hvordan produktet skulle utformes. Utviklerne mente at Ethans rolle som brukerambassadør var et risikomoment i prosjektet. Følgende notat ble skrevet i et møtereferat midt i Elaboration-fasen.

> *" Viktigste nye punkt (som også har vært oppe tidligere) er prosjektgruppas bekymring for at Ethan snakker på vegne av alle brukere. Selv om vi har stor tillit til Ethan, vil brukbarheten av det endelige resultatet i stor grad stå og falle på om Ethan har korrekt oppfatning av brukernes behov." Møtereferat prosjektmøte i Elaboration*

Den originale planen (i tilbudet) var at prosjektet skulle ha en intern og en ekstern referansegruppe. Den eksterne referansegruppa skulle bestå av potensielle brukere og andre interessenter. Men tidlig i prosjektet ble det bestemt at det ikke skulle bli opprettet en ekstern referansegruppe før prosjektet hadde en prototyp til utprøving. Dette etter kundens ønske. Hvorfor ønsket ikke kunden å involvere brukere i dette prosjektet? Kunden mente de hadde den nødvendige kunnskapen for å designe denne applikasjonen. Kundens representanter i prosjektet hadde selv jobbet i de miljøene hvor IT-systemet skulle brukes, og mente derfor at de har god kunnskap om hvilke krav og forventninger som stilles til systemet. Ethan påtok seg rollen som brukerambassadør i prosjektet. Hadde utviklerne spørsmål om brukere og deres arbeid, så var det han som skulle svare på dette. Utviklerne var ikke fornøyde med dette, men valgte å forholde seg til det. En annen grunn til manglende brukermedvirkning kan også være en konsekvens av at dette er et anbudsprosjekt.

6.2.4 Forholdet mellom kunde og designer

Selv om det var lite kontakt med sluttbrukere i dette prosjektet, så var det noe i slutten av Elaboration fasen. Da designet begynte å ta form på slutten av Elaboration fikk utviklerne kontaktinfo til to framtidige sluttbrukere. Kari gjennomførte to møter med disse brukerne, og Susanne deltok på et av disse møtene. Brukerne hadde flere kommentarer på designet til systemet. Men det var få av disse kommentarene som kom med i designet.

Et eksempel

En innvending de hadde var knyttet til krav 6.1.2 (se kapittel 5.3.2). Systemet har funksjonalitet for å registrere forskjellige typer data. La oss for eksemplets skyld si at dette systemet skal registrere biler og tidligere eiere. Krav 6.1.2 sier da at for hver bil så må du også registrere minst en tidligere eier. Brukerne mente at dette var feil og at det ville skape problemer i registreringsprosessen om dette ble tilfellet. Selv om sluttbrukerne stilte seg svært skeptiske til denne måten å gjøre det ble det ikke dette tatt hensyn til i det endelige designet.

> *"Brukerne stusser litt på vektleggingen av eier. De ser et behov for å kunne registrere X direkte (uten at det tilknyttes en Y). De mener det stort sett er det de gjør hele tida i sitt daglige registreringsarbeid." fra rapport møte med brukere Elaboration*

Kunden lot seg ikke overbevise om at dette ble et problem og ønsket at det opprinnelige kravet sto. Jeg hadde en uformell samtale med Susanne og Ole etter at systemet ble lansert. De mente at et stort antall sluttbrukere hadde klaget på denne koblingen og at dette var noe alle ønsket å endre (også kunden), men dessverre var det for sent å gjøre noe nå.

Her ser vi to problemer som utviklerne hadde i dette prosjektet. På den en siden er det vanskelig å overbevise kunden om at brukermedvirkning er en viktig del av systemutviklingsprosessen. Hvis de derimot får tilgang til brukere, kan det være vanskelig å bruke eventuell data som brukermedvirkning kan gi.

På grunnlag av dette kan man få inntrykk av at kunden i dette tilfellet har sabotert sitt eget prosjekt og at deres prioriteringer ikke er rasjonelle. Er dette tilfellet? Mitt svar på dette spørsmålet er nei. Det finnes flere gode grunner til at et prosjekt blir gjennomført på denne måten.

Men det var ikke bare manglende brukermedvirkning som var et problem i dette prosjektet. Utgangspunktet for prosjektet var en kravspesifikasjon som kunden hadde stor tillit til. Men virkeligheten ble en annen når prosjektet hadde kommet i gang. Ifølge utviklerne hadde kunden alt for mange endringsforslag og de ombestemte seg alt for ofte. De ble oppfattet som vinglete. Utviklerne var bekymret for at prosjektet ville komme til å sprekke hvis kunden ville fortsette å komme med så mange endringer. Prosjektdeltagerne var bekymret for at denne vinglingen kunne gjøre det vanskelig å fullføre prosjektet innenfor de gitte rammene.

Endringer er tidkrevende og fører til mye ekstraarbeid. På en annen siden kan man jo si at det er en del av å drive iterativ utvikling. Man har lov til å prøve og feile for å så komme fram til et bra resultat gjennom å lære over tid. Uavhengig av hva man legger i iterativ utvikling, så vil for mye prøving og feiling på denne måten være vanskelig å gjennomføre i dette prosjektet. Utviklerne følte at de var i ferd med å miste kontrollen. Prosjektet er et resultat av en anbudskonkurranse, noe som la føringer på hvordan prosjektet kunne gjennomføres. I dette prosjektet var det lite rom for en prøv og feil tilnærming fordi marginene i prosjektet var relativt små.

Det var tydelig at leverandøren var forberedt på at dette kunne skje. Man hadde en egen kravhåndterer i prosjektet selv om kunden ikke så behovet for dette. De hadde også utformet klare retningslinjer for hvordan krav skulle håndteres. Noe som er en viktig del av RUPs metodikk. Kravhåndtering var likevel et problem i dette prosjektet. Hvorfor ble det slik?

Et slikt prosjekt krever mye av forholdet mellom kunde og designer. Vi har allerede sett en del på problemene som utviklerne i Cairo hadde. Men et slikt prosjekt er heller ikke lett sett fra kundens perspektiv. Å kjøpe tjenester på dette nivået er ikke nødvendigvis en behagelig opplevelse. En som har sett nærmere på dette er tidligere toppleder i McKinsey & Company David H. Master. Han lister en rekke bekymringer kunder har når de kjøper konsulenttjenester [Master 1997]. Noen av disse er:

> *"I'm worried. By the very fact of suggesting improvements of changes, these people are going to be implying that I haven't been doing it right up till now. Are these people going to be on my side?*
>
> *I'm feeling ignorant, and don't like the feeling. I don't know if I've got a simple problem or a complex one. I'm not sure I can trust them to be honest about that: it's in their interest to convince me it's complex. "* [side 113, Master 1997].

Masters hovedpoeng er at tillit er veldig viktig i forholdet mellom kunde og konsulent. Tillit til at en konsulent ønsker å skape faktiske verdier for kunden og at konsulenten er kompetent til dette. Han hevder at tillit ikke er noe som kommer automatisk, det er noe som man gjør seg fortjent til over tid.

Interaksjonsdesigner Indi Young [Young 2003] hevder at når alt annet er likt, så er det ofte de med mest makt som tar den endelige avgjørelsen på et designproblem. Sett fra designeren perspektiv er problemet å få gjennomslag for de løsningene hun mener er riktige, og ofte er det mye fram og tilbake slik som i Delta-prosjektet. Youngs forslag til designere er å basere sine designforslag på data framfor erfaring. Young hevder at man ved å samle brukerdata på en ryddig metodisk måte, lettere kan tjene respekt og tillit hos kunden og vil lettere kunne ha fornuftige design diskusjoner som man blir enige om.

Den beste løsningen kan være å designere selv må få anledning til å utforske problem domenet. Dette innebærer å jobbe med kunde/markedsavdeling for å forstå fortningsmålene til systemet og å kunne tilbringe mye tid sammen med potensielle brukere for å forstå systemets brukskontekst. Alan Cooper kaller dette å gi designere ”skinn in the game” [Cooper 1999]. Det er viktig for designere å få ”skinn in the game” for å kunne prestere. Men å gi designere ”skinn in the game” er ikke lett. Ofte krever det organisasjonsmessige endringer. I et anbudsprosjekt kan det være nødvendig med strukturelle endringer i forholdet mellom kunde og leverandør.

6.2.5 Forholdet mellom kunde og leverandør

Var kontrakten årsaken til manglende brukermedvirkning og iterativ utvikling i dette prosjektet? Grudins beskrivelse av typiske anbudsprosjekt minner mye om situasjonen i dette prosjektet.

Kunden hadde allerede utviklet en kravspesifikasjon før Cairo ble involvert. Jeg har ikke fått tilgang til de økonomiske betingelsene i kontrakten, men prosjektet hadde en stram økonomisk ramme. Ved store endringsforslag var det nødvendig med reforhandlinger av betingelsene. Kunden var derimot fornøyd med sin opprinnelige kravspesifikasjon og hadde stor tro på at dette skulle gå bra. De ønsket derfor ikke å involvere brukere.

Selv om enkelte av utviklerne ønsket å involvere brukere i prosjektet, la aldri leverandøren noe stort press på kunden for at dette skulle skje. Det kan være mange forklaringer på dette. Cairo var ansvarlig for både design (detaljering av den opprinelige kravspesifikasjon) og implementasjon (programmering) av dette systemet. Dette gjør det kontraktsbasert på fastpris. I så måte fikk de en interessant dobbeltrolle. På den ene siden skal de designe et bra produkt. Noe som etter ISO 13407 innebærer blant annet iterativ utvikling og brukermedvirkning. På den andre siden skal de tjene penger og gå med overskudd. Iterativ utvikling koster penger og i et fastprisprosjekt blir antall iterasjoner fort begrenset av hvor mye penger som er tilgjengelig. I en anbudssituasjon blir ofte leverandører presset på pris. Brukermedvirkning fører også til økt risiko for leverandøren. For leverandøren er det på kort sikt det mest behagelige å forholde seg til en spesifikasjon. Det er mye lettere å lage et system som er i henhold til en kravspesifikasjon enn å måtte forholde seg til brukere.

På en side kan vi si at de økonomiske rammebetingelsene i dette prosjektet hindret brukersentrert design, men manglende fokus på brukersentrert design kan også ha andre forklaringer. Kunnskapsnivået og holdningen til prosjektdeltagerne har en avgjørende rolle. Selv om flere av prosjektdeltageren hadde et bevisst forhold til brukersentrert design, så var det ikke alle som hadde det. Kunden hadde ikke et bevisst forhold til brukersentrert design. Å gjennomføre brukersentrert design er ressurskrevende. Det er derfor viktig å ha de andre prosjektdeltagerne med seg. Dette er en utfordring i kundeprosjekter.

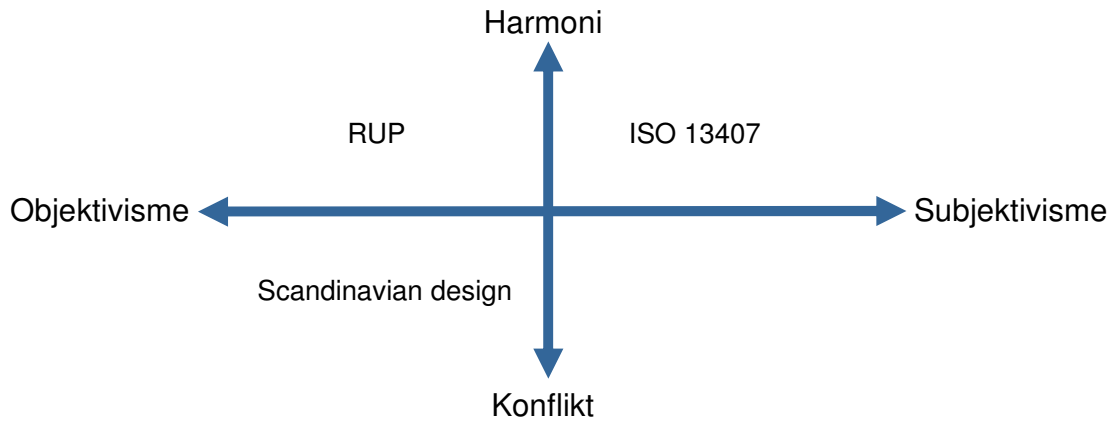
6.2.6 Paradigmer i systemutvikling

I hvilken grad kan vi bruke erfaringene fra dette prosjektet til å forbedre RUP? For å svare på dette spørsmålet har jeg tenkt å bruke modellen utviklet av Rudy Hirschheim and Heinz K. Klein (se kapittel 2.4). Hvor er det naturlig å plassere RUP i denne modellen, og hvor passer ISO 13407 best?

I kapittel fire konkluderte vi med at RUP i større grad må ta hensyn til menneskelige faktorer i sin systemutviklingsprosess. Blant annet bør man ha større fokus på forstå systemets brukskontekst og metoder for å kunne kommunisere dette. Dette betyr altså at RUP må bevege seg til høyre på objektivisme-subjektivisme aksen(altså mot subjektivsme). Tidligere i dette kapitlet har vi sett på noen av årsakene til at brukermedvirkning kan være vanskelig i utviklingsprosjekter. Hirschheim og Klein poengterer at systemutviklingsprosessen kan være påvirket av ulike inntresser. Det kan fort oppstå ulike typer konflikter og makten mellom de ulike aktørene kan være skjevt fordelt. RUP har ingen virkemiddel for å løse slike konflikter, og de viktigste tekstene om RUP[Jacobsen et al. 1999, Kruchten 2000] har ikke noen form for fokus på denne problematikken. Jeg vil derfor plassere RUP i funksjonalismen.

ISO 13407 oppfordrer til tverrfaglighet og benytter seg av metoder hentet fra blant annet psykologi og antropologi. Kontinuerlig testing opp mot potensielle brukere er også et viktig element. Jeg vil derfor plassere ISO 13407 til høyre for RUP i objektivisme-subjektivisme aksen. Som RUP har ISO 13407 ingen fokus på de utfordringene som vi finner i Hirschheim og Kleins orden-konflikt akse. Jeg vil derfor plassere ISO 13407 innen radikal strukturalisme.

Den Skandinaviske design tradisjonen er et eksempel på sosial relativisme[Hirschheim & Klein 1989]. Her finner vi et stort fokus på makt og interessekonflikter. Et tilbakevendende tema er forholdet mellom arbeidsgiver og arbeidstaker. Det er også stort fokus på at brukere skal være involverte i prosjektet og ha reelle påvirkningsmuligheter. Systemutvikleren er på brukerens side. De er derimot ikke så stort fokus på kompleksiteten ved å forstå brukere og ved å forstå arbeid, som du ofte finnes i brukersentrerte metoder.



Figur 6.2 Ulike utviklingsmetodikker inndelt i paradigmer.

6.2.7 Begrenset rasjonalitet

Hvordan kan vi ta høyde for konflikter og skjev maktfordeling i en utviklingsmetodikk? For å se nærmere på dette kan vi bruke John Forester's kategorisering av begrenset rasjonalitet (se kapittel 2.5). Forester har laget en oversikt over grader av kompleksitet en leder vil møte i en organisasjon. Ved å kjenne hvor komplekst et problem er vil man kunne finne ulike strategier som passer for hver situasjon. De fire typene er: kognitive begrensninger hos prosjektleder, sosial differensiering –arbeidsdeling, pluralistisk konflikt og reproduserbare strukturelle skjevheter. Jeg vil her se på hvordan RUP håndterer disse utfordringene.

Begrenset rasjonalitet 1: Kognitive begrensninger
Dette er Forester's enkleste scenario. Vi har kun en planlegger som ikke er i konflikt med den eksterne verden. Utfordringene for denne planleggeren ligger i å ta de riktige avgjørelsene.

RUP tilnærming til dette problemet er å tilby prosjektleder og utviklere bedre analytiske hjelpemidler. UML-diagrammer og risikoanalyse er eksempler på dette. I tillegg finnes det en rekke IT-system som kan hjelpe utviklere. Rational har en rekke informasjons-systemer som kan hjelpe til med prosjekt gjennomførelsen. For eksempel Rational Rose, RequistePro, ClearCase, ClearQuest og TestStudio.

Begrenset rasjonalitet 2: Sosial differensiering –arbeidsdeling
Alle utfordringene i begrenset rasjonalitet 1 gjelder også i dette scenariet. Men nå er ikke vår aktør lenger alene. Nå må hun forholde seg til andre aktører som hun samarbeider med. Alle disse aktørene er samarbeidsvillige, men kan ha andre perspektiv og en annen virkelighetsforståelse.

Økt rasjonalitet kan oppnås ved å etablere gjensidige kommunikasjons- og læringssystemer. RUP i seg selv er et kommunikasjons- og læringssystem. En av hovedhensiktene til RUP er å gi de ulike aktørene i et systemutviklingsprosjekt et felles språk de kan bruke. Selve utviklingsprosessen og UML er de viktigste elementene i dette språket.

Begrenset rasjonalitet 3: Pluralistisk konflikt
Inkluderer også utforingene i begrenset rasjonalitet 1 og 2. I tillegg forutsettes at ikke alle aktører er fullt ut samarbeidsvillige.

For å øke rasjonaliteten må man her skape fellesinteresser/ interessebalanse mellom de ulike aktørene. Verken RUP eller ISO 13407 har, slik jeg ser det noe fokus på denne problemstillingen.

Begrenset rasjonalitet 4: Reproduserte strukturelle skjevheter
Her inkluderes begrenset rasjonalitet 1 til 3. I tillegg til interessemotsetningene i begrenset rasjonalitet 3, så er aktørens evne til å handle og investere ulikt fordelt, noe som setter sterke føringer på hvordan aktørene handler. Disse skjevhetene er reproduserbare.

Anbudsproblematikk er et eksempel på reproduserbare strukturelle skjevheter. Siden anbudsprosjekter ofte fører til manglende brukermedvirkning, så fører det også til at brukere av IT-system har mindre makt i slike prosjekt. Utviklere har også mindre mulighet til å gjennomføre sine arbeidsoppgaver på en ønskelig måte.

Økt rasjonalitet kan skje på flere måter. Ved å endre lovverket i forhold til anbud i offentlig sektor kan man for eksempel bedre arbeidsvilkårene for systemutviklere. Det finnes også andre måter å angripe dette problemet. Dette skal jeg komme tilbake til i kapittel 7. Verken RUP eller ISO 13407 har, slik jeg ser det, noe fokus på denne problemstillingen.

6.3 Foreløpig konklusjon

Vi har allerede kommet fram til at RUP trenger å integrere brukersentrerte metoder i sin utviklingsprosess (se kapittel 4). RUP må gi systemutviklere verktøy for å analysere og beskrive behovene til kunder og brukere og å undersøke brukskonteksten. RUP bør også ha bedre verktøy og artefakter for å dokumentere denne kunnskapen. Hvilke verktøy som er mest hensiktsmessig er derimot avhengig av egenskapene til systemet som skal utvikles, type prosjekt og konteksten til prosjektet. Å planlegge disse aktivitetene bør være en viktig del av RUP.

Det vi har lært fra Delta-prosjektet er at slike metoder kan være vanskelige å integrere. Det er ikke nok å integrere metoder som er brukersentrerte, vi må i tillegg ta høyde for de konfliktene og strukturene som eksisterer i faktiske organisasjoner. Det finnes mange typer konfliktsituasjoner som oppstår i systemutviklingsprosessen. I dette kapittelet har vi sett nærmere på noen av disse. Slik jeg ser det tar ikke RUP høyde for slike konflikter. For eksempel: hvordan skal man legge til rette for iterativ utvikling og brukermedvirkning i en anbudsprosess? Anbudsproblematikk er et eksempel på en struktur som kan hindre brukersentrert design. Det er også mange andre aspekter som kan skape problemer for brukersentret design. Forskjellige elementer som kontrakt, forholdet mellom ledelse og bruker, økonomi, tid tilgjengelig med mer, legger føringer for hvilke metoder man kan bruke og hvilken form prosessen kan ha. Denne problemstilingen må bli mer tydelig i RUP!

Hvis RUP som metodikk tar høyde for disse aspektene vil det være lettere å kunne integrere brukersentrerte metoder på en måte som vil fungere i praksis. Jeg vil diskutere forskjellige tilnærminger til dette i kapittel 7.

7 Forslag til forbedringer

Jeg vil i dette kapittelet svare på mitt tredje forskningsspørsmål. Hvordan kan brukersentrerte metoder integreres i RUP?

7.1 To nivå

Det jeg har funnet så langt i denne oppgaven er at RUP bør forbedres på to nivå. På den ene siden må brukersentrerte designmetoder integreres i RUP. På den andre siden ser jeg at det er mange aspekter som legger hindringer for brukersentrerte designmetoder. Disse aspektene må bli mer synlig i RUPs utviklingsfilosofi, slik at man kan ta høyde for dem under planleggingen av utviklingsprosjekt.

Brukersentrerte metoder

Brukersentrerte metoder bør være en integrert del av RUPs standardmetodikk, og ikke forvist til en egen plug-in. De fleste utviklingsprosjekt kan dra nytte av brukersentrerte metoder i en eller annet form. Det er derimot ikke nok å legge til brukersentrerte metoder i selve prosessen. Det bør bli synlig at brukersentrerte designmetoder er en viktig del av RUPs utviklingsprosess. Blant annet bør brukersentrerte metoder være en del av RUPs suksesskriterier.

Nøyaktig hvilke metoder som trengs er forskjellig fra gang til gang. Det er blant annet avhengig av type IT-system, utviklingskontekst og kunnskapsnivået til utviklerne. Å finne de riktige metodene bør derfor være en viktig aktivitet i RUP. RUP bør ha fokus på å markedsføre brukersentrerte metoder. De brukersentrerte metodene bør ha en sentral plass i de verktøyene og bøkene som blir produsert.

Rammebetingelser

I kapittel 5 og 6 så vi på noen organisasjonsmessige hindringer for brukersentrert systemutvikling. Hvordan organisasjonsmessige aspekter påvirker systemutviklingsprosessen er i veldig liten grad et tema i RUP. RUPs verktøy og de mest sentrale bøkene om RUP [Jacobsen et al. 1999, Kruchten 2000] berører ikke dette teamet. Ønsker man å drive brukersentrert systemutvikling er det derimot et viktig emne. For å legge til rette for brukersentrert systemutvikling må denne diskusjonen bli en del av RUP. På den ene siden må Rational bevisstgjøre sitt publikum på disse utfordringene. På den andre siden bør RUP kunne tilby verktøy for å løse slike problemer. I kapittel 7.2.1 vil jeg se noe nærmere på hvordan anbudsproblematikken beskrevet i kapittel 5 og 6 kan angripes.

7.2 Forbedringer i caseprosjekt

Jeg vil her se nærmere på hvordan brukersentrert systemutvikling kunne vært praktisert i Delta-prosjektet.

7.2.1 Rammebetingelser

Siden anbudsprosjekter ofte fører til manglende brukermedvirkning, så fører det til at brukere av IT-system har mindre makt i slike prosjekt. Det er vanlig at formen på anbudsprosjekter gjør iterativ utvikling vanskelig. Utviklere har derfor mindre mulighet til å gjennomføre sine arbeidsoppgaver på en ønskelig måte. Altså, vi har en lav grad av brukermedvirkning og en lav grad av iterativ utvikling. Dette er typiske trekk ved anbudskonkurranser i forbindelse med utviklingen av IT-system. Vi kan derfor si at Delta-prosjektet innehar reproduserbare strukturelle skjevheter, Forresters fjerde grad av begrenset rasjonalitet (se kapittel 2.5). Økt rasjonalitet skjer ifølge Forrester gjennom kombinasjon av maktutøvelse, ideologiproduksjon og interesseutjevning (kompatible incentivsystemer). Hvis vi tar utgangspunkt i at vi ønsker å legge til rette for brukersentrert systemutvikling og iterativ utvikling, hvilke hjelpemidler kan vi ta i bruk? Antagelig må vi endre formen på anbudsprosjektene slik at vi får en interesseutjevning mellom de ulike partene (utviklere, kunde, brukere og andre interessenter)

Fra produkt til prosess

[Grøndbæk et al. 1993] foreslår å gå fra et produktfokus til et prosessfokus. Dette innebærer at prosessen i seg selv blir et mål. Det primære målet er ikke utformingen av selve produktet, men å gjennomføre den prosessen som er mest egnet for å bygge et slikt produkt. Prosessfokus kan bidra til å bedre rammebetingelsene for brukersentrert design. Delta-prosjektet hadde i utgangspunktet et prosessfokus siden de ville bruke RUP som utviklingsmetodikk. De ønsket å benytte seg av brukersentrerte metoder. Dette skjedde ikke i dette prosjektet.

Et problemområde kan være forholdet mellom design og programmering. En av de mest tidkrevende aktivitetene i programvareutvikling er programmering. Hvis en leverandør ønsker å maksimere fortjenesten, så bør man ikke bruke mer resursser på programmering enn nødvendig. Det betyr at leverandøren ofte må ta valg i forhold til om man vil ha best mulig system eller høyest mulig fortjeneste. De beste løsningene kan ofte medføre mer programmering. Ønsket om fortjeneste kan også hemme graden av iterativ utvikling. Kan kunden stole på at leverandøren bestandig tar de avgjørelsene som gir best mulig produkt? Kunden av et IT-system har sjelden kompetansen til å vurdere alle avgjørelser som blir tatt.

Skille design og programmering
En måte å bedre situasjonen på, kan være å skille mellom design og programmering. Det vil si, la ulike leverandører ta ansvaret for systemets utforming og implementasjon (implementasjon betyr i dette tilfellet selve konstruksjonen av systemet).

Det er vanlig å gjøre et forarbeid i forbindelse med anbudsprosjekt. I Delta-prosjektet ble det utformet en kravspesifikasjon i forkant av anbudsrunden. Denne kravspesifikasjonen ble utformet av kunden og en ukjent leverandør. En slik kravspesifikasjon er nødvendig fordi den gir leverandøren et innblikk i hva som skal utvikles. Den er også nødvendig fordi den gir testbare mål slik at man kan ha en enighet mellom kunde og leverandør om hva som skal utvikles. Det er stor forskjell på en kravspesifikasjon og et komplett design. I prosessen med å utvikle et komplett design har leverandøren en dobbeltrolle. På den ene siden skal leverandøren designe et best mulig system, på den andre siden skal prosjektet ha høyest mulig overskudd. For å sikre seg fortjeneste på prosjektet må leverandører ofte velg lite tilfredsstillende løsninger. Hvis leverandøren i tillegg er presset på pris, så vil marginene være små. I verste fall blir marginene så små at det ikke lønner seg for leverandøren å lage et bra produkt. Hvis vi skiller design og konstruksjon vil vi kunne fjerne denne dobbeltrollen.

Målet bør være å skape en vinn-vinn situasjon mellom alle parter i prosjektet. Et eksempel på utforming kan være følgende: Et designoppdrag blir lagt på anbud. Potensielle leverandører blir vurdert på pris, kompetanse og designprosess. Viktige kriterier bør være iterativ design, brukermedvirkning og dyktige designere. Siden dette firmaet ikke skal produsere ferdig kode, bør de benytte seg av bruk-og-kast prototyper. Leverandøren bør også belønnes hvis de gjør en bra jobb. Hvis det er mulig å opprette objektive kriterier knyttet til kvaliteten på designet, så kan disse være utløsere for en eventuell bonus. En bonus kan for eksempel knyttes til prosentvis nedgang i opplæringstid for systemet. Når dette firmaet har utviklet et detaljert designdokument, legges programmeringsprosjektet ut på anbud. Potensielle leverandører blir vurdert på pris, kompetanse og programmeringsprosess. Kriteriene for en eventuell bonus kan blant annet knyttes til rask leveranse, bra ytelse og få programmeringsfeil. Det bør være mulig å utvikle et samarbeid mellom de som designer og de som programmerer, slik at vi kan få iterativ utvikling også under programmeringsfasen. Et slikt samarbeid må ha strenge rammer for å fungere.

Å skille mellom design og implementasjon på denne måten ligner på arbeidsdelingen som er vanlig i byggebransjen. Ofte får arkitektkontor ansvaret for å finne en hensiktsmessig arkitektur. Arkitektkontoret kan få oppdraget på flere måter, for eksempel via en konkurranse. Byggingen blir lagt ut på anbud når arkitektene har funnet en overordnet arkitektur. Arkitektene og vinneren av anbudskonkurransen jobber så sammen for å detaljere løsningen.

En slik organisering vil forhåpentligvis kunne skape en interesseutjevning som legger til rette for brukersentrert design og vinn-vinn situasjoner mellom de ulike partene. Designerne vinner fordi de blir belønnet for å designe et bra system. Programmererne vinner fordi de får et

forutsigbart prosjekt med mulighet for profitt. Og kunden øker sine sjanser for å få et bra IT-system til slutt. For å få dette til å fungere, så er det viktig at ikke samme leverandør er ansvarlig for både design og implementasjon.

Jeg her har kommet med to tilnærminger til kontraktsproblematikk. Kontrakter innen systemutvikling er et stort og komplekst tema. Blant annet setter lover føringer på hvordan de kan utføres. Mitt mål i denne oppgaven er ikke å komme med utfyllende råd om hvordan slike kontrakter bør utformes, men heller å poengtere at dette er viktige tema i forhold til utviklingen av brukervennlige IT-system. Konflikter og ulike syn / viktighetsoppfatning mellom ulike grupperinger er også viktige elementer som man må ta hensyn til under planleggingen av IT-prosjekt. Først når vi greier å legge det rette grunnlaget, kan vi benytte oss av brukersentrerte metoder. Jeg vil i det følgende se nærmere på metoder som kunne kommet godt med i Delta-prosjektet.

7.2.2 Brukermedvirkning

Utviklerne i Delta-prosjektet hadde noe kontakt med brukere, men de kunne med fordel involvert brukerne i større grad. Hva slags type medvirkning burde vi ha i dette prosjektet? Først må vi se nærmere på hva slags brukere vi har i dette prosjektet. Det er et stort antall potensielle brukere ved flere organisasjoner. Noen av disse organisasjonene er mer sentrale enn de andre. De vil bruke verktøyet oftere og har behov for mer funksjonalitet enn de andre. Selv om de vil bruke verktøyet ofte, er det ikke en sentral del av deres hverdag. I dette prosjektet er det viktig at systemutviklerne har en forståelse for brukernes faktiske arbeidsoppgaver og brukskonteksten til systemet. Arbeidsoppgavene er preget av en høy grad av domenekunnskap. Det er også viktig å forstå de som skal bruke systemet. For eksempel, hva er brukerens motivasjon for å bruke verktøyet? Videre er det viktig å bruke denne kunnskapen til å utforme systemet på best mulig måte. For eksempel, hvilke funksjoner skal prioriteres?

Faktiske arbeidsoppgaver er vanskelige å forstå. En organisasjons formelle beskrivelse av ulike roller er sjeldent informativ nok til å informere design. Hva som er den opprinnelige planen og hvordan arbeid faktisk utføres, er ofte to forskjellige ting [Suchman 1987, Miller 1992]. Det finnes mange ulike metoder. En populær metode er workshops. Workshops var en hyppig brukt metode i Delta-prosjektet. Deltagerne i disse workshopene var utviklerne og kunden, men det ble også gjennomført en workshop med potensielle brukere. Workshops kan være nyttige verktøy. Blant annet kan de brukes til å skape en felles forståelse blant de ulike interessentene i prosjektet. De er derimot ikke like godt egnet til å forstå arbeidsoppgaver og brukskontekst. [Nielsen 2001] rapporterer flere problem med selvrapportering (se også kapittel 2.3): det er ikke uvanlig å vri noe på sannheten for å få svar som er sosialt akseptable (dette gjelder spesielt workshops og lignende metoder), menneskelig hukommelse er for svak til å gi de detaljene som kan være nyttige i brukergrensesnittdesign og mennesker rasjonaliserer ofte sine handlinger.

En måte å unngå noen av disse problemene er å benytte seg av etnografisk metode. Problemet med etnografisk metode er at den ikke tar hensyn til at innsamlet data skal brukes til design og at etnografiske studier er svært tidkrevende. En metode som låner fra etnografi er Contextual Inquiry. Contextual Inquiry egner seg til dette prosjektet fordi det er en metode som har høyt fokus på å forstå arbeid. Denne metoden er tilpasset de behovene man har i et systemutviklingsprosjekt. Jeg vil derfor anbefale bruk av Contextual Inquiry eller lignende metoder i slike prosjekt. Du kan lese mer om Contextual Inquiry i kapittel 2.2.4.

7.2.3 Dokumentasjon og kommunikasjon

Den kunnskapen man tilegner seg bør dokumenteres på en hensiktsmessig måte. I RUP er det vanlig å bruke diverse UML-modeller. Den viktigste i forhold til brukergrensesnittdesign er Use-Case-modellene. I kapittel fire konkluderte vi med at Use-Case-modeller ikke er tilstrekkelig. Vi bør integrere artefakter som i større grad er tilpasset de behovene vi har når vi skal designe IT-system. Jeg vil her foreslå bruken av ulike artefakter som kunne vært nyttig i Delta-prosjektet. Disse er: Personas, Scenarios og Work Models. Slike artefakter trenger ikke kun være nyttig for designere. I Delta-prosjektet var det mye fram og tilbake mellom kunde og designer. Et sett med artefakter som er basert på reelle data vil kunne forenkle disse diskusjonene og forhindre mye fram og tilbake. Disse artefaktene har flere bruksområder: utgangspunkt for design, kommunisere design og utgangspunkt for testing. Disse artefaktene er beskrevet i kapittel 2.2.5.

Men det er problemer knyttet til å bruke disse artefaktene. Selv om disse verktøyene virker enkle på overflaten så er det en høy terskel for å ta de i bruk. Det krever et visst modningsnivå for å lage og bruke disse artefaktene. For eksempel: det blitt rapportert ulike problemer med bruk av personas [Gruding & Pruitt 2003, Blomquist & Mattias 2002]. Det er derfor viktig å bygge kompetanse på dette over tid.

7.2.4 Brukergrensesnittdesign

Å begynne med blanke ark kan være skremmende. Susanne i Delta-prosjektet mente at spranget fra kravspesifikasjon til brukergrensesnittdesign var særdeles vanskelig. Jobben til Susanne kunne vært lettere med modellene beskrevet over, men er det tilstrekkelig? I RUP har man en detaljert beskrivelse av brukergrensesnittaktiviteten (se kapittel 2.1.4). Ved hjelp av Use-Case-modeller og en steg-for-steg prosedyre skal man kunne designe brukergrensesnittene. Tilnærmingen som er beskrevet i 2.1.4 ble ikke brukt i Delta-prosjektet. Dette kan være fordi den ikke er spesielt kjent. Tilnærmingen er kun publisert i en bok [Kruchten et al. 2001], som jeg har kjennskap til.

En kritikk av denne tilnærmingen kommer fra Göransson, Lif og Gulliksen (se kapittel 4.4.1). De mener at brukergrensesnittdesign er en kreativ prosess, som ikke kan beskrives som en steg-for-steg prosedyre. Mer konkret hevder de at det vil være uhensiktsmessig å følge prosedyren steg for sted, fordi det vil føre til et skjermbilde per Use-Case. Noe som igjen vil føre til fragmenterte system. Jeg er enig i kritikken til Göransson, Lif og Gulliksen, men samtidig mener jeg at det er mulig å tilby en prosess som gjør brukergrensesnittdesign lettere.

En annen prosess for brukergrensesnittdesign er foreslått av Alan Cooper og Robert Reimann [Cooper & Reimann 2003]. Denne prosessen er mer overordnet enn RUPs. Samtidig er den så konkret at den kan bidra til å gjøre arbeidet med brukergrensesnittdesign lettere. Blant annet foreslår Cooper og Reimann ulike konsepter som positur og patterns som kan bidra til å gjøre jobben med brukergrensesnittdesign lettere. Målet er ikke å gi detaljerte steg-for-steg instruksjoner, men heller å foreslå aktiviteter som gjør prosessen med å tegne skjermbilder lettere. En slik tilnærming må legge til rette for prøving, feiling og kreativ utfoldelse. Jeg skal ikke gå i dybden på denne metoden. Poenget er at det finnes alternativer til RUPs metode som i større grad baserer seg på den kunnskapen man har tilegnet seg innen brukersentrert design.

7.2.5 Brukertester

Det ble kun gjennomført en brukertest i Delta-prosjektet. Dette var på slutten av Elaborataion-fasen. Et viktig prinsipp i brukersentrert design er kontinuerlig testing [ISO 13407] mot brukere. Kontinuerlig testing kunne vært svært nyttig i Delta-prosjektet. Blant annet kunne dette bidra til å gjøre forholdet mellom designere og kunden mer produktivt. På samme måte som data fra en contextual inquiry kan bidra til å fokusere diskusjoner kunne data fra brukertester gi samme effekt. Tidlig i prosjektet kan man begynne med papirprototyper. Disse tar kort tid å utvikle, og er billige. Dette gir designerne muligheter til å eksperimentere og prøve ulike alternativ. Når man har kommet et stykke ut i Elaboration fasen kan man programmere mer realistiske prototyper. Det er viktig at man benytter anledningen til å teste systemet i Construction-fasen. Også etter at systemet er utviklet kan det være lurt å planlegge brukertester. Da har man muligheten til å teste et ferdig system, og man kan teste systemet på erfarne brukere. Dette kan være verdifulle innspill i en eventuell videreutvikling av systemet.

Det kan også være verdifull informasjon selv om man ikke har mulighet til å utbedre selve applikasjonen. Kunnskap om svakheter i systemet kan for eksempel brukes når man skriver brukermanualer eller når man oppretter brukerstøttetjenester.

7.3 Konsekvenser for RUP

I avsnitt 7.2 kom jeg med forslag til hvordan vi kunne bedre Delta-prosjektet. Her vil jeg se på mulige endringer i RUPs utviklingsmetodikk. Jeg vil komme med forslag til nye metoder og artefakter som kan inkluderes i RUP. Det er viktig å presisere at det ikke er nok å legge til nye metoder. En endring krever retoriske endringer. Retorikken er viktig for hvordan en utviklingsmetode blir tolket. RUP må fremheve viktigheten av brukersentrerte metoder. De som skriver tekster for Rational bør ha høyere fokus på menneskelige aspekter.

7.3.1 Nye suksesskriterier

RUP hevder å være basert på suksesskriterier fra flere tusen vellykkede prosjekt. Basert på mine funn vil jeg foreslå to nye suksesskriterier. På den ene siden bør RUP i større grad benytte seg av brukersentrerte metoder. For å understreke alvoret, bør dette være et eget suksesskriterium. På den andre siden er det viktig at det er en match mellom rammebetingelsene til prosjektet, prosessen og metodikken. Brukersentrerte prisniper blir oversett på grunn av ulike organisasjonsmessige faktorer, og de økonomiske rammer må stå i stil til planene for brukersentrert design. For eksempel, hvordan kan en anbudsrunde støtte iterativ utvikling og hvordan kan vi lage en anbudskonkurranse der utvikling vil skje med tilstrekkelig brukermedvirkning? Jeg foreslår to nye suksesskriterier:

- Det er viktig å benytte seg av brukersentrerte designmetoder når man skal kartlegge brukerbehov og designe løsninger.

- Prosjektets rammebetingelser må stå i stil til prosjektets prosess og metodikk. Rammebetingelsene til et prosjekt setter sterke føringer på hva slags type utviklingsmetodikk som er mulig.

7.3.2 Nye aktiviteter

Jeg vil her foreslå aktiviteter som bør integreres RUP, eller som bør få større oppmerksomhet enn de har i dag.

Større bevissthet rundt utviklingskontekst i prosjektplanlegging

I kapittel 5 og 6 så vi på noen av de organisasjonsmessige hindringene til brukersentrert systemutvikling. Hvordan organisasjonsmessige aspekter påvirker systemutviklingsprosessen er i veldig lav grad et tema i RUP. På den ene siden må Rational bevisstgjøre sitt publikum på disse utfordringene. På den andre siden bør RUP kunne tilby verktøy for å løse slike problemer. I 7.2.1 så jeg nærmere på hvordan anbudsproblematikken beskrevet i kapittel 5 og 6 kan angripes. Problemer knyttet til anbud er ikke de eneste hindringene. I andre kontekster vil man ha helt andre utfordringer. Det er utenfor rekkevidden til denne oppgaven å lage en komplett oversikt over disse utfordringene og å finne løsninger, men dette bør være et viktig tema i RUP.

Når man starter et nytt prosjekt, bør man være bevisst på rammebetingelsene til prosjektet. Hvis man ønsker å jobbe brukersentrert, så vil ulike kontekster gi ulike muligheter og problemer. Dette bør man være bevisst på under prosjektplanleggingen. Å planlegge prosjektet i forhold til utviklingskonteksten bør være en egen aktivitet. Denne aktiviteten må starte tidligst mulig i prosjektet. Hvilke muligheter man har er avhengig av ulike faktorer, blant annet makt og posisjon. I et anbudsprosjekt, har kunden gode muligheter for å legge til rette for brukersentrert utvikling. Leverandørene kommer derimot inn på senere tidspunkt, og har derfor mindre muligheter.

Utforme brukervennlighestskrav

Kravene til et IT-system skal være konkrete og målbare. De funksjonelle kravene i kravspesifikasjonene i Delta-prosjektet var konkrete og målbare. I Delta-prosjektet hadde man et krav knyttet til brukervennlighet:

> *"Brukergrensesnittet skal være enkelt å forstå, og ikke være vanskeligere å bruke enn en vanlig Internett-side. Den skal kreve minimalt med spesialopplæring, kompetanse og maskinvare." Hentet fra kravspesifikasjonen til Delta-prosjektet.*

Dette kravet er for vagt. Det er ikke konkret og det er ikke målbart. Å utforme brukervennlighetskrav er en stor oppgave. I de prosjektene hvor man ønsker å gjøre dette, bør det være en egen prioritert oppgave. Det bør derfor få større fokus i RUP

Iterativ utvikling
Er iterativ utvikling i RUP det samme som brukersentrert design? Göransson, Lif og Guliksen [Göransson et al. 2003] hevder at RUPs tilnærming til iterativ utvikling skiller seg kraftig fra iterativ utvikling i brukersentrert design. De skriver følgende:

> *"The definition of iteration within RUP differs significantly from how it is defined in for instance UCD. In RUP, an iteration is "A distinct sequence of activities with a base-lined plan and valuation criteria resulting in a release (internal or external)" [17]. The activities inside the iteration are laid out as a waterfall. This prohibits iterations to formally occur within workflows and activities. From a UCD point of view an iteration is a refinement of a certain part of the system (an increment), going through the stages analysis, design and evaluation until goals are reached."* [side 3, Göransson et al. 2003]

For å være i henhold til ISO 13407 må vi jobbe iterativt for å møte de brukervennlighetskravene som har blitt definerte. Det betyr at vi må analysere, designe og evaluere helt til vi har møtt brukervennlighetskravene. Dette krever justeringer i flere av arbeidsprosessene til RUP. Hovedutfordringen med iterativ utvikling er uansett ikke å utarbeide de riktige arbeidsprosessene. Dette er i min mening relativt enkelt. Det som er vanskelig er å legge til rette for iterativ utvikling i en organisasjonsmessig kontekst. Det er mange organisasjonsmessige aspekter som legger hindringer for iterativ utvikling, disse bør adresseres av RUP.

Brukermedvirkning
Som tidligere nevnt, bør brukermedvirkning være en viktigere del av RUP.
Rational bør ha et mer bevisst forhold til brukermedvirkning. Brukermedvirkning som tema må være mer synlig i de tekstene som Rational produserer.

Brukertesting
De RUP-aktivitetene som inneholder brukertesting finner vi hovedsakelig i requirements- og business modelling-disiplinene. Disse disiplinene finner for det meste sted i Inception og Elaboration. Disse aktivitetene er også en del av construction- og transitionfasene, men i betydelig mindre grad enn i Inception og Elaboration. Se figur 2.1 i kapittel 2. For eksempel, er brukbarhetstesting kun en del av brukergrensesnittprototypaktiviteten. Brukbarhetstester er ikke en del av de overordnede systemtestene. ISO 13407 argumenterer for kontinuerlig brukertesting gjennom hele prosjektet. Det betyr at slike aktiviteter bør være sentrale også i Construction-fasen. Også i Transition-fasen kan det være aktuelt med noen brukersentrerte aktiviteter. Dette fordi de kan være nyttige innspill til brukerveiledninger, brukerstøtte, nye versjoner av systemet og for å gjøre mindre endringer. Skal RUP følge ISO 13407 må brukertesting få en mer sentral del i metodikken.

Overgang mellom design og implementasjon
I denne oppgaven foreslår jeg nye artefakter som personas, scenarioer og work models (se 7.3.4). Slike artefakter er mer egnet til å støtte design og å kommunisere design. De er derimot ikke egnet til mer teknisk utvikling (for eksempel programmering). Da trenger man artefakter som er mer tekniske av natur. For eksempel Use-Case modeller og tilstandsdiagram. Det bør derfor være en bro mellom brukersentrert design og programmering. Hvordan bygge en slik bro kan du lese mer om i [Kapittel 9, Caroll 2000]

7.3.3 Nye roller

Basert på mine funn i kapittel 4 og i casestudiet, vil jeg foreslå fire nye RUP roller. Disse er: feltstudieansvarlig, interaksjonsdesigner, designkommunikator og brukertest ansvarlig. Som i RUP er det ikke en rolle per prosjektdeltager. En prosjektdeltager kan ha flere roller. De ulike rollene kan kombineres på ulike måter, avhengig av hva slags kompetanse som er tilgjengelig og hva slags type prosjekt det er. Samtidig som jeg foreslår fem nye roller vil jeg også argumentere for at et utviklingsprosjekt ikke har for mange designere. For mange kokker kan føre til kaos i en designsituasjon. Det er bedre at en mindre gruppe designere med utgangspunkt i prosjektets visjon kan utforme et helhetlig design. Ofte er mellom to og tre designere nok, selv i store prosjekt.

Feltstudieansvarlig
Tidligere i oppgaven har jeg argumentert for viktigheten av å være brukersentrert. Hvordan man skal involvere brukere varier fra prosjekt til prosjekt. I Delta-prosjektet ville jeg anbefalt Contextual Inquiry som teknikk. I andre prosjekt ville andre teknikker vært hensiktsmessige, for eksempel rollespill [Svanæs & Seland 2003]. De viktigste oppgavene til feltstudieansvarlig er å planlegge brukermedvirkningen, gjennomføre brukermedvirkning og å dokumentere funnene. Samtidig er det ofte vanskelig å få gjennomført brukermedvirkning. Det finnes mange organisatoriske hindringer. Feltstudieansvarlig bør være bevisst på de utfordringene som finnes og være i stand til å håndtere disse på best mulig måte. Feltstudieansvarlig bør sammen med prosjektleder tidlig i prosjektet kartlegge de ulike utfordringene knyttet til å gjennomføre brukermedvirkning i det bestemte prosjektet. Videre bør de finne virkemidler som kan øke mulighetene for brukermedvirkning. Resultatet av arbeidet til feltstudieansvarlig er ulike artefakter som dokumenterer funnene fra brukermedvirkningen. Dette bør være artefakter som personas, scenarios, work models og video.

Interaksjonsdesigner
Det er viktig å benytte data som blir samlet inn på en best mulig måte, for å lage et brukervennlig grensesnitt. Dette er hovedoppgaven til interaksjonsdesigneren. Selv om det er ulike roller, kan det være samme person som er feltstudieansvarlig og interaksjonsdesigner. Fordelen med dette er at det minker gapet mellom bruk og design. En for hør grad av spesialisering kan føre til problemer. En feltstudieansvarlig som ikke er involvert i

designprosessen kan ha problemer med å se hva som er viktigst å studere, og en interaksjonsdesigner som ikke er med på brukermedvirkningen kan ha problemer med å tolke data fra feltstudiespesialisten. Problemer knyttet til integrasjon av disse to fagområdene diskuteres nærmere i [Cooper & Reiman 2003]. Med utgangspunkt i prosjektets visjon / forretningsmål, tekniske begrensninger og kunnskap om sluttbrukerne skal interaksjonsdesigneren designe de ulike brukergrensesnittene. Resultatet av arbeidet til interaksjonsdesigneren er designskisser og papirprototyper. Rollen interaksjonsdesigner erstatter rollen User interface designer.

Designkommunikator

I min casestudie ser vi at det er mye fram og tilbake mellom designere og kunde. Dette har mange årsaker. En av disse er ineffektiv kommunikasjon mellom kunde og designere. I kapittel 6.2.4 argumenterer jeg for at designavgjørelser må baseres på data og at kommunikasjonen mellom kunde og designer er en svært viktig del av prosjektet. Å kommunisere designløsninger kan være svært utfordrende. Jeg mener at dette bør være en egen rolle. Designkommunikatoren er ansvarlig for å kommunisere designet til de ulike interessentene i prosjektet. Dette kan være interessenter som kunden, brukere, ledelsen, markedsføringsavdelingen og programmerere. De ulike interessentene har ofte ulike behov, og designkommunikator må være oppmerksom på disse. Resultatet av arbeidet til designkommunikatoren er et designdokument som beskriver systemets brukergrensesnitt og ulike presentasjoner. En designkommunikator kan erstatte en kravhåndterer i små og mellomstore prosjekt.

Brukertestansvarlig

Prosjektdeltageren med denne rollen er ansvarlig for å planlegge, forberede, utføre og dokumentere brukertester. Funnene fra brukertester skal brukes til å bedre designet til systemet.

7.3.4 Nye artefakter

I kapittel 4 konkluderte jeg med at de tilgjengelige UML-modellene ikke var tilstrekkelige. I brukersentrert design har man utviklet andre artefakter som er mer egnet til å støtte designprosessen. Jeg foreslår her at tre av disse blir introdusert som nye elementer i RUP. Disse artefaktene er personas, scenarioer og work models. Jeg vil ikke si så mye om de ulike artefaktene. Det er det andre som gjør bedre, men jeg skal begrunne hvorfor jeg har tatt de med

Personas

I kapittel 4 argumenterer jeg for at man trenger mer ”soft” informasjon når man skal lage brukervennlige IT-system. Dette inkluderer blant annet utfyllende informasjon om systemets brukere. Hvilke mål har de? Hva er deres holdninger og følelser i forhold til arbeidsoppgaver og andre viktige spørsmål? Hensikten med personas er å kommunisere slik informasjon.

Du kan lese mer om personas i kapittel 2.2.5

Scenarioer

Det er selvfølgelig ikke nok å vite hvem brukeren er. Systemutviklere bør også ha god kunnskap om de arbeidsoppgavene som skal gjennomføres. I kapittel 4 hevder jeg at Use-Case modeller ikke er tilstrekkelige fordi de mangler essensiell informasjon. Disse bør derfor suppleres med scenarioer.

Du kan lese mer om scenarioer i kapittel 2.2.5

Work Models

Det hevdes fra flere hold at det er lurt å bruke grafiske modeller når man skal designe IT-system[Beyer & Holtzblatt 1998, Suchman 1989]. Use-Case modeller er grafiske. Men Use-Case modeller er for abstrakte, og fanger ikke opp aspekter som uformell kommunikasjon og arbeidskultur. En alternativ tilnærming til å visualisere arbeid grafisk er work models. Work models tar sikte på å visualisere arbeid fra ulike ståsted. Dette kan være arbeidsflytt, kulturelle faktorer eller det fysiske arbeidsstedet.

Du kan lese mer om work models i kapittel 2.2.5

8 Konklusjon

Jeg vil i dette kapittelet presentere min konklusjon og ha en diskusjon rundt min forskningsmetode.

8.1 Konklusjon

Mitt første forskningsspørsmål i denne oppgaven er: **Hvordan er RUPs tilnærming til systemutvikling sammenlignet med brukersentrerte metoder?** RUP og brukersentrert design representerer to ulike paradigmer innen utviklingen av IT-system. RUP har sitt utspring i et teknisk orientert miljø og har sin styrke i metodens evne til å håndtere de tekniske utfordringene man finner i store utviklingsprosjekt. Brukersentrert design har derimot et fokus på menneskelige aspekter i forhold til teknologi og er derfor mer egnet for å utvikle brukervennlige løsninger. De bruker ulik retorikk for å rettferdiggjøre sin eksistens. RUP markedsfører ønsket om en solid og repeterbar prosess. I brukersentrert design er det derimot brukeren som står i sentrum. Her er brukervennlighet et sentralt begrep. For å kunne utvikle brukervennlige løsninger er det flere faktorer som må ligge til rette. Noen av de mest anerkjente er brukermedvirkning, iterativ utvikling og å ha en god forståelse av systemets brukskontekst.

Hvilke utfordringer har vi ved å integrere slike metoder i RUP? Mitt andre forskningsspørsmål i denne oppgaven er: **Hvilke faktorer påvirker den faktiske anvendelsen av brukersentrert metode i RUP-prosjekter?** For å se nærmere på dette spørsmålet, har jeg gjennomført en casestudie. Min casestudie viser at brukermedvirkning og iterativ design kan være vanskelig å gjennomføre på grunn av organisatoriske hindringer. Også andre casestudier viser ulike organisatoriske hindringer til brukermedvirkning og iterativ design [Grudin 1991, Grudin & Poltrock 1994]. Organisatoriske utfordringer er i veldig lav grad et tema i RUP.

I Delta-prosjektet får utviklerne minimal tilgang til brukere og utviklingsprosessen er ikke så iterativ som den kunne vært. Det er mange forklaringer på dette, men den mest sentrale er forholdet mellom kunde og leverandør. I dette prosjektet, så ønsket ikke kunden brukermedvirkning. Kundens prosjektleder påtok seg en rolle han selv ga navnet brukerambassadør. Har leverandøren her et ansvar for å påpeke problemene knyttet til en slik måte å arbeide? Her har leverandøren en dobbelt rolle. På den ene siden skal leverandøren designe et bra produkt. Noe som etter ISO 13407 innebærer blant annet iterativ utvikling og brukermedvirkning. På den andre siden skal de tjene penger og gå med overskudd. Iterativ utvikling koster penger og i et fastprisprosjekt blir antall iterasjoner fort begrenset av hvor mye penger som er tilgjengelig. I en anbudssituasjon er leverandører ofte presset på pris. Brukermedvirkning fører altså til økt risiko for leverandøren. For leverandøren er det lettere å forholde seg til en kravspesifikasjon. Det er mye lettere å lage et system som er i henhold til en kravspesifikasjon enn å måtte forholde seg til brukere.

Mitt siste forskningsspørsmål er: **Hvordan kan brukersentrerte metoder integreres i RUP?** I kapittel 2 så jeg på noen brukersentrerte metoder. Metoder som Contextual Inquiry, personas og scenarioer. Slike metoder bør bli en sentral del av RUP. Det finnes også en rekke andre brukersentrerte metoder. Konsekvensene for RUP er beskrevet detaljert i kap. 7.3. De viktigste anbefalingene er:

Nye suksesskriterier:

- Det er viktig å benytte seg av brukersentrerte designmetoder når man skal kartlegge brukerbehov og designe løsninger.
- Prosjektets rammebetingelser må stå i stil til prosjektets prosess og metodikk. Rammebetingelsene til et prosjekt setter sterke føringer på hva slags type utviklingsmetodikk som er mulig.

Nye aktiviteter:

- Større bevissthet rundt utviklingskontekst i prosjektplanlegging
- Utforme målbare brukervennlighestskrav
- Iterativ utvikling i henhold til ISO 13407
- Større fokus på brukermedvirkning
- Større fokus på brukertesting

Nye roller:

- Feltstudieansvarlig
- Interaksjonsdesigner
- Designkommunikator
- Brukertestansvarlig

Nye artefakter:

- Personas
- Scenarioer
- Work Models

Det er ikke nok å kun integrere metoder som er brukersentrerte, vi må også ta høyde for de konfliktene og strukturene som eksisterer i faktiske organisasjoner. I kapittel 6 har jeg sett nærmere på hvordan organisatoriske aspekter påvirker systemutviklingsprosessen. Disse utfordringene er ikke et tema i RUP. For eksempel: hvordan skal man legge til rette for iterativ utvikling og brukermedvirkning i en anbudssituasjon? Forskjellige elementer som

kontrakt, forholdet mellom ledelse og bruker, økonomi, tid tilgjengelig med mer, legger føringer for hvilke metoder man kan bruke og hvilken form prosessen kan ha. Denne problemstillingen må bli mer tydelig i RUP. Slike utfordringer må også bli mer tydelig i brukersentrerte metoder.

Hvis vi kan skape det riktige miljøet, så vil det være lettere å benytte seg av brukersentrerte metoder. Dette er utfordrende, og berører mange aspekter. Det første steget er å skape en større bevisstgjøring rundt disse aspektene. Jeg mener at det er innenfor rammene til en utviklingsmetodikk å se, og finne virkemidler for å løse slike problem. Problemer knyttet til anbud kan løses ved strukturelle endringer. For eksempel ved å skille design og konstruksjon(se kapittel 7.2.1). En utviklingsmetodikk kan ikke løse alle de organisasjonsmessige utfordringene, men RUP må ha en større bevissthet rundt dette aspektet. Kunnskap og holdninger er også viktig. Brukersentrert design er kunnskapskrevende, og kan ikke reduseres til en enkel prosedyre. Dyktig prosjektdeltagere med en glød for brukersentrert design er også viktig.

8.2 Metodediskusjon

Jeg vil her vurdere gjennomføringen av casestudiet. For å gjøre dette vil jeg bruke en metode utviklet av [Klein og Myers 1999]. Se også kapittel 3.2.7. I artikkelen "A set of principles for Conducting and Evaluating Interpretive Field Studies in Information Systems" presenterer Klein og Myers syv prinsipper for å utføre og evaluere kvalitet innen fortolkende feltstudier. Jeg skal her se på min case studie i lys av disse prinsippene.

Den hermeneutiske sirkel
Ifølge prinsippet om den hermeneutiske sirkel, så oppnår mennesker forståelse ved å se på både enkeltelementene og helheten enkeltelementene former. Modning over tid er viktig for å oppnå en høyre forståelse. Dette prinsippet må ses i sammenheng med de andre prinsippene til Klein og Myers.

Kontekstualisering
Det er viktig å reflektere rundt den sosiale og historiske bakgrunnen, for å forstå hvordan den nåværende situasjonen oppsto.

Jeg brukte en del tid på å sette meg inn i historien til organisasjonene som var involverte i dette prosjektet. Dette var nyttig for å forstå hvorfor systemutviklingsprosessen ble som den ble. Det er dessverre begrenset hva jeg kan skrive om dette i oppgaven av hensyn til anonymitet. Organisasjonsmessige aspekter ble en viktig faktor i dette casestudiet. Jeg så på forholdet mellom de ulike utviklerne, utviklerne og brukerne og kunde og leverandør.

Interaksjon mellom forsker og forskningsobjektene
Det er viktig med refleksjon rundt hvordan forholdet mellom forsker og forskningsobjekt påvirker datainnsamlingen.

Jeg brukte flere teknikker for å samle data fra forskningsobjektene. De viktigste var intervju og observasjon. Jeg benyttet meg av semistrukturerte intervju, og jeg prøvde å gjennomføre disse i henhold til rådene gitt av [Robson 1993]. Jeg følte at informantene stolte på meg, og at de var villige til å gi meg den informasjonen jeg trengte. Det var derimot vanskelig å spørre på en riktig måte. Jeg la fort merke til at det var vanskelig å gjennomføre gode intervju. Det var fort gjort å stille for komplekse spørsmål. Det skjedde også ved noen anledninger at intervjuene ble for ustrukturerte. Å gjennomføre gode kvalitative intervju krever trening. Noe som jeg manglet før jeg begynte dette studiet.

Jeg observerte ulike typer møter. I disse møtene var jeg flue på vegen. Jeg filmet møtene og tok notater. Mitt nærvær, kombinert med at jeg filmet møtene kan ha påvirket deltagerne. Det er vanskelig å si hvor mye jeg påvirket deres oppførsel. Kanskje førte filmingen til at de holdt noe tilbake. Forholdet mellom de ulike deltagerne, og hvordan det påvirket

utviklingsprosessen er den mest interessante observasjonen i dette casestudiet. Jeg tror ikke mine observasjoner av prosjektet har påvirket dette aspektet.

Abstraksjon og generalisering
Hvordan kan man relatere forskningsresultatene til teoretiske og generelle konsepter?

Jeg har kun gjort en casestudie av et bestemt prosjekt. Selv om mange av mine funn er typiske for fagområdet, så kan det være mange faktorer som påvirker det endelige resultatet. Det er derfor ikke hensiktsmessig i å generalisere mine funn. Funnene kan derimot brukes som et utgangspunkt for en større studie.

Jeg prøver i denne oppgaven å se min casestude i forhold til relevante teorier. Blant annet benytter jeg meg av Hirschheim & Kleins paradigme-inndeling av systemutviklingsmetoder, og John Forester's kategorisering av begrenset rasjonalitet.

Veksling mellom teoretisering og lesing av data
Hva er forskjellen mellom de oppfatningene man hadde når man utviklet forskningsdesignet og de man fikk fra de faktiske funnene? Hadde forskeren forutinntatte holdninger? I hvor stor grad passer funnene? Hvor lydhør var man for å endre oppfatning?

Jeg var forberedt på at problemstillingen i denne oppgaven ville endre seg underveis. Dette er et vanlig fenomen i fortolkende feltstudier. Før jeg begynte feltstudiet hadde jeg ikke fokus på organisatoriske aspekter. Når jeg analyserte casestudiet skjønte jeg at de organisatoriske aspektene hadde stor påvirkning på prosjektet som jeg observerte. Dette førte til endringer i mitt fokus.

Flere fortolkninger
Ut fra et sett med data er det mulig å gjøre ulike tolkninger. Det øker kvaliteten å synliggjøre at de ulike tolkningene finnes.

Jeg har gjort noen sammenligninger med andre kvalitative studier, og sett mine funn i lys av ulike teorier. Jeg presenterte også mine funn til de jeg studerte. De var enige i de tolkningene jeg presenterte (se kapittel 5.6)

Mistenkeliggjøring
Det er viktig å være mistenkelig overfor mulig fordreining av virkeligheten. Forskningsobjektet kan ha motiver for å skjule eller gi falske opplysninger. Slike fordreininger kan skje ubevisst. I tillegg til å forstå selve dataene må forskeren kunne lese og tolke det som skjuler seg bak de uttalte ordene.

Jeg var bevisst på dette da jeg jobbet med forskningsdesignet mitt. Jeg bestemte meg tidlig for å benytte både observasjon og intervju. Dette er en vanlig teknikk i kvalitativ studier. Det kan være stor forskjell på hva informanter sier de gjorde og hva man faktisk gjorde eller vil gjøre.

Mulige forbedringer

Dette var min første kvalitative studie og jeg ser muligheter for forbedringer. Det er spesielt et område jeg ville satt mer fokus på hvis jeg skulle gjøre studiet om igjen. Jeg la fort merke til at det var vanskelig å gjennomføre gode intervju. Å gjennomføre gode kvalitative intervju krever trening. Det hadde jeg ikke da jeg startet dette studiet.

8.3 Forslag til videre forskning

Jeg har i denne oppgaven gjort en kvalitativ studie av et bestemt prosjekt. Jeg har ikke forsøkt å generalisere mine funn. Vi kan derimot bruke data fra denne studien for å designe en kvantitativ undersøkelse. Denne vil kunne gi oss svar som kan generaliseres.

En ting vi kan se nærmere på er anbudsprosjekt og brukersentrert design. Er det vanlig at brukermedvirkning og iterativ design forsvinner fra anbudsprosjekt. Hvis så, finnes det unntak? Et annet aspekt er kundens rolle i forhold til utformingen av brukervennlig IT-system. Hvor involverte er kunden, og hvordan forholder de seg til brukersentrert design. En slik studie vil måtte inneholde noe komplekse spørsmål og vil være vanskelig å gjennomføre ved hjelp av vanlige spørreskjema. En mulig utforming kan være strukturerte intervju.

Min studie kan også være et utgangspunkt for nye kvalitative studier. Kundens holdninger til brukersentrert design kan være tema i en slik studie. I min studie hadde jeg fokus på leverandøren av verktøyet. En kvalitativ studie av kunden i et slikt prosjekt kan avdekke nye aspekter. Ved å gjøre en slik studie kan vi finne svar på spørsmål som: i hvilke grad er avgjørelsene som blir tatt politiske, hvordan blir de tatt og i hvilken grad påvirker anbud og økonomi utformingen av prosjektet?

Referanser

Bansler, Jørgen P. & Bødker, Keld (1993): A Reappraisal of Structured Analysis: Design in an Organizational Context. ACM Transactions on Informations Systems, Vol. 11, No. 2. April 1993, Pages 165-193

Beyer, Hugh & Holtzblatt, Karen (1998): Contextual Design, Defining Customer-Centered Systems.

Blomberg, Janette et al. (1997): Back To Work: Renewing Old Agendas for Cooperative Design. Computers and design in context.

Blomquist, Åsa & Arvola Mattias (2002): Personas in Action: Ethnography in an Interaction Design Team. NordiCHI, October 19-23, 2002.

Carroll, John M. (2000): Making Use: Scenario-based design of human-computer interactions. The MIT Press

Carroll, John M. & Go, Kentaro (2004): The Blind Men and The Elephant: Views of Scenario-Based System Design. Interactions / November & Desember

Cart, S. T., Moran, T. P., Newell, A. (1983): The psychology of human-computer interaction.

Caspers, Jones (1995): Patterns of large software systems: Failure and success. Computer, Volume 28, Number 3, March 1995.

Constantine, Larry L. & Lockwood, Lucy A. D.(2001): Structure and Style in Use Cases for User Interface Design. Artikkel som bidrag i Object Modeling And User Interface Design, Designing Interactive Systems

Cooper, Alan (1999), The Inmates Are Running the Asylum: Why High-tech Products Drive Us Crazy and How to Restore the Sanity. Que

Cooper, Alan & Reiman, Robert (2003): About Face 2.0, The Essentials Of Interaction Design. Wiley Publishing, Inc.

Cooper, G. and J. Bowers (1995). Representing the user: Notes on the disciplinary rhetoric of HCI. In Social and Interactional Dimensions of Human-Computer Interfaces. ed. P. Thomas. Cambridge, Cambridge University Press

Cronin, Dave(2003): RUP & Goal-Directed Design: Toward a New Development Process. http://www.cooper.com/content/insights/newsletters/2003_07/RUP_&_GDD.asp

Earthy, J (1998): Usability Maturity Model: Human Centeredness Scale. Version 1.2. INUSE European Usability Support Center

Ehn, Pelle (1993): ”Scandinavian Design: On Participation and Skill” s. 47-78 I D. Schuler og A. Namioka (eds), Participatory Design: Principles and Practices. Hillsdale, N.J., Lawrence Erlbaum Ass.

Eriksen, Anette & Mustaparta, Silja (2003): Brukersentrert systemutvikling i praksis. (Diplomoppgave ved IDI – NTNU)

Forester, John (1989): Planning in the Face of Power. University Of California Press

Grudin, Jonathan (1991): Bridging the Gaps Between Developers and Users. Computer Volume 24 , Issue 4 Special issue on instruction sequencing. Side 59 - 69

Grudin, Jonathan og Poltrock, Steven E. (1994). Organizational Obstacles to Interface Design and Development: Two Participant-ObserverStudies. Technical Report, MCC Human Interface Laboratory.

Grudin, Jonathan & Pruitt , John (2003): Personas: Practice and Theory. Proceedings of the 2003 conference on Designing for user experiences

Grønbæk Kaj, Grudin Jonathan, Bødker Susanne & Bannon Liam (1993): Achieving Cooperative System Design: Shifting from a product to process focus. D. Schuler and A. Namioka, editors, Participatory design: principles and practices, Lawrence Erlbaum Ltd., 1993.

Göransson Bengt, Lif Magnus, Gulliksen Jan (2003): Usability Design—Extending Rational Unified Process with a New Discipline

Hirschheim, R. & Klein, H. (1989): Four Paradigms of Information Systems Development. Communications of the ACM. 32(10), pp.1199-1216.

Hughes John, King Val, Rodden Tom & Andersen Hans (1994): Moving Out from the Control Room: Ethnography in System Design

ISO 13407 (1999): Human-centred design processes for interactive systems.

Jacobson Ivar, Booch Grady, Rumbaugh James (1999): The Unified Software Development Process. Addison Wesley

Kirakowski J. (1994): The Use of Questionnaire Methods for Usability Assessment http://sumi.ucc.ie/sumipapp.html

Klein, Heinz K. & Myers Michael D. (1999): A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Quarterly archive Volume 23 , Issue 1 (March 1999). Special issue on intensive research in information systems. Side: 67 – 93.

Kruchten Philippe (2000): The Rational Unified Process An Introduction, Second Edition. Addison Wesley

Kruchten Philippe, Ahlquist Stefan, Bylund Stefan (2001): User Interface Design in the Rational Unified Process Artikkel som bidrag i Object Modeling And User Interface Design, Designing Interactive Systems

Leffingwell, Dean & Widrig, Don (2001): Managing Software Requierments. A Unified Approach.

Master, David H. (1997): Managing the Professional Service Firm, Free Press

Miler, S.E. (1992): Political implications of participatory design, side 93-100 i M.J. Muller et al. (ed). PDC'92. Proceeding of the Participatory Design Conference. Palo Alto, CA: Computer Professionals for Social Responsibility, 1992.

Mumford, E. (1984): Participation- from Aristotle to today. Side 95-104 i T.M.A Belemans (ed), Beyond Productivity: Information Systems Development for Organizational Effectiveness. Elsevier Science Publishers (North-Holland).

Myers, Brad A. (1994): Challenges of HCI design and implementation. Side 73-83 i ACM Interactions (January 1994).

Nielsen, Jakob (2001): First Rule of Usability? Don't Listen to Users http://www.useit.com/alertbox/20010805.html

Nielsen, Jakob (2003): Usability 101: Introduction to Usability http://www.useit.com/alertbox/20030825.html

Norman, Donald (1998): The Invisible Computer. MIT Press

Robson, Colin (1993): Real world research. Blackwell Publishing

Suchman, Lucy A. (1987): Plans and situated actions: the problem of human-machine communication. Cambridge University Press

Svanæs, Dag og Seland, Gry (2003): Putting the users center stage: role playing and low-fi prototyping enable end users to design mobile systems. In Proceedings of CHI 2004: 479-486, ACM Press.

Taylor, Frederick Winslow (1911): Principles of Scientific Management

Theofanos, Mary (2005): Common Industry Format approved as international standard. Side 46-47 i Interactions (September + October 2005).

Young, Indy (2003): Keep Office Politics Out of Your Design
http://www.adaptivepath.com/publications/essays/archives/000251.php