Masteroppgåve

Hausten 2007

”Fokus på og innhald til ikkje-funksjonelle krav innan systemutvikling”

Oddbjørg Stadheim

Institutt for Informasjons- og medievitenskap

Universitetet i Bergen

Forord

Etter mange månader med arbeid er masteroppgåva ferdig og klar til innlevering. Arbeidet med oppgåva har teke mykje tid, men har vore ein svært interessant periode av studietida mi. Det å få lov til å bruke så mykje tid på å sette seg inn i eit sjølvvalt tema, er eit privilegium. Eg valde å sjå nærmare på ikkje-funksjonelle krav, eit lite omtala omgrep innan systemutvikling og informasjonsvitenskap. I løpet av tida eg har jobba med oppgåva har eg lært mykje om desse krava, og deira plass i teori om systemutvikling. Eg har også fått mykje generell kunnskap som til dømes kor viktig det er å ta seg tid til å vurdere alle sider av ei sak, og at planlegging er nesten halve jobben. Konklusjonen min etter arbeidet er i alle fall klar; ikkje-funksjonelle krav fortenar meir merksemd, både i teori og praksis innan systemutvikling.

Det har likevel ikkje alltid vore like enkelt å leite seg fram i mangfaldet av teoribøker, metodar og meiningar. Då har det vore godt å ha personar rundt som kunne hjelpe meg i rett retning.
Takk til Institutt for informasjons- og medievitenskap for hjelp og svar på alle spørsmål som eg har hatt undervegs.
Ei spesiell takk til rettleiaren min, Jon Iden. Han var til stor hjelp ved spesifisering av oppgåva, og ved val av litteratur til datamateriale og anna støttelitteratur. Det var også godt å få oppmuntrande kommentarar og råd slik at eg kunne kome meg vidare når eg stod fast.
Takk også til studievener for viktige pauser undervegs i skrivinga. Når eg hadde tankar som trong lufting, var det godt å ha nokon å kaste ball med.
I slutten av arbeidet med oppgåva går den største takka til familiemedlemmer som har lese korrektur og kome med gode råd til formuleringar. Spesielt vil eg takke Gunnar som har budd saman med meg gjennom heile denne perioden. Det har nok ikkje alltid vore like kjekt å høyre på gjentekne utgreiingar om ikkje-funksjonelle krav og kvifor dei er så viktige. Sjølv om eg ikkje alltid har vore like takknemleg for råd og rettleiing undervegs, set eg stor pris på det no når oppgåva endeleg er ferdig!

Oddbjørg Stadheim
Hellesylt november 2007

Innhaldsliste

FORORD **1**

INNHALDSLISTE **2**

1. INNLEIING **4**

1.1. Bakgrunn for val av tema 4

1.2. Viktige omgrep 5
1.2.1. Systemutvikling 5
1.2.2. Drift 6
1.2.3. Ikkje-funksjonelle krav 7

1.3. Forskingsspørsmål og forventningar 8

1.4. Val av datamateriale 9

1.5. Oppbygging av oppgåva 10

2. TEORIKAPITTEL **11**

2.1. Systemutviklingsteori og historie 11
2.1.1. 1950- og 1960-talet, dei tidlege systemutviklingsåra 12
2.1.2. 1970-talet, ”code-and-fix”, strukturert systemutvikling og fossefallsmetoden 12
2.1.3. 1980-talet, Objektorientert systemutvikling 13
2.1.4. 1990-talet, RUP (Rational Unified Process) & UML 14
2.1.5. 2000-talet, Agile metodar og ekstremprogrammering (XP) 15

2.2. Krav og kravspesifisering 16

2.3. Ikkje-funksjonelle krav 17
2.3.1. Ulike modellar av ikkje-funksjonelle krav 18

2.4. Drift og ITIL 19

2.5. Utarbeiding av forskingsmodell 21
2.5.1.Definisjon av omgrepa i modellen 24

3. METODEKAPITTEL **29**

3.1. Startfase 29

3.2. Kvalitativ metode 30

3.3. Tekstanalyse 31

3.4. Datamateriale 31

3.5. Teori og analyse 33

3.6. Validitet, reliabilitet og generalisering 34

4. PRESENTASJON AV DATAMATERIALET OG DATAFUNN 36

4.1. Om Systemutviklingsbøkene 36
4.1.1. Theoretical Analysis of Information Systems (1973) 36
4.1.2. The Analysis and Design of Computer Based Information Systems (1985) 38
4.1.3. SSADM version 4 – a Practical Approach (1995) 39
4.1.4. Information Systems Development – Methodologies, Techniques and Tools (2003) 40

4.2. Presentasjon av datafunn 41
4.2.1. Krav (Requirements) 41
4.2.2. Funksjonelle krav (Functional Requirements) 42
4.2.3. Ikkje-funksjonelle krav (Non-functional Requirements) 43
4.2.3.1. Dei ulike omgrepa som høyrer inn under produktkrav 43
4.2.3.2. Organisasjonskrava 'delivery', 'implementation' og 'standards' 48
4.2.3.3. Dei eksterne krava – 'interoperability', 'ethical' og 'legislative' 48
4.2.4. Tabell over ikkje-funksjonelle krav i datamateriale 49

5. DRØFTING 52

5.1. Forventingar og forskingsspørsmål 52

5.2. Vidareutvikling av modell med bakgrunn i datafunn 57

5.3. Eiga vurdering av ikkje-funksjonelle krav i systemutviklinga 60
5.3.1. Kva krav skal ein sjå på? 61
5.3.2. Kven skal vere med under kravspesifisering og utvikling? 62

6. KONKLUSJON 64

LITTERATURLISTE 68

VEDLEGGSLISTE 71

1. Barry Boehm sin modell av kvalitetskrav 72

2. Mylopoulos et. al si liste over ikkje-funksjonelle omgrep 73

3. ISO9126 Quality Model 75

4. Somerville sin modell av ikkje-funksjonelle krav 76

1. Innleiing

Temaet for denne masteroppgåva er ikkje-funksjonelle krav og deira rolle ved utvikling av nye datamaskinbaserte system.
Innleiinga presenterer temaet, viktige omgrep, forskingsspørsmål og forventingar, val av datamateriale og den vidare oppbygginga av oppgåva.

1.1. Bakgrunn for val av tema

Meir enn 40 år etter at IBM lanserte sine første datamaskiner har dei blitt ein viktig og naturleg del av samfunnet. I takt med den teknologiske utviklinga, har ein også sett utviklinga av ulike datamaskinbaserte system. For oss, som brukarar av desse systema, er det viktig at dei fungerer slik som forventa. Når vi betalar rekningane i nettbanken, ønsker vi å være trygge på at transaksjonen er sikker. Vi forventar at pengane går dit dei skal, og at uvedkomande ikkje får tilgang til kontoen vår. Dersom vi skal ta flytoget til Gardermoen, stolar vi på at billettautomaten, og systemet som styrer denne, er tilgjengelig sjølv kva tid på døgnet det er. Sekretæren som skal betale ut lønningar, er også avhenging av at verksemda sitt personalsystem er tilgjengeleg. Dersom noko likevel ikkje fungerer, må det kunne rettast opp utan for store vanskar og utan at det tek for mykje tid.
Ved utvikling av nye datamaskinbaserte system, er fokus oftast på funksjonane som dette systemet skal ha. Eksempel på slike funksjonar kan vere korleis ein transaksjon i nettbanken skal utførast, eller kva felt som skal utfyllast ved registrering av nye tilsette i eit personalsystem. Funksjonane til eit system vert også kalla funksjonelle krav.
Eksempla ovanfor viser at det ikkje berre er dei funksjonelle krava til eit system som er viktige. Sikkerheit, tilgjengelegheit og moglegheit for rask gjennoppretting er alle døme på ikkje-funksjonelle krav. Dette er ein type krav som legg definerar korleis funksjonane til systemet skal utførast. Brukarane er avhengig av at desse krava er implementert som ein eigenskap ved systemet, og utviklarane må difor i stor nok grad legge vekt på desse ved utviklinga. Bakgrunn for val av tema for denne masteroppgåva er difor ønsket om at dei ikkje-funksjonelle krava skal kome meir fram i lyset, og etter kvart bli tillagt større vekt i systemutviklinga.

1.2. Viktige omgrep

Dei neste avsnitta presenterer viktige omgrep knytt til oppgåva sitt forskingstema. Systemutvikling, fordi det er her dei ikkje-funksjonelle krava skal nyttast. Drift, fordi det er her dei ikkje-funksjonelle krava skal oppfyllast. Ikkje-funksjonelle krav, fordi dette er hovudfokuset til oppgåva.

1.2.1. Systemutvikling

Med systemutvikling meinar ein ”utvikling av større informasjonssystemer som skal støtte arbeidsoppgavene i en bestemt bedrift” (Hansen og Hjertø 2003:95). Systemutvikling er altså prosessen med å produsere nye datamaskinbaserte system. Eit informasjonssystem er ”et system for innsamling, bearbeiding, lagring, overføring og presentasjon av informasjon.” (Andersen 1989:14). Christensen, Grønland og Methlie (1994) seier at meininga med eit informasjonssystem (IS) er å halde orden på informasjonsflyt og informasjonstilgang innanfor eit verksemdsområde. Eksempel på slike system er billettbestillingssystem, pasientjournalsystem og rekneskapssystem. I dag har ein også IS som støttar prosessar og interorganisatorisk samhandling.

Systemutvikling som profesjon kan sporast tilbake til 1960-talet. Utviklinga av datateknologien gjorde det mogleg å utvikle system til bruk i ulike verksemder. Dei første systema vart utvikla og implementert utan spesifikke og formaliserte metodar. Dette førte ofte til system som ikkje oppfylte dei forventa krava frå brukarar og verksemder. Etter kvart vart det klart at ein trong betre styring på utviklingsprosessen, og resultatet vart dei første utviklingsmetodologiane basert på the Systems Development Life Cycle (SDLC) som kom på 1970-talet. Dette er ein metode for systemutvikling som består av ulike fasar systemutviklinga skal gå gjennom. Dei fyrste fasane av denne metoden går på analyse og design, dei neste ser på sjølve utforminga og implementeringa av systemet, medan dei siste fasane har med innføring og bruk av systemet å gjere. I løpet av dei fyrste fasane skal ein kome fram til ein kravspesifikasjon, som seier noko om kva krav systemet skal oppfylle. Utviklarane må, i løpet av utviklingsprosessen, identifisere og ta omsyn til krava frå ulike aktørar. Desse krava skal så implementerast som ein del av systemet, og oppfyllast medan systemet er i drift.

Utvikling av datamaskinbaserte system har såleis vore ein del av dataverda i over 40 år. I løpet av denne tida har samfunnet endra seg, og dataverda med den. Ein har fått mindre maskiner med meir kapasitet, og fleire og større bruksområde for maskina. Informasjonsteknologien si stadig endra rolle har stor innverknad på systemutviklingsprofesjonen. I dag dreier systemutvikling seg i mykje større grad om kommunikasjon mellom menneske, og samarbeid mellom menneske og datamaskiner om løysing av arbeidsoppgåver (Skagestein 2005).

1.2.2. Drift

Systemutviklinga konsentrerer seg om det som skjer frå ein kunde tek kontakt med ein utviklar med ønske om eit nytt datasystem, til dette systemet er ferdig utvikla og klar til bruk. Systemet vert så innført i brukarorganisasjonen, og dette er starten på driftsfasen. Drift vil i denne samanhengen tilsvare det engelske omgrepet ”IT Service Management”. Dette omgrepet omfattar tenester som yting, tilgjengelegheit, brukarstøtte, opplæring og vedlikehald (Trienekens, Bouman & Van Der Zwan 2004). Driftspersonell har ansvar for den daglege drifta av systemet, og skal sørgje for at det leverer det som kunden forventar. I dette ligg også eit ansvar for vedlikehald, gjennoppretting og vidareutvikling av verksemda sine datamaskinbaserte system.

Eit område som har fått lite merksemd både i forsking og litteratur, er samarbeidet mellom utviklings- og driftspersonell ved utvikling av nye system. For at eit system skal ha høg tilgjengelegheit frå første dag, må systemet sine driftsmessige behov klargjerast og utformast før systemet vert teke i bruk. Godt samarbeid mellom utviklingspersonell og driftspersonell er avgjerande for at systemet skal tilfredstille brukarane sine behov. Alle dei ulike krava til systemet, både funksjonelle og ikkje-funksjonelle, bør nedfellast i eit dokument. Dette dokumentet vert kalla ein kravspesifikasjon, og gjev retningslinjene for korleis systemet skal oppføre seg. Utarbeidinga av dette dokumentet skal skje i ein tidleg fase av systemutviklingsprosessen, og fungere som grunnlaget for korleis utviklinga skal skje.

Driftspersonellet si oppgåve er å halde, oppnå og oppfylle dei krava som vert utarbeida og avtalt i starten av utviklinga. Driftspersonell har normalt ikkje anna val enn å forhalde seg til det ferdig utvikla systemet som er implementert i verksemda. Dersom dette systemet ikkje stemmer med dei krava som kunden har, vil dei

driftstilsette heller ikkje kunne oppfylle desse gjennom den daglege drifta av systemet.

1.2.3. Ikkje-funksjonelle krav

Bakgrunnen for val av tema for oppgåva er, som nemnt tidlegare, eit ønske om å få meir fokus på ikkje-funksjonelle krav som ein viktig del av utviklinga av nye system. Systemutvikling eit komplekst felt med mange involverte faktorar, både menneskelege og maskinelle. Ein del av utviklaren og kunden sine oppgåver er å forsøke å identifisere alle desse faktorane, slik at ein får eit system som er tilpassa desse på best mogleg måte. Ved starten av eit systemutviklingsprosjekt er det kunden som skal fortelje utviklaren kva slags system dei ønskjer. Kunden har ofte tenkt gjennom på førehand kva dei vil ha, men ønska kan vere uklare og vanskelege å beskrive i ein kravspesifikasjon. Mange systemutviklingsprosjekt endar difor opp med system som ikkje oppfyller krava til brukarane. System som ikkje oppfyller viktige krav skaper problem både i forhold til økonomi og arbeid hos dei som skal bruke det aktuelle systemet. Difor er det viktig at utviklaren brukar god tid på å forstå kva krav kunden forventar systemet skal oppfylle. Krava vil også endre seg undervegs, og ein bør difor gå tilbake til kravspesifikasjonen på ulike stadium av utviklinga.

Dei ikkje-funksjonelle krav er ein del av kravspesifiseringa, og vert nedfelt i eit dokumentet som fungerer som grunnlag for korleis utviklinga skal skje. Ein slik kravspesifikasjon inneheld både funksjonelle og ikkje-funksjonelle krav. Funksjonelle krav seier kva systemet skal gjere, medan dei ikkje-funksjonelle krav seier korleis funksjonane skal utførast. Denne oppgåva skal konsentrere seg om dei ikkje-funksjonelle krava. Oppgåva vil forsøke å klargjere kva som ligg i omgrepet ikkje-funksjonelle krav, og finne ut på kva måte desse krava vert omtala i litteraturen. Dersom ein ikkje tek omsyn til dei ikkje-funksjonelle krava vil det kunne gje feil design, feil utvikling og feil system. Desse krava som skal spesifiserast i starten av ei utvikling skal seinare oppfyllast gjennom drift. Dersom ein ikkje tek omsyn til krava under utviklinga, vil driftspersonell få problem med å oppfylle desse krava seinare når systemet vert innført i verksemda.

1.3. Forskingsspørsmål og forventningar

Ikkje-funksjonelle krav er eit omgrep med uklart innhald og bruksmåte innan systemutviklinga. Temaet kan likevel angripast på mange måtar, og eg har valt å konsentrere meg om systemutviklingslitteraturen sin presentasjon av denne typen krav.

Oppgåva skal ta for seg fire bøker om systemutvikling, og sjå kva desse seier om identifisering og bruk av ikkje-funksjonelle krav. Dette skal skje i ein historisk dimensjon, og bøkene vil difor vere frå fire ulike tiår. For å lettare kunne velje bøker, har eg halde meg til dei som har vore nytta i undervisning ved Institutt for Informasjonsvitenskap ved Universitetet i Bergen. Målet med oppgåva er å få ein betre oversikt over systemutviklingslitteraturen, og korleis denne presenterer og vektlegg ikkje-funksjonelle krav som ein del av ein utviklingsprosess.

Forskingsspørsmålet som oppgåva konsentrerar seg om er:

Har fokuset og innhaldet til ikkje-funksjonelle krav endra seg innan systemutviklingsfaget dei fire siste tiåra?

Med innhald vil ein her meine omgrepsforståinga. Er det andre forhold som kjem inn under omgrepet i dag enn det var for 10, 20 og 30 år sidan? Fokus går på kor mykje dette omgrepet vert omtala i systemutviklingslitteraturen i løpet av 70-, 80-, 90- og 2000-talet. Dette vil verte vurdert ut frå kor mykje plass ikkje-funksjonelle krav har fått i dei ulike bøkene.

I tillegg til ei overordna problemstilling vil oppgåva også sjå på eit par forventingar til datamaterialet. Den første av desse er på mange måtar utgangspunktet for val av tema for oppgåva. Eigen og andre si erfaring med systemutviklingslitteratur og undervisning tilseier at ikkje-funksjonelle krav er eit tema som vert lite omtala. Forventning nummer ein er difor:

Datamaterialet har lite om ikkje-funksjonelle krav.

Kompleksiteten i datastrukturen til informasjonssystem vil likevel tilseie at både innhald og fokus av dette omgrepet har endra seg. I tillegg har IT fått ein viktigare plass i samfunnet, og det har kome nye krav til t.d. sikkerheit på grunn av nye datatilbod som e-handel og bruk av Internet. Vi kan dermed forvente ei utvikling med meir vekt på ikkje-funksjonelle krav. Den andre forventninga er difor:

Det er meir om ikkje-funksjonelle krav i dei nyare bøkene enn dei eldste.

Den siste forventninga kjem også ut frå den generelle endringa innan systemutviklingsteori sidan 70-talet, og det at ein i dag legg meir vekt på dei innleiande fasane av eit systemutviklingsprosjekt som førebuing, analyse, kravspesifisering og design.

1.4. Val av datamateriale

Det har vore ei utvikling innan både datateknologi og systemutviklingsmetodar i løpet av dei fire tiåra oppgåva skal ta for seg. Spørsmålet er om dette viser igjen i forhold til vektlegging av ikkje-funksjonelle krav. Har systemutviklingslitteraturen meir om ikkje-funksjonelle krav i dag enn tidlegare?

For å lettare kunne velje ut teori om systemutvikling, som er eit stort felt, har eg valt å konsentrere meg om litteratur nytta i undervisning. Valet som vert gjort i bruk av bøker reflekterer trendar i tida i forhold til kva teoriar som er rådande på fagfeltet. Universitetet i Bergen har, ved Institutt for Informasjonsvitenskap, hatt kurs i informasjonssystem/systemutvikling sidan tidleg på 1970-talet. Dette gjev dermed ein naturleg avgrensing i val av kva litteratur oppgåva skal sjå på. Eg har valt ut ei bok nytta i undervisninga frå kvart av dei tiåra instituttet har undervist i systemutvikling. Til å velje ut bøkene har eg fått hjelp av professorar som sjølv har undervist i faget opp gjennom åra.

Oppgåva vil sjå på dei ulike bøkene sitt fokus på og innhald om ikkje-funksjonelle krav. Målsetjinga er å sjå om det har vorte større vektlegging på ikkje-funksjonelle krav ved utviklinga av nye informasjonssystem.

Spørsmålet om det ein lærer på universitetskurs har direkte innverknad på korleis ein arbeider når ein kjem ut i arbeidslivet er interessant, men vil ikkje verte diskutert i denne oppgåva.

1.5. Oppbygging av oppgåva

Første kapittelet gjev bakgrunn for val av tema og introduserer forskingsspørsmålet og eit par forventingar til datamaterialet.
Det neste kapittelet presenterer kort den historiske utviklinga innan datateknologi og systemutvikling. Dette for å vise korleis metodar for utvikling av informasjonssystem og bruken av desse systema har endra seg. Deretter presenterer oppgåva teori om kravspesifisering, ikkje-funksjonelle krav og drift. Dette er alle tema som høyrer inn under systemutviklinga og seinare bruk av informasjonssystem. Kapitlet introduserer også forskingsmodellen som skal nyttast ved innsamling av data frå bøkene i datamaterialet. Denne modellen er utarbeida på bakgrunn av teori om ikkje-funksjonelle krav både frå eit systemutviklingsperspektiv og eit driftsperspektiv.
Oppgåva vil vidare sjå på den metoden som har blitt nytta i arbeidet med oppgåva. Metodekapittelet tek føre seg dei forskingsmetodane som er valt for oppgåva. Det neste kapittelet presenterer dei data som er funne i datamaterialet med utgangspunkt i modellen. Desse data vert drøfta på bakgrunn av forskingsspørsmålet og forventingane som oppgåva starta med i kapittel 5. I dette kapittelet vil eg også seie noko om mi vurdering av ikkje-funksjonelle krav og deira rolle i systemutviklinga.
Oppgåva vert avslutta med ein konklusjon der eg oppsummerer funna og arbeidet med oppgåva.

2. Teorikapittel

Dette kapittelet presenterer den teoretiske bakgrunnen som oppgåva kviler på. Ein del av oppgåva sitt forskingsspørsmål er den historiske dimensjonen til systemutviklinga. Første del av teorikapittelet ser difor på den teknologiske, metodiske og forretningsmessige utviklinga til datamaskina og datamaskinbaserte system, frå 1950/60-talet og fram til i dag.
Andre del av teorikapittelet tek for seg teori om ulike tema som heng saman med ikkje-funksjonelle krav. Denne delen startar med teori om krav og kravspesifisering, deretter ser oppgåva på dei ikkje-funksjonelle krava og ulike modellar som tek for seg denne typen krav. Til slutt kjem teori om drift og rammeverket ITIL.
Siste, og også største del av kapitlet, er via utarbeidinga og presentasjon av modellen som skal nyttast under datainnsamling og analysedelen.

2.1. Systemutviklingsteori og historie

Systemutvikling er arbeidet med ”utvikling av større informasjonssystemer som skal støtte arbeidsoppgavene i en bestemt bedrift” (Hansen og Hjertø 2003:95).” Sjølve utviklinga har dermed eit konkret mål – ein skal få eit system som skal hjelpe dei tilsette med å utføre ein eller fleire av sine arbeidsoppgåver. Dette arbeidet med å utvikle eit nytt system ”… er en kompleks og krevende oppgave …” (Hansen og Hjertø 2003:96). Komplekst fordi eit slikt system består av mange ulike delar som, sett saman, skal oppfylle kunden sine forventningar og ønskjer. Krevande fordi eit utviklingsprosjekt ofte tek lang tid, og fordi ein har ei mengd ulike faktorar, både tekniske og menneskelege, som må takast omsyn til.

Når ein skal utvikle eit informasjonssystem kan ein nytte ulike metodar, prosedyrar og teknikkar. Ein metode er ”en detaljert oppskrift for hvordan man skal gå fram når man skal lage eller endre et datamaskinbasert informasjonssystem” (Skagestein 2002:51), medan prosedyrar og teknikkar ”viser detaljert fremgangsmåte for en oppgave …” (Hansen og Hjertø 2003:145).
Dei fleste systemutviklingsmetodane har sitt utgangspunkt frå System Development Life Cycle (SDLC). Dette er ein metode for systemutvikling som består av ulike fasar systemutviklinga skal gå gjennom. I løpet av dei nærare 60 åra informasjonsteknologien har vore tilgjengeleg, har ein sett stor vekst og endring.

Dette gjeld både for sjølve teknologien, systemutviklingsprosessen og metodar for denne, samt bruken av informasjonssystem i arbeidslivet og samfunnet elles. Dei neste avsnitta presenterer difor eit kort samandrag av den historiske utviklinga til informasjonsteknologien.

2.1.1. 1950- og 1960-talet, dei tidlege systemutviklingsåra

I 1951 introduserte J. Presper Eckert & John Mauchly UNIVAC I (UNIVersal Automatic Computer). Etter kvart kom også IBM med sine fyrste datamaskiner. Maskinene vart programerte ved hjelp av assembly, også kalla systemkode. På midten av 50-talet fekk ein høgnivå programmeringsspråk med tilhøyrande kompilatorar (Skagestein 2005). Datamaskinene vart også meir avanserte og programmeringsspråk som COBOL, PL/1 og Fortran. gjorde programmeringa og tilgangen til data enklare.

Det var få, eller ingen, som hadde spesifikk kompetanse når det gjaldt utvikling av datamaskinbaserte system. Utviklinga skjedde difor i to fasar. Først vart ein liten del av systemet programmert. Deretter vart koden endra for å rette opp feil, endre eller tilføre ny funksjonalitet (Hansen og Hjertø 2003). Denne metoden å drive systemutvikling på har fått namnet ”code-and-fix” (Boehm 2006).
Arbeidet med utvikling av dei første datamaskinbaserte systema var vanskeleg og tok tid. Likevel såg ein at det i datateknologien låg eit stort potensial for vidare utvikling og bruk.

2.1.2. 1970-talet, ”code-and-fix”, strukturert systemutvikling og fossefallsmetoden

Dei tidlegaste dataåra var prega av at informasjonsteknologi og datamaskiner var noko nytt. Fram til midten av 1970-tallet såg ein på IT (eller EDB) som eit reint rasjonaliseringsverktøy. Siktemålet med IT-bruk var eine og aleine kostnadseffektivisering – ’efficiency’. Kostnadseffektiviseringsperspektivet var oppteke av å automatisere eksisterande oppgåver i organisasjonen. Formålet var å oppnå forbetra produkt og å minske kostnader (Christensen, Grønland og Methlie 1994).
Utviklinga av nye system skjedde etter ”code-and-fix” metoden (sjå avsnittet ovanfor). Denne metoden gav ein utviklingsprosess som var lite forutsigbar og vanskeleg å kontrollere. Denne måten å drive utvikling på førte også til system som vart levert for seint, var dyre og fulle av manglar (Hansen og Hjertø 2003). Etter kvart

vart det klart at ein trengde betre organiserte metodar for å gjennomføre dei store utviklingsprosjekta som dukka opp. Ein såg dermed ei oppblomstring av ulike systemutviklingsmetodar, det vil seie ”oppskrifter” på korleis ein systemutviklingsprosess skal utførast, og ei bølgje av strukturert systemutvikling på midten av 1970-talet (Boehm 2006).

Dei første og allment brukte metodane innan strukturert systemutvikling var fossefallsmetodane. Desse metodane var godt representert i lærebøker og i kommersielle systemutviklingsmetodar på 70-talet (Skagestad 2002). Det eksisterer mange ulike variantar av fossefallsmetoden, men alle består av ein lineær sekvens av fasar. Ein fase skal ferdigstillast før ein startar på den neste (Whiteley 2004). Grunnfasane i metoden er analyse, utforming og realisering (Skagestad 2002). Fossefallsmetoden er lite fleksibel i det at det ikkje er mogleg å gå tilbake til tidlegare fasar for å utføre endringar. Bruk av denne metoden gjer det dermed vanskeleg å tilpasse seg kunden sine skiftande krav. Metoden er, i sin idealiserte form, lite brukt i dag. Den fungerar likevel som eit utgangspunkt for andre metodar, mellom anna SSADM (Structured Systems Analysis and Design Method) som kom på 1980-talet (Whiteley 2004).

Fossefallsmetoden er del av den strukturerte systemutviklinga, ein teori som skulle sørgje for at ein fekk orden på systemutviklingsprosessen og sikre betre system. Mot slutten av dette tiåret såg ein likevel at det oppstod problem som følgje av den formelle og sekvensielle utviklinga. Det vart mellom anna brukt for lite tid på kravspesifisering og design, sidan ein såg på programmering som den viktigaste delen (Boehm 2006). Dette førte til andre metodar og form for utvikling på 1980-talet.

2.1.3. 1980-talet, Objektorientert systemutvikling

I dette tiåret vart minicomputeren introdusert av IBM, ei nyheit som vart overgått av Apple sin introduksjon av microcomputeren. IBM svarte og kom med PC (Personal Computer) tidleg på 80-talet.
I tillegg kom det fleire ulike programvareverktøy for utvikling. Eksempel på desse er CASE-verktøy (Computer Aided Software Engineering)(Boehm 2006). Ein såg også introduksjonen av Database Management system som Oracle og Ingres.

Programmeringa vart gjort lettare med introduksjon av fjerde generasjons programmeringsspråk som Java og C++.

Det vart teke mange ulike initiativ for å prøve å løyse dei problema ein såg i det førre tiåret. Introduksjonen av objektorienterte metodar førte til ein objektorientert systemutvikling (Boehm 2006). Denne presenterte ein meir evolusjonær måte å drive utvikling på, og vart etterkvart den naturlege måten å designe datasystem (Whiteley 2004). Objektorientert utvikling gjorde arbeidet mindre statisk. Det vart mogleg å gå tilbake til tidlegare fasar og endre på systemdelane dersom det kom nye krav og ønske frå kunden under arbeidet eller etter at utviklingsarbeidet var ferdig. Utviklinga var likevel ikkje berre positiv. Ein gjekk frå utvikling av store system til utvikling av mindre systemdelar. Dette førte til lappesystem som ikkje alltid passa saman, og datapersonell som måtte drive brannslokking i større grad enn utvikling.
Utover på 1970-tallet og vidare over i 80-talet starta ein å interessere seg for informasjon som ein ressurs i avgjersle- og styringssamanheng. Siktemålet med bruk av IT vart utvida til også å omfatte måleffektivisering – 'effectiveness'. Dette perspektivet er retta mot informatisering og støtte for individuelle saksbehandlarar og dei som skal ta ulike avgjersler. Formålet var å legge grunnlaget for betre avgjersler, meir effektiv kommunikasjon og gjennom dette betre styring. Omgrep som avgjerslestøtte og leiingsinformasjon fekk ekstra merksemd då PC-bølgja kom for fullt tidlig på 1980-tallet (Christensen, Grønland og Methlie 1994).

2.1.4. 1990-talet, RUP (Rational Unified Process) & UML

Dette tiåret såg ei endring i fokus frå det å sjå på maskina som ein artefakt, til å fokusere på behandling og utveksling av informasjon. Utover 1990-talet vart nettverk viktigare og viktigare. Ein fekk lokale nettverk mellom arbeidarar i ein og same organisasjon, og større nettverk som kunne gå mellom ulike kontorbygningar og etter kvart også mellom ulike land. Det vart lettare og kommunisere mellom maskiner og over store avstandar. Ein fekk databehandling med klient/tenar, og Internett vart teke i bruk av vanlege folk og bedrifter. Java programmering og Html letta programmeringa av internettsider og nye datasystem. Ein såg også ei oppblomstring i "Open Source" programvareutvikling. Dette gav mellom anna plattformar og programmeringsspråk som Linux, Apache, Python og Mozilla.

Som på 80-talet var den evolusjonære utviklinga den rådande metoden også dette tiåret, og kom i form av ulike modellar. Ein av desse er RUP, Rational Unified Process. Dette er eit prosessrammeverk for systemutvikling, utvikla av Rational Software, og har sin bakgrunn i Barry Boehm sin spiralmodell for utvikling. Modellen er basert på Unified Process metoden, som er eit sett av utviklingsteknikkar og beste praksis. Hovudpoenga i denne metoden er iterativ utvikling av programvare, krav til styring, komponentbasert arkitektur, visuell modellering av programvare, krav til kvalitet og kontroll av endringar av programvare. Metoden nyttar iterativ og inkrementell utvikling. Inkrementell, eller trinnvis utvikling, er ei blanding av strukturert og evolusjonær systemutvikling. Denne metoden tek meir omsyn til at analyse, utforming og realisering påverkar kvarandre gjensidig. Den er basert på ein strategi med delleveransar. Ein har ei felles analyse og utformingsfase for heile systemet, og deretter vert kvart delsystem utvikla (Skagestein 2005). Unified Modelling Language (UML) er ein teknikk som har vorte nytta til å modellere korleis systemet skal sjå ut. Her tek ein med dei ulike objekta og delane av systemet, samt kommunikasjonen mellom desse delane.
Midt på 1980-tallet retta ein merksemda mot IT som strategisk virkemiddel. Siktemålet med bruken av IT vart dermed ytterlegare utvida til også å omfatte konkurranseeffektivitet – 'competitiveness'. Konkurranseeffektivisering dreier seg om strategisk marknadsorientering, innovasjon og nyskapning, og organisatorisk transformasjon. Formålet er mangfaldig, men omfattar mellom anna tilpassing til omgivnadene og skaping av konkurransefortrinn. IT er i dag blitt "et generelt verktøy for effektivisering, styring og fleksibel tilpassing til konkurransemessige forutsetninger i omgivelsene" (Christensen, Grønland og Methlie 1994:25).
I løpet av 90-talet vart det også lagt meir vekt på brukarvenlegheit, og forsking innan HCI. I tillegg vart brukaren ein større del av sjølve utviklinga (Boehm 2006).

2.1.5. 2000-talet, Agile metodar og ekstremprogrammering (XP)

I løpet av dei fyrste åra av dette tiåret har ein sett raskare og mindre maskiner med betre kapasitet. Internett er teke i bruk i stadig aukande grad, og store nettverk gjer kommunikasjon og deling av data enklare. I dag er det fokus på det å kunne effektivisere og gjere raskare henting av data, lagring av data og bruk av denne informasjonen i verksemder.

Ein har fortsett trenden mot raskare utvikling av applikasjonar. Aukande frustrasjon med tung planlegging, spesifikasjonar og anna dokument har gitt seg utslag i introduksjonen av ulike agile metodar (Boehm 2006). Agile metodar er ein moderne metode der det vert lagt vekt på å kome fram til køyrbare system så raskt som mogleg. I dette ligg det at ein berre utfører det som er høgst nødvendig av analyse og utformingsaktivitetar. Systema vert utforma slik at dei er lette å tilpasse og endre undervegs (Skagestein 2005)
Ekstrem Programmering (XP) høyrer til under dei agile metodane, og er ei ny tilnærming til utvikling basert på utvikling og leveranse av svært små delar. Ein har korte utviklingsinkrement, brukarmedverknad i utviklingsteam, enkel design, parprogrammering og kontinuerleg integrering (Boehm 2006).

2.2. Krav og kravspesifisering

Ved utvikling av eit system tek ein utgangspunkt i dei overordna tekniske og økonomiske krava til det framtidige systemet (Skagestein 2005).

IEEE standard 729 for Software Engineering(IEEE83) definerar krav som:

1) vilkår eller eigenskap som ein brukar treng for å løyse eit problem eller oppnå noko.
2) vilkår eller eigenskap som må møtast eller behandlast av eit system…for å tilfredstille ei kontrakt, ein standard, spesifikasjon eller andre formelle dokument som ligg til grunn

(Davis 1993:15, mi oversetting)

ISO 9000:2000 definerer krav som ”…behov eller forventning som er angitt, vanligvis underforståtte eller obligatoriske” (Hansen og Hjertø 2003:31). Hansen og Hjertø seier også at krav er ”et utsagn som beskriver funksjonelle egenskaper, ytelsesparametre og andre egenskaper ved det ønskede produktet.” (2003:33). Brukaren og andre som har interesse i systemet, har ulike forventingar og ønskjer til systemet. Krava vert den formelle beskrivinga av desse forventingane og ønska (Hansen og Hjertø 2003).

Krav til eit nytt informasjonssystem kan igjen delast inn i to hovudgrupper, funksjonelle krav og ikkje-funksjonelle krav (Gurholt og Hasle 2003). Dei funksjonelle krava beskriv kva systemet skal gjere, korleis det skal reagere til ulike output/input og oppføre seg i ulike situasjonar. Dei ikkje-funksjonelle krava kjem med avgrensingar på funksjonane, og gjeld for heile systemet under eitt (Sommerville 2004).

Dei funksjonelle og ikkje-funksjonelle krava til eit system vil som oftast vere beskrive i eit dokument kalla ein kravspesifikasjon (Skagestein 2005). ”Kravspesifikasjonen er et dokument som gir en samlet beskrivelse av de kravene brukerne stiller til det framtidige informasjonssystemet” (Andersen 1989:37). Dette dokumentet er også ein del av ein avtalen mellom kunden og utviklaren om korleis systemutviklinga skal utførast (Skagestein 2005).

Det å utforme ein kravspesifikasjon er meir enn berre å lage ei liste med krav. Ein skal avgjere kva systemet skal kunne gjere, korleis det skal utføre desse funksjonane og korleis det skal sjå ut. Kravspesifikasjonen er dokumentet som samlar desse beskrivingane, og dermed fungerar som utgangspunktet for resten av utviklingsprosessen. Andersen seier det på denne måten ”kravspesifikasjonen er bindeleddet mellom analysefasen og utformingsfasen” (1989:36).

Kravspesifikasjonen tek for seg alle dei ulike krava som systemet skal oppfylle. Ovanfor vart både dei funksjonelle og ikkje-funksjonelle krava definert. Sidan temaet til denne oppgåva er dei ikkje-funksjonelle krava, vert berre desse presentert nærmare.

2.3. Ikkje-funksjonelle krav

Ei utfordring med teori omkring ikkje-funksjonelle krav er at det finst mange ulike definisjonar på kva som vert omfatta av ikkje-funksjonelle krav. Det er eit stort mangfald av typar ikkje-funksjonelle krav, noko som kan vere med på å skape vanskar. I tillegg nyttar ulike forfattarar ulike namn på det som er det same omgrepet. Der Davis (1993) snakkar om ’non-behavioral requirements’ og Skagestein (2005) om ’kvalitetsmessige krav’, kan ein også bruke omgrepet ikkje-funksjonelle krav. Ein er likevel einige om at ein fellesnemnar for denne typen krav er at dei er med på å legge

avgrensingar på funksjonane, og gjev positive (oppetid, responstid, kapasitet) eller negative (kven som skal ha tilgang, kva ein ikkje kan bruke systemet til) grenser som driftspersonellet skal halde orden på.

Usemja innan teorien viser at det er uendeleg mange måtar ein kan klassifisere og gruppere ikkje-funksjonelle krav på. Dette gjer det vanskeleg å finne ein definisjon som er omfattande og klar nok til å fungere som grunnlag for oppgåva si tekstanalyse. Løysinga har difor vore å lage ein eigen modell. Denne skal lagast på bakgrunn av ikkje-funksjonelle krav henta frå både systemutvikling og drift. Avsnitta nedanfor vil difor først presentere ulike andre modellar av ikkje-funksjonelle krav, og deretter teori om drift der ein også ser på dei ikkje-funksjonelle krava.

2.3.1. Ulike modellar av ikkje-funksjonelle krav

I ein modell frå 1978 (sjå vedlegg 1) presenterar Barry Boehm sine krav til kvalitet. Her definerar ein kvalitet til eit system ved hjelp av ulike atributtar. Mange av desse tilsvarar omgrep som denne oppgåva definerer under termen ’ikkje-funksjonelle krav’. Eksempel på dette er ’maintainability’ og ’reliability’. Eit av hovudpunkta til Boehm er at ein skal legge vekt på dei faktorane som er viktige for den spesifikke applikasjonen.

Mylopoulos et. al presenterar i si bok, *Non-Functional Requirements in Software Engineering* (2000), ei liste over omgrep som vert omfatta av ikkje-funksjonelle krav. Boka presenterer ein måte å representere og analysere ikkje-funksjonelle krav ved hjelp av eit rammeverk som skal gjere det lettare for utviklarar å forhalde seg til ikkje-funksjonelle krav. Lista med ikkje-funksjonelle krav er utarbeida for å gje lesaren ei kjensle av den breidda av områder som kjem inn under dei ikkje-funksjonelle krava. Denne lista er på langt nær fullstendig. Ein må vurdere frå situasjon til situasjon kva krav som skal vere med i ein utviklingsprosess. Lista gjev likevel eit godt utgangspunkt, og dekkjer viktige ikkje-funksjonelle krav som mellom anna sikkerheit, brukarvenlegheit og tilgjengelegheit. Lista er lagt ved oppgåva som vedlegg 2.

ISO9126 har utvikla ein kvalitetsmodell (sjå vedlegg 3) for vurdering av systemprodukt. Hovudkategoriane her er ‘portability’, ‘functionality’, ‘reliability’, ‘maintainability’, ‘efficiency’ og ‘usability’. Fire av desse kategoriane er del av dei

ikkje-funksjonelle krava, medan dei andre to (’functionality’ og ’usability’) er eigne typar krav.

Ian Sommerville har i si bok Software Engineering (2004, 7.utg) ein annan modell av systemeigenskapar (sjå vedlegg 4). Denne deler dei ikkje-funksjonelle krava inn i tre hovudgrupper; produktkrav, organisasjonskrav og eksterne krav, som igjen vert delt inn i mindre undergrupper. Produktkrava er dei krav som spesifiserar måten produktet oppfører seg på. Eksempel på dette inkluderer ytingskrav, som kor raskt systemet må utføre oppgåver og kor mykje minne dette krev, krav til kor mykje feil som kan aksepterast, og krav til brukarvenlegheita. Organisasjonskrav kjem frå retningslinjer og prosedyrar i kunden og utviklaren sin organisasjon. Eksempel på dette er kva standardar som skal nyttast, avgrensingar på implementeringa av systemet i forhold til bruk av programmeringsspråk og systemdesign, og krav til når systemet og dokumentasjonen skal leverast. Eksterne krav omfattar alle dei krav som kjem frå faktorar utanfor systemet og utviklingsprosessen. Dette er krav om korleis systemet skal samarbeide med system i andre organisasjonar, lovpålagde krav som må følgjast eller etiske krav.

Oppgåva har tidlegare snakka om korleis dei ikkje-funksjonelle krava er med på å legge retningslinjene for det arbeidet som driftspersonell skal utføre etter at systemet er teke i bruk. Det vil difor vere naturleg at ein også tek omsyn til kva drift legg i dette omgrepet. Dersom det viser seg at innhaldet til omgrepet ikkje er identisk, talar dette for å utarbeide ein forskingsmodell der begge sidene sine ikkje-funksjonelle krav vert teke med.

2.4. Drift og ITIL

I første kapittel snakka oppgåva om forholdet mellom drift og systemutvikling, og at det å identifisere og legge vekt på ikkje-funksjonelle krav er viktig for at ein skal kunne få ein god drift av systemet. Oppgåva har definert drift som det arbeidet som skal til for å halde eit system i gang. ITIL er eit rammeverk som seier noko om beste praksis for arbeidet med drift (van Bon et al. 2004)**.** Eit rammeverk er eit sett med retningslinjer og råd som ein kan nytte og tilpasse for å forbetre sin eigen driftskvardag. Med beste praksis snakkar ein her om arbeidsmåtar som er velprøvd i

industrien og som gjev gode resultat (Hansen og Hjertø 2003). Beste praksis tyder ikkje at dette er den einaste måten å utføre noko på, men er eit forslag til korleis noko kan gjerast. Det er så opp til kvar enkelt å plukke og tilpasse desse måtane å arbeide på til sin eigen organisasjon.
ITIL-rammeverket er delt inn i fleire ulike bøker, der kva av desse tek for seg ei oppgåve innan drift. Her ser ein både på det som skal skje direkte i forhold til systemet (Service Management), på forholdet til forretningsdelen (The Business Perspective), forholdet til teknologien (ICT Infrastructure Management), på sikkerheit (Security Management) og på forholdet mellom systemutvikling og drift (Application Management).

I forhold til denne oppgåva er det den sistnemnde boka som er interessant. ITIL Application Management seier at eit viktig tema som har eksistert ei stund, er problemet med å få utviklarar av applikasjonar og driftsdelen til å samarbeide tettare (Hertroys et. al 2002). Historisk sett har ein delt handtering og bruk av datamaskiner inn i tre delar; forretningsmessig, utvikling og drift. ITIL Application Management meiner at dette er noko ein bør revurdere. For å unngå problem ved ferdige applikasjonar, må dei som utviklar og dei som driftar system og applikasjonar samarbeide.

Ein anna tema innan ITIL er Service Level Aggreements (SLA). Dette er avtalar som skal gjerast med kunden for å fastlegge ønska servicenivå. Drift har så ansvar for at systemet skal oppfylle krava i denne avtala. Ifølge ITIL skal ein SLA omfatte tilgjengelegheit, avtalt oppetid, behandling av incidents, sikkerheit (hacking, integritet og konfidensialitet), vedlikehald (når og kor ofte), endringshandtering, katastrofeløysingar og økonomi. Det er vanleg at det er servicepersonellet til verksemda som tilbyr tenesta, som utarbeidar SLA saman med kunden.

ITIL rammeverket ser på systemet frå eit driftssynspunkt. Ein har eit system som er ferdig utvikla, og no skal dette brukast på ein best mogleg måte i forhold til kva kunden ønskjer. Rammeverket er med på å organisere metodisk korleis ein skal få det beste ut av eit system, og korleis driftspersonellet skal organiserast i forhold til systemet og til kvarandre.

Application Management deler krav til systemet inn i funksjonelle krav, ikkje-funksjonelle krav, og krav til 'usability'. Dette til skilnad frå Sommerville som har 'usability' med som ein underkategori til dei ikkje-funksjonelle krava.
Ikkje-funksjonelle krav, sett frå eit IT Service Management perspektiv, ser på nødvendigheita for ei responderande, tilgjengeleg og sikker teneste, og tek for seg tema som utrulling, operasjonar, systemstyring og sikkerheit. Det å designe for ikkje-funksjonelle krav betyr at ein gjev desse krava ei like stor grad av viktigheit som dei funksjonelle krava, og inkluderar dei som ein påkravd del av designfasen. Ikkje-funksjonelle krav bør også testast på lik linje med dei funksjonelle krava.

2.5. Utarbeiding av forskingsmodell

For å kunne utføre ei undersøking av det utvalde datamateriellet, må ein ta utgangspunkt i teori om emnet. I dette tilfellet skal oppgåva bruke teori om systemutvikling, drift og ikkje-funksjonelle krav til å hente ut dei data frå datamaterialet som seinare skal analyserast og drøftast. Fokus er på identifisering og bruk av ikkje-funksjonelle krav. I mangel på klare definisjonar av kva som høyrer inn under ikkje-funksjonelle krav, vert ein del av oppgåva å utarbeide ein modell. Denne tek utgangspunkt i Sommerville (2004) sin modell av ikkje-funksjonelle krav, og modifiserar den med ikkje-funksjonelle krav frå ITIL sin Application Management (Hertroys et. al 2002).

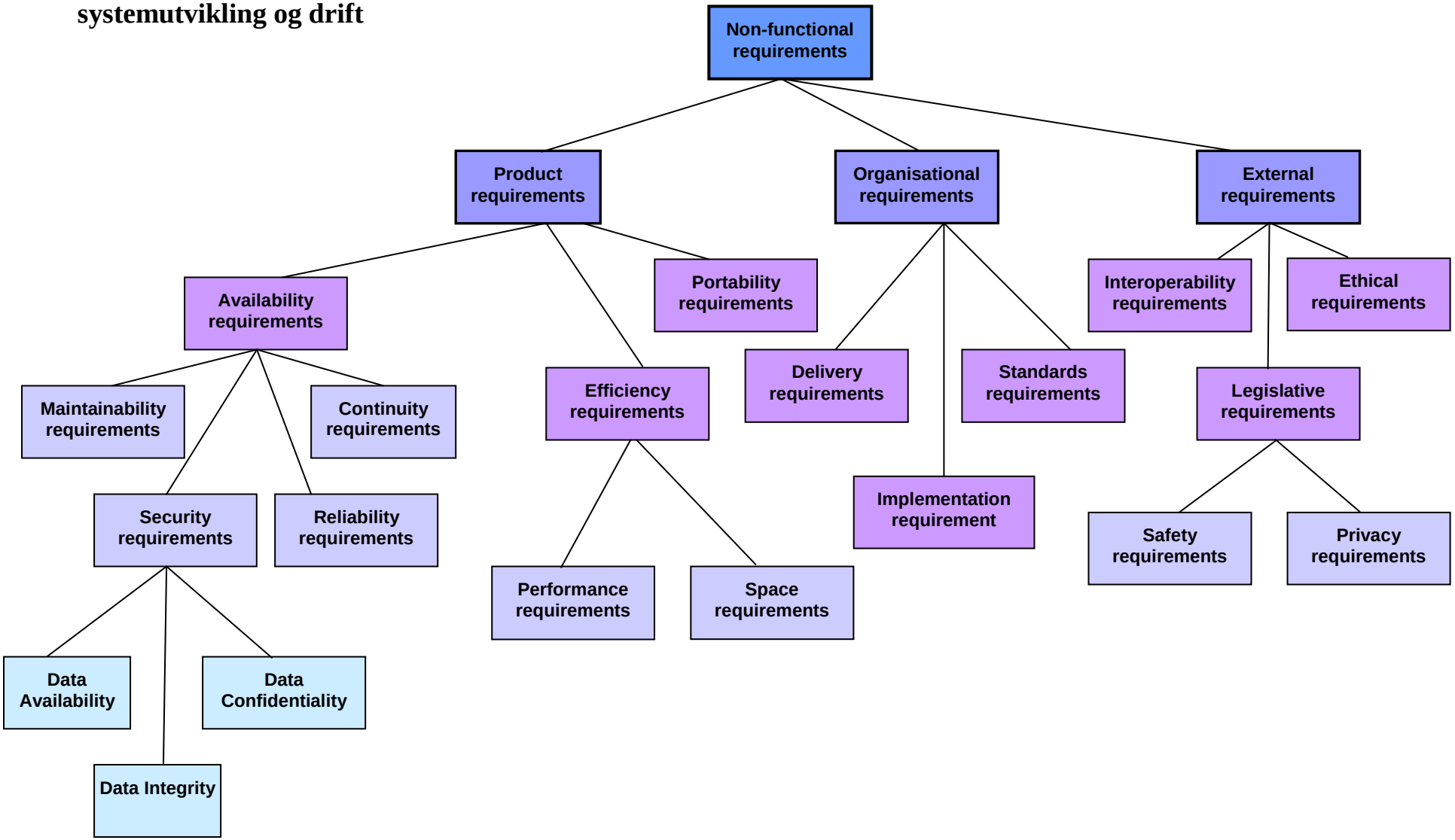
Sommerville sin modell av ikkje-funksjonelle krav er presentert i ei bok der ein tek for seg systemutvikling sett frå ein utviklar sitt synspunkt. Slik sett ser ein også på dei ikkje-funksjonelle krav frå systemutviklarane sitt synspunkt.
Sidan noko av meininga med oppgåva er å sjå på forholdet mellom utvikling og drift når det gjeld ikkje-funksjonelle krav, vert det viktig med ein modell som kan fange opp dei krav som vert lagt til grunn på begge sider. ITIL Application Management (Hertroys et. al. 2002) er ei bok som tek føre seg eit driftssynspunkt på systemutvikling, og dermed også viktige ikkje-funksjonelle krav for drift.

Med bakgrunn i dei skilnadene eg har funne i Application Management, har eg utvida Sommerville sin modell med omsyn på ikkje-funksjonelle krav frå drift til systemutvikling.

Modellen tek utgangspunkt i Sommerville si tredeling av ikkje-funksjonelle krav. ’Availability’ er sett inn som ein underkategori til ’product requirements’, og ’reliability’ er lagt inn under denne. I tillegg er ‘maintainability’, ‘security’ og ‘continuity’ lagt inn under ‘availability’. I Application Management er ’usability’ ein eigen type krav. Eg har difor valt å ta denne typen krav ut av modellen. Resten av Sommerville sin modell er uforandra.

Modellen har ei overvekt av typar ikkje-funksjonelle krav som har med produktet, dvs systemet å gjere. Etter mi meining er dette er ein naturleg fordeling då dette er dei ikkje-funksjonelle krava som ofte skaper mest problem og treng mest vektlegging. Dette samsvarar også med ITIL sitt driftsperspektiv som legg større vekt på krav til systemet og bruken av det, og at utviklinga og bruken av systemet skjer i forhold til organisasjonen og deira forretningsverksemd sine krav.

Modell av ikkje-funksjonelle krav i systemutvikling og drift



2.5.1.Definisjon av omgrepa i modellen

For å få ei betre forståing av modellen, vert dei ulike omgrepa i modellen presentert nedanfor. Alle desse omgrepa høyrer inn under termen ’ikkje-funksjonelle krav’, og er med på å legge føringar på korleis eit system skal utføre sine funksjonar. Desse føringane kan gi positive (oppetid, responstid, kapasitet) eller negative (kven som skal ha tilgang, kva ein ikkje kan bruke systemet til) grenser som driftspersonellet skal halde orden på. Ikkje-funksjonelle krav har innverknad på systemet som eit heile, og ikkje berre til dei enkelte funksjonane som systemet skal utføre.

Product requirements vert på norsk der same som produktkrav. Sommerville (2004) seier at dette er krav som spesifiserar måten produktet oppfører seg på. Det omfattar dermed alle dei ulike ikkje-funksjonelle krava som går på systemet som eit produkt som skal brukast i ei verksemd eller ein organisasjon.

Availability requirements er krav til systemet sin tilgjengelegheit. I følgje ITIL sin glossary (Evans et. al. 2001) er dette ei IT-teneste sin evne til å utføre sine kravde funksjonar innanfor eit avtalt tidspunkt eller tidsperiode.
Tilgjengelegheit vert ofte beskrive i prosent, det vil seie den prosentvise tida tenesta faktisk er tilgjengeleg for brukarane innanfor den avtalte tenestetida (Evans et. al. 2001). IEEE siere at ’availablity’ er den grad eit system eller ein komponent er operasjonell og tilgjengeleg når kravd (IEEE90).
Availability er eit samleomgrep som sjølv omfattar fleire ulike ikkje-funksjonelle krav.

Maintainability requirements er krav til vedlikehald og oppretthalding av systemet. ITIL sin glossary seier at dette er ein komponent eller IT teneste sin evne til å oppretthalde, eller bli reparert til ein tilstand der den kan utføre sine kravde funksjonar. Det omfattar også krav til vedlikehald, og dei avgrensingar, prosedyrar og ressursar ein har ved utføring av dette (Evans et. al. 2001).

Reliability requirements kan på norsk omsettast til krav til pålitelegheit. IEEE seier at dette er ein komponent eller IT teneste si evne til å utføre den kravde funksjonen under avtalte betingelsar for ei avtalt

tidsperiode (IEEE90). Denne beskrivinga av omgrepet liknar mykje på ’availablity’, men her er det meir vekt på det at ein skal kunne stole på at systemet er til dømes tilgjengeleg til avtalte tidspunkt.

Security requirements er krav til sikkerheit, og då meir spesifikt dei data som informasjonssystemet inneheld, behandlar og leverar. ITIL glossary seier at dette er prosessen med å sikre at tenester vert brukt på riktig måte, og av dei riktige personane. Ein skal kunne stole på at dei data ein legg inn i, eller hentar ut av systemet er tilgjengelege, har integritet og er konfidensielle (Evans et. al. 2001). Dermed kan datasikkerheita delas opp i tre mindre delar.

Data Availability handlar om tilgjengeligheit på data. ITIL seier at dette handlar om krav til at autoriserte brukarar skal kunne ha tilgang til dei data dei treng, når og kvar dei trenger det (Hertroys et. al. 2002).

Data Integrity kan omsettast til dataintegritet. I dette ligg det at alle autoriserte brukarar skal kunne føle seg sikker på at dei data som vert presentert til dei er riktige (Hertroys et. al. 2002).

Data Confidentiality handlar om datafortrulegheit. I til Application Management seier at dette går ut på at ingen skal kunne sjå og bruke data i systemet utan at dei er autoriserte til det.

Continuity requirements tek for seg krav til kontinuitet av tenesta og dei data som systemet tilbyr. Dette er krav til kva som skjer ved feil på komponentar/systemet. I dette ligg det krav til gjenoppretting av data, det å ta backup og planlegging om korleis ein skal få systemet raskt opp å gå igjen.

Efficiency requirements er krav til systemet sin effektivitet. IEEE seier at dette er mål for å avgjere om eit system eller komponent utfører sine tildelte

funksjonar med minst mogleg bruk av ulike ressursar (som mellom anna CPU, Minne, I/O – einingar og Netverk) (IEEE90). Effektivitet kan igjen delast opp i to typar ikkje-funksjonelle krav.

Performance requirements omfattar krav til korleis systemet skal utføre sine oppgåver. Her er det snakk om systemet sine funksjonar, og om det går raskt nok og om kunden/brukaren kan utføre dei operasjonane ein treng.

Space requirements omhandlar krav til systemet sin kapasitet. Her er det snakk om kapasiteten til systemet. Har ein nok lagringsplass? Er komponentane i stand til å handtere den mengde data som går gjennom på ein dag?

Portability requirements kan omsettast som krav til systemet sin fleksibilitet. Dette handlar om moglegheita til å flytte eit system eller ein komponent frå ein type omgivnader (hardware eller software) til ein annan (IEEE90).

Den andre hovudgruppa ikkje-funksjonelle krav er dei organisasjonsmessige krava – 'organisational requirements'. Dette er krav som kjem frå retningslinjer og prosedyrar i kunden og utviklaren sin organisasjon (Sommerville 2004). Medan produktkrava ofte kan vere vanskelege å spesifisere, er dei organisasjonsmessige krava på mange måtar lettare å få ned på papiret. Krav til levering kan tidfestast, implementeringa vert eit val mellom ulike alternativ til språk og plattform, og bruk av ulike standardar vil allereie vere fastlagt i både utviklaren og kunden sin organisasjon.

Delivery requirements tek for seg krav til leveringa av det nye systemet. Dette omfattar tema som når systemet skal leverast, kvar dette skal skje og korleis det skal skje. Desse krava kan kome både frå kunden og utviklaren.

Implementation requirement omfattar ulike krav til sjølve implementeringa av systemet. Dette omfattar å svare på spørsmål som kva programmeringsspråk ein skal nytte, kva plattform systemet skal vere på, kva type komponentar skal ein nytte, med meir. Også her kan krava kome frå både kunden og utviklaren.

Standards requirements ser på krav til bruk av ulike standardar. Her er det snakk om standardar som anten kunden eller utviklaren sin organisasjon nyttar. Dette kan vere kvalitetsstandardar, behandling av informasjon, måtar å arbeide på, med meir.

Ekstene krav – 'external requirements', er den tredje og siste hovudgruppa av ikkje-funksjonelle krav. Sommerville (2004) seier dette omfattar alle dei krav som kjem frå faktorar utanfor systemet og utviklingsprosessen . Eit system utfører ikkje sine funksjonar i eit vakuum, men i samhandling med menneska, organisasjonen og samfunnet som det er ein del av. Dette gjer at det, ved utvikling av nye system, også vil kome krav frå omgivnadene til korleis systemet skal oppføre seg.

Interoperability requirements tek for seg krav til samhandling med andre system. IEEE seier at dette er to eller fleire system, eller komponentar, si evne til å utveksle informasjon, og til å bruke den informasjonen som har blitt utveksla (IEEE90).

Ethical requirements kan omsettast som etiske krav. Dette er ein universell moral som ein organisasjon adopterer og lever etter. Desse krava kan anten kome frå utviklarane, kundane eller frå samfunnet utanfor sjølve utviklings- og brukarsituasjonen.

Legislative requirements omfattar alle dei krav som kjem frå ulike lover og regelverk, og som både kunden og utviklarane må forholde seg til. Dette kan igjen delast inn i to hovudgrupper.

Safety requirements er krav til sikkerheit, men då den lovfesta sikkerheita. Sikkerheita under produktkrava gjekk på data som systemet inneheld, behandlar og leverar. Her er det snakk om å beskytte eksterne partar mot systemet, til dømes brot på opphavsrettar, lisensar med meir.

Privacy requirements er krav til personvern. Dette er lover og reglar om privatliv, både til brukarar, utviklarar og andre som har med systemet å gjere. Det er viktig at ingen private data i systemet vert gitt ut til andre som ikkje skal ha tilgang til desse.

3. Metodekapittel

Vitskapleg metode er framgangsmåtar eller teknikkar ein nyttar for å gje svar på forskingsspørsmål (Ringdal 2001). Slik sett vert metode ein vesentleg del av ei masteroppgåve. Det finst mange ulike metodar, men alle har til felles at dei gjev råd om korleis ein skal gå fram for å finne gode svar (Østbye et. al 2002). I følgje Gentikow (2005) skal metodelære først og fremst hjelpe forskaren med å treffe hensiktsmessige val. Dette gjeld både val av datamateriale og val av korleis dette materialet skal nyttast under arbeidet med eit forskingsspørsmål.
Forskingsspørsmål kan ha ulike løysingar og svar. Det vert difor viktig at ein vel den framgangsmåten som verkar å gje den mest hensiktsmessige informasjonen i forhold til problemstillinga (Gentikow 2005). Den valde forskingsmetoden skal også vere med på å sikre at arbeidet med forskingsspørsmålet og datamaterialet vert utført på riktig måte. Metoden skal hjelpe til med opent å gjere reie for framgangsmåtane som er nytta og vise korleis fakta, argument og resonnement fører fram til ein bestemt konklusjon.

Dette kapittelet vil vidare gå nærmare inn på korleis arbeidet med denne oppgåva vart lagt opp, kva metode som har vorte nytta, og korleis denne er brukt.

3.1. Startfase

Eit forskingsprosjekt, som denne masteroppgåva, kjem frå eit ønske om å finne ut noko meir om eit tema eller svare på eit spesifikt spørsmål. Valet av denne oppgåva kom ut frå eit ønske om å finne ut meir om ulike systemutviklingsmetodar. Under arbeidet med prosjektbeskrivinga endra dette seg noko, og den endelege oppgåva konsentrerer seg om ein del av prosessen med å utvikle nye datasystem – dei ikkje-funksjonelle krava.

Etter at prosjektbeskrivinga var ferdig og godkjend, starta arbeidet med sjølve oppgåva. Det sentrale i arbeidet med ei masteroppgåve er utarbeiding av eit forskingsspørsmål. Dette var difor det første som vart gjort. I tillegg vart det formulert to forventningar til datamaterialet som oppgåva også tek føre seg.
Utveljing av relevant teori om ikkje-funksjonelle krav var neste steg. Som ein del av dette arbeidet vart det også utarbeida ein modell av ikkje-funksjonelle krav som

oppgåva har nytta under datainnsamlinga og analysen. Deretter starta arbeidet med å lage eit forskingsdesign. Dette er ein samla plan for innsamling og arbeid med data (Østbye et. al 2002). I eit forskingsdesign skal ein avgjere kva metode som skal nyttast i arbeidet, kva datamateriale ein skal nytte og korleis ein skal tolke eller analysere datamaterialet for å kome fram til eit svar på forskingsspørsmålet.
I valet mellom ulike metodetypar har denne oppgåva valt å nytte kvalitativ metode. Valet fall også på tekst som datamateriale. Datamaterialet vart gjennomgått ved hjelp av teorien og modellen. Målsettinga var å finne data om ikkje-funksjonelle som så skulle brukast til å svare på forskingsspørsmålet

Slik dei fleste metodebøker forslår, har arbeidet med oppgåva vorte utført i ulike fasar. Førebuingsfasen var den første. Her gjekk arbeidet ut på å finne fram relevant teori til oppgåva sitt tema, utarbeide ein modell, og å velje metode. Det neste steget var å finne fram til datamateriale og starte innsamlinga av data med bakgrunn i modellen. Data frå bøkene vart deretter systematisert, analysert og tolka i den neste fasen. Arbeidet med oppgåva vart avslutta med ein oppsummeringsfase der forskingsspørsmål, teori og data vart sett saman i forsøk på å kome fram til ein konklusjon.

3.2. Kvalitativ metode

Ein skil vanlegvis mellom kvalitative og kvantitative metodar i samfunnsforsking (Ringdal 2001). I valet mellom å studere systemutvikling i eit breitt perspektiv med kvantitativ metode eller å gå i djupna med kvalitativ metode, har oppgåva valt å nytte den kvalitative. Den kvalitative metoden arbeidar med små utval av tekstdata. Fokuset i denne typen metode er på meining og formålsforklaringar, og metoden er undersøkande.
Målet med den kvalitative metoden er å forstå røynda slik den vert oppfatta av dei personane som forskaren studerer. Denne oppgåva skal sjå på systemutviklingslitteratur. Forskingsobjekta vert difor forfattarane av desse bøkene, og målet er å finne ut kva desse seier om ikkje-funksjonelle krav i systemutviklingsprosessen.

3.3. Tekstanalyse

Det finst mange ulike typar kvalitative metodar. Denne oppgåva skal i hovudsak konsentrere seg om tekstanalyse. 'Tekstanalyse' er ein generell nemning på kvalitative studiar av tekstar.

Analyse tyder å dele opp noko i sine enkelte delar. Formålet med analysen er å finne fram til enkeltdelar og eigenskapar ved til dømes ein tekst, som gjer det enklare å forstå heilskapen og kome fram til ei tolking eller ein konklusjon (Dysthe et. al. 2004). I ein tekstanalyse stiller forskaren spørsmål til teksten, og prøver å finne ut meir om den (Østbye et. al 2005). Dette kan vere temaet og innhaldet til teksten, eller korleis dette vert presentert til lesaren. Det å studere tekstane gjev oss ikkje kunnskap om korleis tekstar vert mottekne eller verkar på enkeltindivid eller samfunnet (Østbye et. al 2005). Ein vil likevel kunne få ein betre forståing av den spesifikke teksten sin objektive presentasjon av sitt tema. Tekstar inngår i samanhengar og kontekstar, og representere desse. Det vert difor også viktig å sjå på desse kontekstane for å forstå teksten best mogleg.

Det ein vanlegvis forbind med tekst i eit kvalitativt studie er resultatet av eit intervju. Tekstane denne oppgåva nyttar som datamaterialet er ulike bøker innan systemutviklingslitteraturen. Desse skal samanliknast med omsyn til deira presentasjon av ikkje-funksjonelle krav. Bokteksten er meir omfattande og lengre enn intervjuteksten. Den er likevel ein tekst på lik linje med alle andre, og vanlege tekstanalyse teknikkar vil difor kunne nyttast som ved med analysering av eit intervju.

3.4. Datamateriale

Ein analyse startar med innsamling av data frå eit datamateriale (Gentikow 2005). Datamaterialet til denne oppgåva er fire systemutviklingsbøker som har vorte nytta i undervisninga ved Institutt for informasjonsvitenskap (Ifi) ved Universitetet i Bergen. Den historiske dimensjonen til forskingsspørsmålet gjer at datamaterialet består av ei bok frå kvart av dei tiåra som systemutvikling har vore undervist i ved Ifi, det vil seie 1970, -80, -90 og 2000. Bøkene er valt ut med hjelp frå fakultetet og tilsette som sit inne med informasjon om litteratur som har vorte nytta. Bøkene frå 1970 og 1980-talet er anbefalt av Svein Nordbotten. Gunnar Soleim og Andreas Opdahl har hjelpt til med å finne bok frå 1990-talet. Frå 2000-talet ser oppgåva på den boka som er på pensumlista hausten 2006.

Innsamlinga av data skal skje etter ulike nivå av informasjon. For å kunne svare på forskingsspørsmålet vert det viktig å få ei overordna kjensle av boka, forfattaren og innhaldet i teksten. Deretter skal ein, med bakgrunn i modellen utarbeida i teorikapittelet, sjå på dei delane av teksten som omhandlar ikkje-funksjonelle krav. Her kan det verte aktuelt å hente ut sitat som viser kva ulike tekstar seier om det same omgrepet. Under arbeidet med å samle inn data, vert det viktig å ha forskingsspørsmålet og forventingane til datamaterialet i bakhovudet. Ein må likevel vere open for interessante beskrivingar i teksten som ikkje direkte relaterar seg til oppgåva sitt forskingsspørsmål, men som likevel kan vere relevant.

Modellen oppgåva presenterer i teorikapittelet består av ulike omgrep som alle høyrer inn under ikkje-funksjonelle krav. I arbeidet med datamaterialet er første steg å sjå etter desse ulike omgrepa i teksten. Her vert det viktig å vere merksam på at det er variasjonar i bruka av omgrep og i innhaldet til dei ulike omgrepa. Ein må også passe på at ein ikkje misser noko i oversettinga frå engelsk til norsk. Dette er alltid ein fare når ein arbeider med tekstar på andre språk enn det som er eins eige morsmål. Interessante sitat om det aktuelle temaet skal hentast ut, og er med på å vise boka og forfattaren sitt forhold til systemutvikling og ikkje-funksjonelle krav.

Det å trekke ord og setningar ut av ein tekst, og plassere dei i ein annan, er noko ein skal vere varsam med. Omgrep i ein tekst er i stor grad kontekstavhengige. Det vert difor viktig at eg ser og forstår dei i forhold til den konteksten dei vert presentert i og den konteksten dei er tenkt brukt i. Som oftast vil dette gå fram av forordet til forfattaren i dei ulike bøkene. Bøkene er alle skrivne av forskarar innan systemutvikling og informasjonssystem. Dette gjer at ein her vil forvente anna tilnærming til temaet enn om det var praktikarar som hadde skrive bøkene.
Ein må vere klar over at det i kvar bok er forfattaren sitt synspunkt eller forståing av det aktuelle temaet som vert presentert. Dette treng ikkje reflektere det som var den generelle meininga av temaet på denne tida. Bøkene er likevel interessante då dei gjev uttrykk for ein trend innan systemutviklinga. Valet av akkurat desse bøkene til undervisning tilseier også at det som står i den er interessant for studentar å lære om innanfor det aktuelle feltet.

Dei ulike omgrepa som vert presentert i modellen skal nyttast som overordna kategoriar for innsamling og organisering av data. Planen er å lese gjennom bøkene to gongar. Under den første gjennomlesinga skal ein finne fram til og hente ut hovudpunkta. Her skal ein konsentrere seg om dei delane av teksten som direkte relaterer seg til systemutvikling, drift, ikkje-funksjonelle krav og dei ulike underkategoriane i modellen. Desse data vert deretter systematisert under dei ulike kategoriane i modellen av ikkje-funksjonelle krav, med sitat og interessante problemstillingar i forhold til tema.
Den andre gjennomlesinga skal vere med på å sikre at ein ikkje har hoppa over noko, eller tolka noko feilaktig for å få det til å passe til forskingsspørsmålet og forventingane.
Under datainnsamlinga må ein arbeide strukturert og systematisk. Dette for å sikre eit godt resultat og at ein ikkje må gjere arbeidet å nytt. Slik sett vert det også viktig å ha teori og metode klarlagt på førehand, slik at ein har eit godt grunnlag for det vidare arbeidet med oppgåva.

3.5. Teori og analyse

Etter at ein er ferdige med innsamlinga av datamaterialet startar arbeidet med å analysere funna og finne svar på forskingsspørsmålet og forventingane. ”Det kan være en god idé å begynne analysen med å gå tilbake til utgangspunktet: ens intensjoner og problemstillinger” (Gentikow 2005:132). Østbye et. al (2005) set opp tre viktige aspekt for analyse av kvalitative data: 1) Data, og analysen av dei, må forankrast i overordna forskingsspørsmål og teoretisk perspektiv. 2) Innsamling av og arbeid med data må skje systematisk. 3) Ein må finne ut kva relevans ulike typar data har for dei forskingsspørsmåla som vert lagt fram.
All analyse inneber ein viss grad av tolking, og det er viktig at ein ikkje tolkar noko inn i materialet som ikkje er der. I ein analyse tek ein også med seg sine kulturelle og historiske erfaringar, noko som kan vere med på å farge tolkinga og resultatet. Dette er noko ein må vere klar over når ein startar analysen av innsamla data.

Planen for denne oppgåva er å sjå på bøkene med utgangspunkt i teorien, og då spesielt modellen, som vert presentert i teorikapittelet. Med bakgrunn i dette skal eg gå systematisk gå gjennom dei ulike bøkene og sjå kva desse seier om identifisering

og bruk av ikkje-funksjonelle krav. Modellen består av ulike omgrep som alle høyrer inn under ikkje-funksjonelle krav. I tillegg skal eg sjå på kva datamaterialet seier om krav generelt, kravspesifisering, ikkje-funksjonelle krav i drift, og korleis ein kjem fram til og skal bruke krava i ein systemutviklingsprosess. Funna vert deretter organisert under dei ulike modellomgrepa dei høyrer til. For å kunne svare på forskingsspørsmålet og forventingane oppgåva har til datamaterialet, vil oppgåva også samanlikne funna frå dei ulike bøkene for å sjå om det har vore ei utvikling. Det å samanlikne bøker og litteratur frå ulike tidsperiodar kan vere vanskeleg. Spesielt gjeld dette bruken av ulike omgrep for same innhald, eller ulikt innhald for like omgrep. Østby et. al (2005) seier at når ein forskar skal ta for seg materiale som strekkjer seg over ein lengre tidsperiode, må vedkomande vere klar over at omgrep og folk sin måte å klassifisere tema på endrar seg over tid. Dette vert viktig å ta omsyn til ved utføring av ei samanlikning.

I tillegg til presentasjonen av datamaterialet har oppgåva ein drøftingsdel der eg vil oppsummere funna i forhold til forskingsspørsmålet og forventingane som vart presentert i innleiinga. Drøftinga ser på om det har vore ei utvikling innan identifisering og bruk av ikkje-funksjonelle krav i løpet av dei fire tiåra som bøkene representerer. Her vil eg også kome med nokre synspunkt på kvifor dei ikkje-funksjonelle krava er viktige å ta omsyn til i ein utviklingsprosess.

3.6. Validitet, reliabilitet og generalisering

Validitet, reliabilitet og generalisering er omgrep som er viktige innan forskinga. Sjølv om dei er mest vanlege innan kvantitative studiar, har ein etter kvart også teke dei inn i studiar som nyttar kvalitativ metode.

Validitet handlar om gyldigheit, om noko er truverdig, og går fyrst og fremst på relevansen av data og analyse i forhold til problemstillinga. Reliabilitet tyder pålitelegheit og gjeld kvaliteten i innsamlinga, omarbeidinga og analysen av data. Behandlinga av data må vere nøyaktig og påliteleg for at resultatet til oppgåva skal vere noko som ein kan stole på. Generalisering skjer ved at ein går frå delane til ein heilskap, og målet er å få samsvar mellom delane og heilskapen (Østbye et. al 2005). Alle desse tre omgrepa skal takast omsyn til ved vurderinga av eit forskingsprosjekt sin metodiske kvalitet. Stor grad av validitet og reliabilitet vert sett på som ein føresetnad for å kunne generalisere funna til analysen (Østbye et. al 2005)

Ved utforminga og arbeidet med denne oppgåva, har desse omgrepa blitt lagt vekt på. Først og fremst vert det viktig å ikkje trekke for bastante konklusjonar frå det datamaterialet oppgåva har nytta. Dei fire bøkene som utgjer datamaterialet gjev ikkje ei fullstendig oversikt over den litteraturen som både har vore skrive og nytta i undervisning i systemutviklingsfaget det siste 40 åra. Eg vil likevel hevde at den er representativ for det forskingsspørsmålet som oppgåva skal forsøke å svare på. Målet til oppgåva er heller ikkje å gje bastante løysingar, men heller vere med på å rette fokus mot ein del av systemutviklingsprosessen som ikkje er så mykje omtala.

4. Presentasjon av datamaterialet og datafunn

Dette kapittelet skal sjå nærmare på datamaterialet til oppgåva og dei funna som har blitt gjort gjennom analysen av dette materialet. Analyse tyder å dele opp noko i sine enkelte delar. Formålet med ein analyse er å finne fram til enkeltdelar og eigenskapar ved til dømes ein tekst, som gjer det enklare å forstå heilskapen og kome fram til ei tolking eller ein konklusjon (Dysthe et. al. 2004). I ein analyse skal ein nytte teori og metode for å hente ut relevant informasjon frå datamaterialet. Informasjonen skal deretter samanfattast med bakgrunn i dei forventingar og det forskingsspørsmålet som oppgåva presenterer i kapittel 1.

Dette kapittelet startar med ein nærmare presentasjon av dei fire bøkene som utgjer datamaterialet. Denne presentasjonen ser på vesentlege trekk ved bøkene. Der vert forfattaren sine eigne ord og tankar om bøkene knytt saman med mitt overordna inntrykk av bøkene ved gjennomlesing, og teksten sitt konkrete innhald.
Oppgåva vil deretter sjå nærmare på dei data som har blitt henta ut av bøkene. Modellen frå metodekapittelet har fungert som grunnlag for innsamling og sortering av data, og presentasjonen av funna vil også følgje denne modellen.

4.1. Om Systemutviklingsbøkene

Alle dei fire bøkene har vorte nytta ved undervisning i systemutvikling ved Universitetet i Bergen. Bøkene vart valt på bakgrunn av forslag frå tilsette ved Institutt for Informasjon- og medievitenskap, og det noverande pensum i kurset i ’Systemutvikling’.

4.1.1. Theoretical Analysis of Information Systems (1973)

Denne boka er skrive av Børje Langefors, ein svensk professor ved Stockholm Universitet, som har hatt stor innflytelse på utviklinga innan skandinavisk og norsk informasjonsvitenskap. Oppgåva har sett på 4.utgåve som kom ut i 1973 og er på 489 sider. Boka er ei fagbok innan systemutvikling og informasjonssystem, og vart brukt i undervisninga av systemutvikling ved Institutt for informasjonsvitenskap, UiB på 1970-talet

Den systemteoretiske tradisjonen oppstod på 1960-talet. Utfordringa på denne tida var å konstruerer store datasystem som skulle kontrollere verksemda til bedrifter og sikre rasjonell og forutsigbar drift. Langefors sine teoriar om systemutvikling på midten av 60-talet vart basis for ein systemutviklingsmetode kalla ISAC (Information Systems for Administrative Control). Denne metoden vert nærmare beskrive i *Theoretical Analysis of Information Systems*, men vert ikkje gått inn på i denne oppgåva.

Denne boka er ein god representant for den systemutviklingsteorien som vaks fram på 1970-talet – *strukturert systemutvikling*. Hovudfokuset til boka er informasjonssystem og utviklinga av desse. Boka tek for seg analyse og modellering av systemstrukturar og viser korleis ein kjem fram til systemeigenskapar ved hjelp av utrekning og simulering.

I motsetnad til den 'typiske' informasjonsvitenskaplege systemutviklingsboka, er denne boka full av matematiske uttrykk, tekniske beskrivingar, fokus på datastrukturar og forslag til utrekningar. Systema vert brote ned til mindre delar, som så vert designa og sett saman ved hjelp av matematikk. Forfattaren har nytta funksjonar og matriser for å vise korleis eit system fungerar og skal settast saman. Dette gjer boka nokså tung å lese og ikkje heilt enkelt å forstå.

Kravspesifisering og ikkje-funksjonelle krav vert ikkje omtala spesifikt. Nokre av omgrepa som fell inn under ikkje-funksjonelle krav vert likevel nemnt. Dette gjeld mellom anna minne, lagringsplass og kapasitet. I tillegg vert kostnader ved utvikling av eit system trekt fram som eit viktig moment. Boka framhevar viktigheita av å designe eit system som er ønska av brukarane eller som brukarane treng, og av at organisasjonen får det systemet dei verkeleg ønskjer. Hensikta til boka er å vise korleis ein kan oppnå dette på ein effektiv måte.
Forfattaren seier at det er den kombinerte effekten av systemet sine komponentar som er viktig. Det å designe eit godt system krev forståing for korleis komponentane i eit informasjonssystem heng saman og samarbeidar. Når ein designar eit system må ein foreta mange val, og alle desse gjev avgrensingar på/til systemet. Når ein så har kome fram til så mange avgrensingar at berre eit alternativ for implementering er mogleg, er designoppgåva ferdig. Forfattaren seier også at design til informasjonssystem må orienterast mot systemet som ein heilheit og ikkje dei enkelte systemdelane. For å

garantere at brukarane sine behov er teke nok omsyn til, må desse avgrensingane spesifiserast fyrst og dei tekniske introdusert så seint som mogleg.

4.1.2. The Analysis and Design of Computer Based Information Systems (1985)

Denne boka er skrive av Joan C. Nordbotten. Nordbotten er førsteamanuensis ved Universitetet i Bergen, og har også vore styrar for Institutt for Informasjonsvitenskap. Boka kom ut i 1985 og er ei fagbok for systemutvikling og informasjonssystem. Den vart brukt i undervisninga av systemutvikling på 1980-talet. Over 399 sider presenterar Nordbotten korleis ein kan gjennomføre analyse og design i samband med utvikling av nye databaserte informasjonssystem. Boka er ein introduksjon til det breie feltet systemanalyse, design og implementering av databaserte informasjonssystem, og er, i følgje forfattaren, skriven som pensumbok for introduksjonskurs i systemanalyse og design.

På midten av 1980-talet var det ein veldig auke i bruken av datamaskinbaserte informasjonssystem. I boka seier forfattaren at mange av desse datasystema vart fiaskoar. I dette ligg det at dei ikkje gav organisasjonen dei informasjonstenestene dei hadde bruk for. Systema var, i mange av desse tilfella, veldesigna. Problemet låg i at fokus under utviklingsarbeidet var for dårleg. Forfattaren seier at den einaste beskyttelsen mot dårlege system er å ”legge meir vekt på analysen av krav til informasjonssystemet og på utviklinga av eit system som møter desse krava” (Nordbotten 1985: xvii, mi oversetting).

Det å utvikle eit informasjonssystem består av ein serie av ulike repeterande oppgåver som generelt følgjer den naturlege systemlivssyklusen med analyse, design og implementering. Analyse og design av informasjonssystem er i all hovudsak ei samling av fakta og krav, strukturering og evaluering av desse, og til slutt design av eit nytt eller modifisert system.
Denne boka er meir samfunnsfagleg enn boka til Langefors. I dette ligg det at boka ikkje nemner matematiske utrekningar, indre datastrukturar og andre tekniske sider ved utviklinga. Fokuset er på designen av systemet, ikkje korleis denne skal implementerast. Boka snakkar mykje om krav, og om viktigheita av å ta omsyn til desse. Forfattaren har også med krav som kjem frå brukarane av eit system, og deler dagens oppfatning om at dette er viktig. Ho omtalar krav i generelle vendingar,

snakkar om korleis systemet skal vere (funksjonar), og er av og til innom eigenskapar til funksjonane, mellom anna sikkerheit, responstid og minnekapasitet.

4.1.3. SSADM version 4 – a Practical Approach (1995)

Denne boka er skrive av Mike Goodland i samarbeid med Caroline Slater. Boka kom ut i 1995 og vart brukt i ved Institutt for Informasjonsvitenskap, UiB på 1990-talet. I introduksjonen seier forfattarane at dette er ei bok som kan nyttast i undervisninga av SSADM-metoden, både ved universitet og høgskular og til sertifisering i SSADM version 4.

I skilnad til dei tre andre bøkene som utgjer datamaterialet til oppgåva, er dette ei bok som tek for seg *ein* spesifikk måte å utvikle databaserte informasjonssystem på – SSADM. Structured Systems Analysis and Design Method (SSADM) vart utvikla av eit statleg organ i Storbritannia i 1982 for å nyttast ved utvikling av informasjonssystem i forvaltninga. Metoden har, i si levetid, blitt forbetra gjennom endring av teknologi og brukarane sine opplevingar og erfaringar med metoden. Forfattaren seier at ”metoden er eit godt eksempel på strukturerte metodar, då teknikkane nytta her vert brukt i alle dei store strukturerte metodar for analyse og design” (Goodland & Slater 1995:v, mi oversetting).

Denne boka er, som Nordbotten si bok, konsentrert om dei samfunnsfaglege aspekta ved systemutvikling, og då i hovudsak fasane; analyse og design. Boka ser ikkje på programmering og implementering av systemet. Framgangsmåten til utvikling av nye informasjonssystem med SSADM vert presentert stegvis, og kvart steg har detaljert beskriving av dei ulike delane som skal vere med.

Boka har to kapittel der ein tek for seg krav, eit for analysering og eit for spesifisering. Her ser ein på korleis ein skal gå fram for å finne fram til og deretter definere ulike krav. Krav vert delt opp i funksjonelle og ikkje-funksjonelle krav, og skal samlast i ein kravkatalog som vert nytta vidare i design og implementering. Boka deler dei ikkje-funksjonelle krava inn i; krav til tenestenivå, restriksjonar på tilgang, gjenoppretting, revidering og krav til kontroll, avgrensingar og arkivering. Dette er den boka som har mest fokus på krav som ein del av systemutviklinga, og som også seier noko konkret om ikkje-funksjonelle krav.

4.1.4. Information Systems Development – Methodologies, Techniques and Tools (2003)

Denne boka er skrive av David Avison og Guy Fitzgerald. Oppgåva har sett på 3. utgåve som kom i 2003. Dette er også ei fagbok innan systemutvikling, og den vert nytta i dag i undervisninga av systemutvikling ved Institutt for informasjonsvitenskap, UiB. Boka er på heile 592 sider, der den presenterar ulike systemutviklingsmetodologiar. Hovudfokuset til boka er informasjonssystem, og meir spesifikt måtar ein kan utvikle desse på.

Forfattarane meiner at denne boka gjev eit godt grunnlag for kurs i informasjonssystem på alle nivå, frå introduksjon til spesialist, og at den er relevant for kurs med både eit datateknisk- og administrasjonsperspektiv.
Boka vert nytta av forelesarar og studentar over heile verda, og dekkjer eit breitt område innan systemutvikling.
I følgje Avison og Fitzgerald vart tidlege applikasjonar til datamaskiner oftast implementert utan bruk av ein spesifikk utviklingsmetodologi. Brukarane sine behov vart ikkje teke omsyn til, noko som igjen førte til at ein fekk design som ikkje passa til det målet ein hadde. I dag har ein fått ein utviklingsprosess med meir vektlegging på analyse og design og meir bruk av metodologiar.

Boka startar med å ta for seg ulike element som høyrer med i ein utviklingsprosess. Mellom desse finn ein teknikkar og ulike verktøy og rammeverk. Deretter tek forfattarane for seg mange ulike metodologiar, og presenterar hovudtrekka ved kvar av desse. Kvalitet og sikkerheit til systemet er dei to omgrepa som vert mest framheva. Dette er to felt det har blitt fokusert mykje på dei siste åra. Utbreiinga av Internet og dermed også nye måtar å drive datakriminalitet på, stiller høge krav til sikkerheit. Kvalitet er eit omfattande omgrep, jf ISO sin kvalitetsmodell, der også ikkje-funksjonelle krav spelar ei viktig rolle.
Ikkje-funksjonelle krav vert ikkje konkret nemnt i boka, men mykje av dei omgrepa som fell inn under denne typen krav finn ein igjen der forfattarane snakkar om kvalitetskomponentar til informasjonssystem. Mellom desse er tilgjengelegheit, effektivitet, vedlikehald og sikkerheit.

4.2. Presentasjon av datafunn

Datamaterialet nytta i denne oppgåva har vore dei fire bøkene som presentert ovanfor. Under analysedelen har desse vorte studert grundig, og relevante datafunn har vorte notert ned undervegs. Modellen, som er presentert i teorikapittelet, har fungert som rettesnor for kva som var interessant å ta notat av. Hovudfokuset under lesinga av bøkene har vore dei ulike ikkje-funksjonelle krava som modellen omfattar. I tillegg har oppgåva sett etter det som bøkene seier om krav generelt og om funksjonelle krav. Desse heng tett saman med ikkje-funksjonelle krav, og høyrer difor med i ei slik undersøking.

Når ein skal leite etter omgrep i ein tekst, er det viktig at ein ikkje ser seg blind på ordbruken. Det er ikkje sikkert at to bøker har brukt same namnet på det som eigentleg er same omgrep. Ved lesinga av bøkene har eg difor forsøkt å halde meg til dei ulike omgrepa sine definisjonar slik dei vert presentert i teorikapittelet. Omgrepa ’krav’ og ’funksjonelle krav’ vert presentert fyrst. Deretter vert dei ulike funna presentert kronologisk etter den hierarkiske oppbygginga til modellen.

4.2.1. Krav (Requirements)

Boka om SSADM definerar krav som eit trekk eller ei moglegheit som brukarane ønskjer ved det noverande eller framtidige systemet (Goodland & Slater 1995). Forfattarane seier at ein må ha krava på plass før ein startar med utvikling av eit nytt system. Når ein skal vurdere ulike framgangsmåtar for prosjektet, kan desse krava fungere som ei sjekkliste for ulike alternativ. Kravspesifiseringa skal baserast på ei forståing av det som skjer akkurat no i omgivnadane, og det noverande systemet skal analyserast og brukast som grunnlag for den nye systemet.

Kravspesifiseringskatalogen definerar alle krava ved å beskrive dei, gje dei ein prioritet, foreslå moglege løysingar og beskrive forretningsfordelane dersom krava vert oppfylt. Forfattarane seier også at kvart krav helst skal skrivast på ein målbar måte slik at det kan testast seinare. SSADM-boka (1995) er den boka som har mest tekst som omhandlar krav, og boka går grundig gjennom korleis ein skal kome fram til, behandle og nytte desse krava under utviklinga.

Nordbotten (1985) si bok er eit eksempel på den generelle systemutviklingsteorien, men også denne boka har med ein del tekst om krav. Forfattaren tek både for seg krav til informasjonen som systemet skal behandle, krav for prosessering av informasjonen

og krav som kjem frå brukarar av systemet. Ho går også nærmare inn på kva desse krava er og korleis dei skal nyttast i utviklinga.
Avison & Fitzgerald (2003) seier at krav til eit informasjonssystem skal registrerast under utviklinga, og at desse krava må vere riktige. I dette ligg det at ein må sørgje for at ein oppfattar dei krava som vert stilt på rett måte slik at systemet vert slik ein ønskjer. Forfattarane seier også at krav kan endre seg undervegs i eit prosjekt, og dette er noko ein må ta høgde for.
Langefors (1973) nemner krav-omgrepet i si bok, men det vert ikkje definert og han knyt det ikkje til nokon konkrete eksempel.

4.2.2. Funksjonelle krav (Functional Requirements)

Dei funksjonelle krava er ei undergruppe av krav på same måten som dei ikkje-funksjonelle krava. Funksjonelle krav tek for seg dei ulike funksjonane som det nye systemet skal oppfylle.
Langefors (1973) har i si bok ikkje noko inndeling i funksjonelle og ikkje-funksjonelle krav. Han seier likevel at ”our approach to the system analysis will be to start by defining the basic functions of the firm (…)” (Langefors 1973: 278). Dette skal gjerast for å finne ut kva organisasjonen har som ’mål’ med det nye systemet. For kvar funksjon må ein definere eit sett med nødvendig informasjon.
Nordbotten (1985) snakkar heller ikkje om funksjonelle krav, men ho nemner funksjonelle informasjonssystem. Dette er eit system som skal støtte opp om ein funksjonalitet i ein organisasjon. Her er det dermed organisasjonen og brukarane som skal definere kva funksjonar som systemet skal oppfylle. Eksempel på slike system er rekneskapssystem og administrasjonssystem.
SSADM - boka (1995) er den einaste som deler krav-omgrepet opp i funksjonelle og ikkje-funksjonelle krav. Dei funksjonelle krava skal vere ein del av kravspesifiseringskatalogen, og vert definert som ”aktivitetar det nye systemet skal utføre” (Goodland & Slater 1995:77, mi oversetting). Desse omfattar mellom anna lagring og henting av data, oppdatering av data, produksjon av rapportar, svaring på forespørslar og interaksjon med andre system.
Avison & Fitzgerald (2003) seier at ein må sjå på om dei funksjonelle krava til det eksisterande systemet er oppnådd. I dette ligg det at ein må kome fram til funksjonelle krav før ein startar med implementeringa av eit nytt informasjonssystem. Dei

funksjonelle krava er dermed ein del av utgangspunktet for utviklinga av det nye systemet.

4.2.3. Ikkje-funksjonelle krav (Non-functional Requirements)

I SSADM-boka (1995) vert ikkje-funksjonelle krav nemnt som ein del av kravspesifiseringa, og dei identifiserte krava skal leggast til som ein del av kravspesifiseringskatalogen. Ikkje-funksjonelle krav omfattar omgrep som utføring, sikkerheit, gjenoppretting, arkivering og revisjon, som alle er viktige eigenskapar ved eit system. Alle krav til eit system skal ideelt sett vere målbare slik at ein seinare kan teste om dei er oppnådde. I følgje Goodland og Slater kan dei ikkje-funksjonelle krava både omhandle heile systemet eller berre ein spesifikk funksjon.
I eit av dei siste kapitla i boka trekk Avison & Fitzgerald (2003) fram ulike kvalitetsmoment som dei meinar skal vere med i eit informasjonssystem. Mange av desse momenta høyrer til under modellen av ikkje-funksjonelle krav, og omgrepet ’kvalitetsmoment’ er på mange måtar deira term for ikkje-funksjonelle krav. I tillegg til denne delen av boka, vert også ikkje-funksjonelle krav omtala under presentasjon av RUP - metodologien.
Nordbotten (1985) brukar omgrepet ’constraints’ som samleomgrep for dei omgrepa som modellen i oppgåva har plassert under ikkje-funksjonelle krav. Definisjonen på ikkje-funksjonelle krav seier at dette er krav som avgrensar informasjonssystemet sine funksjonar. Sidan ’constraints’ kan oversettast med ’avgrensing’ kan ein dermed seie at Nordbotten nemner ikkje-funksjonelle krav, om enn ikkje direkte.
Langefors (1973) seier ikkje noko spesifikt om omgrepet ’ikkje-funksjonelle krav’ i si bok.

Vidare under kjem ei oppsummering av korleis dei ulike omgrepa i modellen vert presentert i bøkene i datamaterialet.

4.2.3.1. Dei ulike omgrepa som høyrer inn under produktkrav

Det er ingen av bøkene som nemner omgrepet ’produktkrav’ spesifikt, men alle bøkene snakkar om denne typen ikkje-funksjonelle krav, anten direkte eller indirekte.

Availability omfattar dei krav som går på systemet sin tilgjengelegheit. Dette er eit krav som går på når brukarane kan nytte systemet – til dømes om systemet skal vere mogleg å bruke heile dagen eller berre i arbeidstida.
Nordbotten (1985) nemner 'availability' som ein del av dei operasjonelle avgrensingane ein må ta omsyn til ved utvikling. Andre operasjonelle avgrensingar er økonomi og 'performance'.
SSADM – boka (1995) nemner 'availability' som ein del av servicenivåkrava, og seier at det har med systemet sin opptreden å gjere. Alle krava, både funksjonelle og ikkje-funksjonelle, vert i SSADM utarbeida på bakgrunn av ei evaluering av det gamle systemet, og av kva organisasjonen eigentleg treng og vil ha i det nye systemet. For 'availability' bør ein, i følgje forfattarane, avtale tenestenivå med kunden.
I boka til Avison & Fitzgerald (2003) vert 'availability' presentert som ein kvalitetskomponent. Dei seier at 'availability' går på om systemet er tilgjengeleg når det skal vere det og der det skal vere det. Kvalitetskomponentane som forfattarane legg fram i boka, er deira syn på kva som bør vere med for å få eit godt informasjonssystem.
Langefors (1973) har i si bok ikkje noko direkte om dette ikkje-funksjonelle omgrepet.

Continuity heng saman med 'availability'. For at eit system skal vere tilgjengeleg må det også ha ein viss grad av kontinuitet, og ein må ha planar for kva ein skal gjere dersom noko går feil.
Nordbotten (1985) seier at ein del av den fysiske sikkerheita til eit system er å ha kopiar av systemet, såkalla 'back-up'. Desse skal vere med på å hindre at ein mister data dersom det skjer noko, til dømes ved naturkatastrofar eller menneskelege feil. Back-up er ein viktig del av det å sikre kontinuiteten til eit system.
I SSADM-boka (1995) er 'recovery' er nemnt som ein del av dei ikkje-funksjonelle krava. 'Recovery' tyder gjenoppretting, og er ein viktig del av 'continuity' - omgrepet. I arbeidet med desse krava må ein ta opp spørsmål som: kor lenge kan ein klare seg utan systemet etter at det går ned, eller kor mykje data kan ein tole å miste ved systemfeil. Forfattarane seier også at krava til gjenoppretting har mykje å seie for val av maskinvare og programvare til systemet, og for planlegging av mellom anna back-up.

Avison & Fitzgerald (2003) seier at det skal vere mogleg å gjenopprette systemet dersom det oppstår ein feil eller systemet går ned. Ein må difor legge planar for dette under utviklinga.
I boka til Langefors (1973) står det ikkje noko direkte om 'continuity'.

Maintainability kan grovt sett oversettast med vedlikehald, og heng saman med begge omgrepa ovanfor. Ein må under utviklinga tenke på korleis ein skal reparere feil ved systemet eller gå fram for å få eit system opp å kjøre igjen.
Nordbotten si bok (1985) har eit heilt kapittel om 'maintenance' og viktigheita av dette i forhold til kvaliteten på systemet. Forfattaren seier at det er viktig at ein er klar over at ting kan gå gale, og dermed legg til rette for å reparere desse feila og gjere systemet tilgjengeleg igjen.
SSADM-boka (1995) konsentrerar seg om dei tidlege fasane til systemet, og går dermed ikkje inn på vedlikehaldstemaet og korleis dette skal utførast. Forfattarane seier heller ikkje noko om krav til denne delen av systemdrifta, eller korleis ein skal legge til rette for dette under utviklinga av systemet.
Avison & Fitzgerald (2003) seier på si side at eit system skal vere lett å vedlikehalde. "Vedlikehald er viktig fordi det er ein aktivitet som tek opp mykje av systemanalytikaren og programmeraren si tid" (2003:142, mi oversetting). Dei har også 'maintainability' som ein kvalitetskomponent ved det nye systemet.
Langefors (1973) si bok seier ikkje noko direkte om dette ikkje-funksjonelle kravet.

Reliability er også ei undergruppe av 'availability'. Dette omgrepet går på det at brukarane skal kunne stole på at systemet leverar det dei ønskjer og forventar. Langefors (1973) seier at det er ein viktig eigenskap for eit datasystem at ein kan stole på funksjonane som det har. Nordbotten (1985) seier at systemet og data i systemet må vere påliteleg. Forfattaren definerar dette som "den forventa prosentvise tida som konfigurasjonen er tilgjengeleg for bruk" (1985:277, mi oversetting). Dette er ein definisjon som i stor grad samsvarar med den oppgåva presenterar i teorikapittelet. Systemet må dermed vere tilgjengeleg ei viss tid (meir enn 98 prosent (1985:277)), og ein må kunne stole på at dette skjer. I SSADM-boka (1995) er 'reliability' nemnt som ein del av servicenivåkrava, og har med systemet sin opptreden å gjere. Forfattarane seier at 'reliability' omfattar spørsmålet om kva som er akseptabel nedetid for systemet, kor mange feil systemet kan ha, og maksimal nedetid for kvar feil. Med

nedetid meinar ein her den perioden der systemet ikkje er tilgjengeleg på grunn av ein eller annan feil. Boka seier også at ein bør avtale tenestenivået til 'relability' med kunden.
Avison & Fitzgerald (2003) seier at eit system bør ha definerte kvalitetsforventingar til påliteligheit, og nemner 'reliability' som eit kvalitetskomponent.

Security er ein viktig del av systemet sine ikkje-funksjonelle krav. Dette handlar om at brukarane skal stole på systemet og dei data som både vert lagt inn og kjem ut. Sikkerheita heng også saman med kontinuitet og det at ein ikkje misser data eller informasjon dersom noko uforutsett skjer med systemet. Denne typen ikkje-funksjonelle krav har tre undergrupper – 'data availability', 'data confidentiality' og 'data integrity'.

Nordbotten (1985) seier at ein treng ei sikkerheitsgruppe som skal etablere sikkerheit og kontrollprosedyrar for systemet. Forfattaren seier at data og informasjon i systemet skal vere tilgjengeleg for brukarane, og i tillegg korrekt og konsistent.
Systemkontrollar er designa for å beskytte systemet sin integritet, og dermed også dataintegriteten. Integritetskontrollen er spesielt opptatt av at systemet skal vedlikehalde, prosessere og presentere korrekt og konsistent data.
SSADM-boka (1995) omtalar 'security' som ein del av dei ikkje-funksjonelle krava. Forfattarane seier her at denne sikkerheita har med tilgangskontroll å gjere, dvs. kven som har rett til å bruke og sjå på kva data. Ein kan også sjå på fysiske sikkerheitsbeskyttelsar. Når det gjeld data integritet seier boka at ein viktig del av det å bygge databaserte informasjonssystem, er å gje oppdatert og korrekt informasjon. Data i systemet må også vere konsekvent og den same uansett kvar i systemet ein finn den.
Avison & Fitzgerald (2003) seier at også at systemet skal vere sikkert. Ingen skal kunne bruke systemet og informasjonen utan at dei har rett til det. For forfattarane er dette ein av kvalitetskomponentane til informasjonssystemet.
Langefors (1973) seier ikkje noko direkte om dette omgrepet.

Portability er kanskje eit av dei meir ukjende ikkje-funksjonelle krava. Det omfattar krav til bruken av systemet og kor fleksibelt det skal vere i forhold til andre system.

SSADM-boka (1995) nemner ’portability’ som ein del av dei interne avgrensingane ein kan finne til eit systemutviklingsprosjekt. Forfattarane definerar ikkje omgrepet noko nærmare.
Avison & Fitzgerald (2003) seier at ’portability’ er ein positiv kvalitet ved eit system, og er nemnt som ein av deira kvalitetskomponentar. Ved utvikling bør ein tenke på at systemet skal kunne køyre på anna utstyr eller andre stader.
Verken Langefors (1973) eller Nordbotten (1985) har noko direkte om dette.

Efficiency er krav til systemet sin prestasjon. Det består av to undergrupper - ’performance’ og ’space’, og er eit av dei omgrepa som alle bøkene nemner.
Langefors (1973) seier mellom anna at ”målet med systemdesign er å finne ein effektiv eller betre-enn-noverande løysing” (1973:266, mi oversetting). Han trekk fram at det ofte vert ei avveging mellom effektivitet og kostnadane ein kan bruke på utviklinga. Forfattaren meiner at både krav til minne og prosesseringshastigheit har vore undervurdert i dei åra datamaskiner har vorte nytta (ca 10 år i 1974). Ved utviklinga må ein difor ta omsyn både til responstid og det å sikre at systemet har nok minne.
Nordbotten (1985) seier at eit datasystemet skal effektivt og på ein god måte tilby dei datatenestene som organisasjonen og brukarane krev. Systemeffektivitet kan målast i form av utføringshastigheit og kapasitetskrav. Ei evaluering av operasjonssystemet si ‘performance’ er ein nødvendig del av det å bestemme om ein skal oppgradere systemet. Responstid og lagringsplass er også krav som ein må ta omsyn til.
SSADM-boka (1995) seier ikkje noko direkte om effektivitet, men snakkar om krav til ’performance’ og ’space’. Desse er nemnt som ein del av dei ikkje-funksjonelle krava. Responstid er nemnt som ein del av servicenivåkrava, og har med systemet sin opptreden å gjere. Forfattarane meinar at krav til systemet sin yteevne er ein vesentleg del av dei krav som brukarane skal vere med å bestemme.
Avison & Fitzgerald (2003) seier at informasjonssystemet skal hjelpe organisasjonen med å forbetre effektiviteten til sine operasjonar, og føre til betre styring og kontroll. Eit system bør difor ha definerte kvalitetsforventingar for effektivitet og for korleis det skal oppføre seg. Forfattarane meinar at systemet bør vere effektivt for å ha høg kvalitet. Avison og Fitzgerald seier ikkje noko direkte om krav til ’space’.

4.2.3.2. Organisasjonskrava ’delivery’, ’implementation’ og ’standards’

Organisasjonskrava er dei ikkje-funksjonelle krava som kjem frå retningslinjer og prosedyrer i utviklaren og kunden sin organisasjon.
Nordbotten (1985) seier at både menneske og organisasjonsstruktur er ein del av systemet sine omgivnader, og at dermed noko ein må ta omsyn til ved utvikling. I SSADM-boka (1995) seier forfattarane at interne krav kan verke inn på designen. Mellom desse krava finn ein obligatoriske fasilitetar, globale servicenivå, kriterium til datalagring og informasjonsformål.
Avison & Fitzgerald (2003) nemner også dei organisasjonsmessige krava som noko ein må ta omsyn til ved utvikling av nye informasjonssystem.
Langefors (1973) har ikkje noko om organisasjonskrav i si bok.

Omgrepa ’delivery’, ’implementation’ og ’standards’ høyrer alle inn under organisasjonskrav. Avison & Fitzgerald (2003) seier ikkje noko i si bok om ’delivery’ og ’standards’, men nemner ’implementasjon’. Dei seier at systemet skal vere mogleg og helst enkelt å implementere. Dette er ein av deira kvalitetskomponentar.
Ingen av dei andre bøkene har noko direkte om nokon av desse omgrepa.

4.2.3.3. Dei eksterne krava – ’interoperability’, ’ethical’ og ’legislative’

Når det gjeld dei eksterne krava seier Nordbotten (1985) at dette også må takast omsyn til. Eit informasjonssystem er ikkje eit lukka system, og må difor forhalde seg til input frå ulike aktørar. Desse eksterne brukarane omfattar kundar og klientar, leverandørar av materiale eller tenester, eigarar eller aksjeeigarar, og kontrollorgan som regjering eller revisor.
I boka om SSADM (1995) seier Goodland og Slater at eksterne krav som kan verke inn på design er kostnader og budsjett og maskinvare-/programvarestandarar.
Avison og Fitzgerald (2003) trekk fram forholdet mellom organisasjonen og omgivnadane. Organisasjonen vil verte påverka av lover og reglar frå regjering og storting, av konkurrentar, leverandørar og kundar. Dersom ein ikkje tek omsyn til dette vil dette gje dårlegare system.
Langefors (1973) har ikkje noko om eksterne krav i si bok.

Interoperability og ethical er to omgrep som kjem frå omgivnadene rundt organisasjonen og systemet.

Nordbotten (1985) seier ikkje noko om etikk i forhold til utviklinga. Ho nemner 'interoperability' og seier at dette heng saman med kompatibilitet og det at ein må kunne flytte eksisterande system til det aktuelle systemet. Dette er eit av dei kriteria som ho meiner eit nyutvikla system må vurderast etter.
Verken boka til Langefors (1973), Avison & Fitzgerald (2003) eller SSADM-boka (1995) har noko direkte om desse to omgrepa.

Legislative handlar om dei lovreglane og retningslinjene ein må følgje under utviklinga. Dette kan vere både lover som gjeld for heile samfunnet som organisasjonen, brukarane og systemet er ein del av, og spesifikke retningslinjer og reglar som ligg nedfelt i utviklaren og kunden sine organisasjonar. Denne typen krav har to undergrupper: 'privacy' og 'safety'.

Nordbotten (1985) seier at ein må forholde seg til nasjonale og lokale lover og reglar som skal beskytte industrien og den enkelte sitt privatliv. Når det gjeld 'safety' seier Nordbotten at ein bør ha systemkontrollar designa for å sikre systemet og systemet sine komponentar. Mellom anna bør ein ha kontroll av fysisk tilgang til komponentane av informasjonssystemet. Dette skal sikre mot uautorisert tilgang og bruk av systemet, og då spesielt beskytte data, prosessar og dataressursar. Her kan ein bruke både fysiske sikringar og passordsystem. 'Privacy' vert berre nemnt hos Nordbotten. Ho seier at ein ved utvikling av eit informasjonssystem må forholde seg til reglar om beskyttelse av privatliv. Systemkontrollar er designa for å sikre at systemet oppfyller krav til personvern.
SSADM- boka (1995) nemner at krav kan kome som følgje av lovpålegg eller lovendringar. Avison & Fitzgerald (2003) seier at ein må ta omsyn til dei krav som kjem frå lover og reglar når ein utviklar eit system.

4.2.4. Tabell over ikkje-funksjonelle krav i datamateriale

Tabellen under er ei oppsummering av kven av bøkene som har med dei ulike ikkje-funksjonelle krava i modellen. I tillegg viser tabellen kven av bøkene som seier noko om krav generelt og dei funksjonelle krava. Oppgåva skil mellom det at bøkene nemnar eit omgrep utan å definere det, og det at ein går grundigare inn på kva omgrepet tyder og kvifor det er viktig.

	Langerfors	Nordbotten	Goodland & Slater	Avison & Fitgerald
Requirements	**O**	**X**	**X**	**X**
Functional Requirements			**X**	**O**
Non-functional Requirements			**X**	**O**
Product Requirements				
Organisational Requirements		**X**	**X**	**X**
External Requirements		**X**	**X**	**X**
Availability		**X**	**X**	**X**
Continuity		**X**	**X**	**X**
Maintainability		**X**		**X**
Reliability	**X**	**X**	**X**	**X**
Security		**X**	**X**	**X**
Data Availability		**X**		
Data Confidentiality				**X**
Data Integrity		**X**	**X**	
Portability			**X**	**X**
Efficiency	**X**	**X**		**X**
Performance	**X**	**X**	**X**	**X**
Space	**X**	**X**	**X**	
Delivery				**X**
Implementation				
Standards				
Interoperability				
Ethical				
Legislative		**X**	**X**	**X**
Safety				
Privacy		**X**		

X = Omtala og forklart/definert på ein eller annan måte
O = Nemnt, men ikkje definert

Som tabellen viser, så er det ingen av bøkene som nemner alle dei omgrepa som modellen har med under ikkje-funksjonelle krav. Dette er som venta. Det er heller ikkje alle bøkene som deler krav inn i funksjonelle og ikkje-funksjonelle.

5. Drøfting

Denne delen av oppgåva oppsummerer funna frå datainnsamlinga. Første delen av kapittelet knyt datafunna saman med forskingsspørsmålet og forventingane, for å kome fram til ein konklusjon. Funna under arbeidet med datamaterialet har også gjort det nødvendig å lage ein oppdatert versjon av modellen av ikkje-funksjonelle krav. Denne vert presentert i punkt 5.2. Siste delen av kapittelet konsentrerer seg om mine eigne vurderingar i forhold til oppgåva sitt tema, og dei funna som har blitt gjort under arbeidet med oppgåva.

5.1. Forventingar og forskingsspørsmål

Oppgåva sitt utgangspunkt var ønsket om å undersøke ikkje-funksjonelle krav, og deira plass i utviklinga av nye informasjonssystem. Dette vart konsentrert i eit forskingsspørsmål og eit par forventingar presentert i innleiingskapitlet.
Denne delen av oppgåva går tilbake til forskingsspørsmålet og forventingane og knyt desse saman med funna frå datainnsamlinga presentert i kapittel 4. Oppgåva vil først ta for seg forventingane og drøfte desse. Deretter vert sjølve forskingsspørsmålet drøfta i forhold til datafunna, for å sjå om arbeidet med oppgåva har ført til eit svar.

Den første forventinga var at *datamaterialet har lite om ikkje-funksjonelle krav.* Denne setninga oppsummerer på mange måtar mitt utgangspunkt for val av tema. Ikkje-funksjonelle krav er eit omgrep eg ikkje har høyrt så mykje om i løpet av studietida mi innan informasjonsvitenskap. Det var dermed naturleg å forvente at det ikkje ville vere mykje om dette omgrepet i dei bøkene som utgjorde datamaterialet til oppgåva.
Omgrepet ’ikkje-funksjonelle krav’ vert ikkje direkte nemnt i Langefors si bok (1975). Langefors har heller ikkje med så mange av dei omgrepa som høyrer inn under dei ikkje-funksjonelle krava, og brukar ikkje mykje tid på kravspesifisering som del av systemutviklinga. Nordbotten (1985) nyttar heller ikkje direkte omgrepet ’ikkje-funksjonelle krav’, men brukar ordet ’constraints’ som samleomgrep for dei ikkje-funksjonelle krava. Denne boka nemner likevel mange av dei ikkje-funksjonelle omgrepa ein finn i oppgåva sin modell, og det vert snakka ein del om kravspesifisering og ikkje-funksjonelle omgrep som del av dette. Avison & Fitzgerald (2003) er så vidt inne på omgrepet ’ikkje-funksjonelle krav’ i samband med ein av

metodane dei presenterar i boka. I tillegg har forfattarane med mange av dei ikkje-funksjonelle krava i samband med deira liste over kvalitetsmoment i eit informasjonssystem. SSADM-boka (1995) brukar mykje tid på dei tidlege fasane av systemutviklinga, og både nemner ikkje-funksjonelle krav og trekk dei fram som ein del av kravspesifiseringa.

Presentert på denne måten kan det verke som om forventinga eg starta med ikkje stemmer – bøkene har ein del om ikkje-funksjonelle krav. Det er likevel nokre motargument. Det som er henta inn av data om ikkje-funksjonelle krav har kome gjennom metodisk jobbing. Under innsamlingsarbeidet har det vorte nytta ein modell med definerte omgrep, og ein har konsentrert seg om ein spesifikk del av teksten. Den vanlege studenten og lesaren vil ikkje lese ei systemutviklingsbok på denne måten. Dei er ute etter å få ei betre forståing av kva som høyrer inn under systemutviklinga, og ser etter den overordna strukturen. Ingen av bøkene i datamaterialet er klare på kvifor ikkje-funksjonelle krav er viktige, korleis ein kjem fram til dei og korleis ein skal ta omsyn til desse i utviklingsarbeidet. Dei ulike omgrepa i modellen av ikkje-funksjonelle krav er spreidd utover i bøkene, og ein får dermed ikkje ei forståing av at dei heng saman. Ser ein fokuset på ikkje-funksjonelle krav i forhold til fokuset på andre delar av systemutviklinga, vert det også klart at det ikkje er mykje plass i bøkene som er brukt til desse krava.
Konklusjonen på forventinga vert difor at ein likevel kan seie at det er lite om ikkje-funksjonelle krav i datamaterialet.

Den andre forventinga heng saman med den første, og seier at *det er meir om ikkje-funksjonelle krav i dei nyare bøkene enn dei eldste.* Denne forventinga kjem også av eiga erfaring med systemutviklingslitteratur. I tillegg har eg ei oppfatning av at ein legg meir vekt på det å analysere og designe det nye systemet før ein startar implementeringa i dag enn ein gjorde for 40 år sidan.
Dersom ein ser på tabellen som oppsummerar datafunna, er det fleire av postane som er utfylt ved dei siste bøkene enn ved den fyrste. Skiljet er likevel ikkje klart då Nordbotten si bok (1985) tek for seg like mange ikkje-funksjonelle krav som dei to seinare bøkene.
Tabellen i punkt 4.2.4 viser talet på omgrep i modellen av ikkje-funksjonelle krav som vert nemnde i bøkene. Når det gjeld omfang og plass, er dette vanskelegare å vurdere.

Det som er klart er at SSADM-boka (1995) tek for seg dei ikkje-funksjonelle krava mykje grundigare enn dei andre bøkene. Her vert store delar av eit kapittel brukt til å definere og beskrive kva del av utviklinga ikkje-funksjonelle krav høyrer til. Boka frå 2000-talet (Avison & Fitzgerald 2003) brukar mindre plass på dei ikkje-funksjonelle krava, eller kvalitetsmoment som dei kallar det. Samanliknar ein med boka frå 1970-talet (Langefors 1973), er det likevel ein vesentleg skilnad. Slik sett har det vore ei viss utvikling.

Forskingsspørsmålet til oppgåva var: *har fokus og innhald til ikkje-funksjonelle krav endra seg innan systemutviklingsfaget dei fire siste tiåra?*
Oppgåva har forsøkt å svare på dette spørsmålet ved å sjå på fire ulike systemutviklingsbøker nytta i undervisninga på 1970-, 1980-, 1990- og 2000-talet.

Det første som ligg i dette forskingsspørsmålet er om data frå bøkene viser ei utvikling i forhold til *fokus* på ikkje-funksjonelle krav. Med fokus meinar ein den mengda med omtale som dei ulike omgrepa har fått i litteraturen, og om ikkje-funksjonelle krav vert trekt fram som ein viktig del av systemutviklinga. Tabellen som viser forholdet mellom dei fire bøkene i kva ikkje-funksjonelle omgrep som er med, viser at det er fleire omgrep som vert omtala i dei siste bøkene enn i dei tidlegaste. Spesielt gjeld dette SSADM-boka frå 1990-talet der ikkje-funksjonelle krav har sitt eige avsnitt, og vert nemnt spesifikt fleire stader i boka. Det er i tillegg meir vekt på generell kravspesifisering, og då også analyse og design, i dei siste tre bøkene. Dette kan henge saman med endringane innan bruk og utvikling av datamaskinbaserte system. Den historiske gjennomgangen i teorikapitlet viser at datamaskina og datasystem har fått ein større og større plass i verksemder og organisasjonar. Dette har gjort det nødvendig å utvikle større og betre system, noko som igjen har ført til ei endring innan måtar å utvikle informasjonssystem på. Utover 1980-talet vart det meir vektlegging på ein organisert systemutviklingsprosess, og det å planlegge før ein implementerar.

I tillegg til at det er skilnad i fokus på dei samla ikkje-funksjonelle krava i bøkene, har også dei ulike ikkje-funksjonelle krava som modellen presenterer ulikt fokus i bøkene. Eksempel på dette er ’reliability’ og ’maintainability’. ’Reliability’ vert nemnt i alle bøkene, og trekt fram som eit av dei viktige krava til eit datasystem. Her er alle

forfattarane einige om at ein må kunne stole på at systemet utfører sine oppgåver. Dette er ein kvalitetskomponent, og på mange måtar også ein av dei viktigaste. Eit informasjonssystem som ikkje er tilgjengeleg når brukarane treng det, eller som ikkje utfører sine oppgåver slik det er tenkt, vil vere meir til hindring enn til hjelp for brukarane i deira kvardag. Dermed vil ein få eit av dei mange systema som kan klassifiserast som for dårlege, og som vert bytta ut med noko anna.
'Maintainability' vert berre omtala i to av bøkene. Nordbotten (1995) behandlar denne eigenskapen ved systemet grundig gjennom eit heilt kapittel, der ho trekk fram viktigheita av dette kravet for å få eit godt datasystem. Avison & Fitzgerald (2003) ser også på dette som ein viktig kvalitet ved eit informasjonssystem. SSADM - boka (1995) har ikkje med noko om 'maintainability'. Denne boka konsentrerer seg om dei første fasane av systemutviklinga, og har dermed ikkje med noko om det som skal skje etter at systemet er ferdig utvikla.

Den andre delen av forskingsspørsmålet ser på om innhaldet til ikkje-funksjonelle krav har endra seg i løpet av dei fire tiåra som datamaterialet ser på. Med innhald meiner ein her omgrepsforståinga - kva som fell inn under omgrepet ikkje-funksjonelle krav. Tabellen som oppsummerar funna av ikkje-funksjonelle krav i dei ulike bøkene, er ikkje til mykje hjelp for å svare på denne delen av spørsmålet. Som eg har nemnt før, er det like mange ikkje-funksjonelle krav som vert nemnt i bøkene frå 1980-talet, 1990-talet og 2000-talet. Dermed kan ein ikkje seie at boka frå 2000-talet (Avison & Fitzgerald 2003) har ein vidare definisjon av ikkje-funksjonelle krav enn boka frå 1980-talet (Nordbotten 1985). Det er også vanskeleg å vurdere innhaldet til dei ulike ikkje-funksjonelle omgrepa då ingen av bøkene har utgreiande definisjonar på kva omgrepa omfattar.
Generelt kan ein likevel seie at det er meir som vert omfatta av ikkje-funksjonelle krav i dag enn tidlegare. Dette har si naturlege forklaring i at datasystema har blitt meir kompliserte, og det vert fleire ting ein må ta omsyn til. Eit omgrep som 'sikkerheit' har i dag eit heilt anna innhald enn i starten av dataalderen. Innføringa av Internett og den auka bruken av datamaskiner i alle delar av vår kvardag, gjer at ein må ta meir omsyn til sikkerheit og også nytte andre former for sikring enn tidlegare.

Krav, og identifisering av desse, er ein del av forarbeidet ved utvikling av datasystem. Dette er i dag ei vanleg oppfatning blant systemteoretikarar og systemutviklarar.

Datafunna viser at kravidentifisering og bruk ikkje har like stor plass i boka frå 1970-talet (Langefors 1973), men at omgrepet vert meir omtala i dei seinare bøkene. SSADM - boka (1995) skil seg ut i positiv grad ved at den tek for seg ikkje-funksjonelle krav som ein vesentleg del av kravspesifiseringa.
Dei ulike produksjonskrava er den typen ikkje-funksjonelle krav som vert mest omtala. Organisasjons- og eksterne krav vert nemnt som noko ein må ta omsyn til og som kan verke inn på utviklingsprosessen, men lite utanom dette. Dette er kanskje også ein type krav som ikkje treng like mykje tid og plass som produktkrava. Det å måtte halde seg til lover og reglar er noko som kjem uavhengig av kven som er brukarar og kva desse meiner.
Dette ujamne forholdet mellom vektlegging av produksjons- og organisasjons/eksterne krav viser også at modellen er riktig i forhold til at den har flest ikkje-funksjonelle omgrep under produksjonskrav.

Konklusjonen vert, ut frå dette, at det har vore ei utvikling i retning av meir fokus på ikkje-funksjonelle krav i systemutviklingslitteraturen. På spørsmålet om det har vore ei endring i innhaldet til ikkje-funksjonelle krav, er det vanskelegare å kome fram til eit klart svar. Her treng ein meir forsking.

Arbeidet med oppgåva har konsentrert seg om forskingsspørsmålet, men dei to forventingane var også del av vurderinga av datamaterialet. Før eg starta dette arbeidet hadde eg ei forventing av at det ikkje ville vere mykje om ikkje-funksjonelle krav i litteraturen i datamaterialet. Dette har vist seg å stemme. Det er ikkje mykje om ikkje-funksjonelle krav i systemutviklingslitteraturen.
Oppgåva har tidlegare sagt at vektlegging av ikkje-funksjonelle krav er viktig for å få gode system. Ikkje-funksjonelle krav er eigenskapar ved systemet som er avgjerande for om systemet passar inn i brukarane si førestilling om eit godt system. Dersom systemet ikkje vert sett på som godt, ved at dei oppfyller krava til brukarane, er det heller ikkje eit godt system. Påstanden min vert difor at dagens systemutviklingslitteratur ikkje fokuserer nok på dei ikkje-funksjonelle krava og vektlegg dei under systemutviklinga i den grad ein burde.

5.2. Vidareutvikling av modell med bakgrunn i datafunn

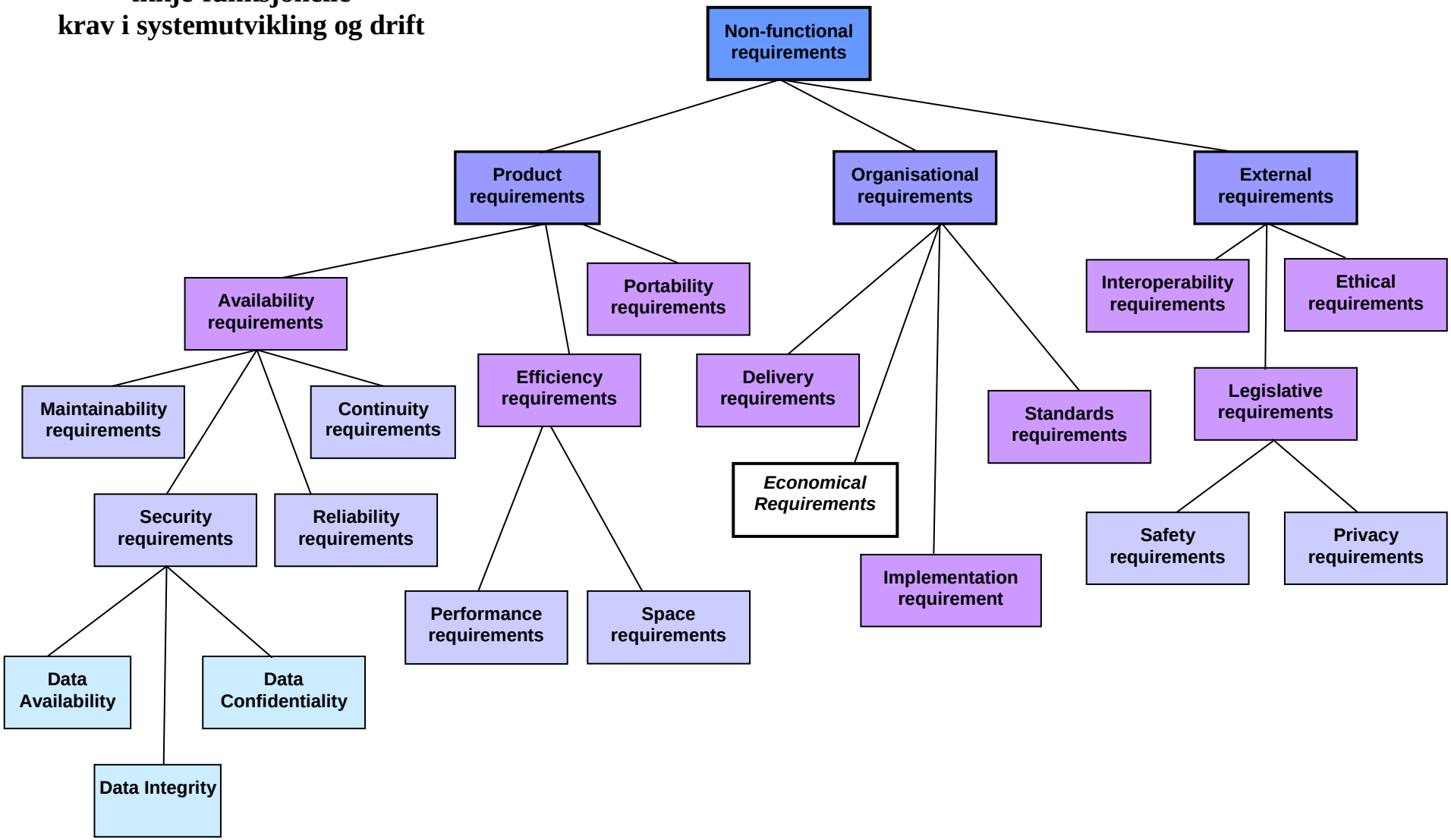
Modellen som vert presentert i teorikapittelet har fungert som ei god rettesnor ved gjennomlesinga av datamaterialet. Under arbeidet med datamaterialet fann eg likevel noko som har gjort det nødvendig å oppdatere modellen.
I tillegg til dei ikkje-funksjonelle krava som modellen omfattar, vart det i alle bøkene snakka om økonomi. Dette gjekk både på økonomi i forhold til kor mykje ein kan bruke på å utvikle systemet, og kor mykje ein kan spare i form av å bruke systemet. Mest fokus er det på økonomi knytt til sjølve systemutviklinga.
Langefors (1973) seier at ein må forta ein vurdering av kostnader opp mot verdi når ein skal planlegge og utvikle eit nytt informasjonssystem. Nordbotten (1985) deler dette synet, og seier at ein må foreta ein økonomisk evaluering før ein startar utviklinga av eit nytt system. Eit forslag til eit nytt informasjonssystem må ha ein positiv økonomisk evaluering. I dette ligg det at dei forventa fordelane/inntektene ved systemet må vere større enn dei forventa kostnadane. I si bok seier Goodland og Slater (1995) at kostnader og budsjett ved utvikling er eit krav som kjem frå utsida av sjølve prosjektet, og då ofte frå kunden sin organisasjon. Avison og Fitzgerald (2003) har med ’economy’ som ein av sine kvalitetskomponentar. Dette går på om systemet er kostnadseffektivt, det vil seie om systemet gjev høgare inntekter til organisasjonen sett i forhold til kostnadane ved utviklinga og bruken av det.

Fokuset på økonomi i datamaterialet gjer at dette omgrepet høyrer heime under modellen av ikkje-funksjonelle krav. Økonomi kan reknast som eit ikkje-funksjonelt krav fordi det ikkje er ein funksjon som systemet skal ha, men noko som kan legge avgrensingar på korleis utviklinga og bruken av systemet skal skje. På bakgrunn av dette har eg laga ein ny modell, med ein ekstra boks for økonomi under organisasjonskrav.

Skilnaden mellom ’security’ og ’safety’ er ikkje enkel, og bøkene snakkar ofte om dette som eitt og same omgrep. Modellen vert likevel ikkje endra på dette punktet, då det er ein vesentleg skilnad mellom den sikkerheita som går på sjølve systemet (’security’) og den som går på fysiske sikkerheitstiltak i systemet sine omgivnader (’safety’).

Sjølv om modellen no er oppdatert, er den nok framleis ikkje komplett. Dette vil krevje meir forsking, og innsamling av meir data om temaet.

Modifisert modell av ikkje-funksjonelle krav i systemutvikling og drift



5.3. Eiga vurdering av ikkje-funksjonelle krav i systemutviklinga

Temaet og forskingsspørsmålet til denne oppgåva, kom frå eit ønske om å lære meir om ikkje-funksjonelle krav og deira plass i systemutviklinga. Eg hadde lite erfaring med dette omgrepet frå før, men syntest det verka interessant og verd meir forsking. Spesielt gjaldt det korleis ei betre forståing av dette omgrepet og bruken av det kan vere med på å forbetre utviklinga av nye informasjonssystem, og dermed også føre til betre system.

For at ein skal få eit fungerande system som skal passe inn i ein organisasjon og utføre dei oppgåvene det er laga for, må ein identifisere og spesifisere dei krav som organisasjonen og brukarane har til systemet. Dette er ikkje ein enkel prosess, og det krev innsats frå både utviklarar og kundar. Det er ikkje alltid kundane klarer å formidle til utviklarane kva krav dei har til systemet. Eksempel på dette er at kundane ønsker at det nye systemet skal vere meir effektivt. Dette er eit vagt krav som kan tolkast på mange måtar. Ynskjer dei meir lagringskapasitet, raskare utføring av oppgåver eller innsparing av kostnader? Her vil utviklarane måtte bruke mykje tid på å identifisere kva det er kunden eigentleg meinar. Dersom dei i staden satsar på ei av tolkingane, kan dette føre til at ein får eit system som ikkje passar inn i organisasjonen og kunden sine eigentlege behov. Krav er også som oftast situasjonsavhengige. Ein må difor bruke tid på å sette seg inn i organisasjonen og deira arbeid slik at ein kan tilpasse systemet til organisasjonen. I tillegg har slike krav ein tendens til å endre seg etter kvart som utviklingsprosessen skrid framover. Dette må ein også kunne ta omsyn til.

Dersom ein skal fortsette utviklinga mot stadig betre informasjonssystem, må ein legge meir vekt på dei delane av systemutviklinga som har direkte innverknad på brukarane. Dei ikkje-funksjonelle krava er vesentlege i arbeidet med å utvikle system som brukarane vil ha. Krava må difor identifiserast og det må lagast løysingar for at systemet i produksjon skal kunne oppfylle og tilfredstille dei. Slik sett er det, etter mi meining, viktig å få systemutviklingsbøker som har med ikkje-funksjonelle krav og som legg vekt på desse som ein viktig del av systemutviklinga og arbeidet med å få gode kvalitetssystem. Bøkene oppgåva har nytta som datamateriale har vorte og vert nytta i undervisninga. Dermed vil det vere mange studentar som ikkje har lært *nok* om

ikkje-funksjonelle krav i systemutviklinga. Oppgåva nemnde tidlegare at den ikkje skal gå nærmare inn i spørsmålet om det ein lærer på universitet og høgskule vert brukt når ein kjem ut i arbeidslivet. Likevel er det nærliggande å tenke at dersom ein ikkje har fått forståinga for kva ikkje-funksjonelle krav er og korleis ein skal kome fram til og bruke dei, vil ein heller ikkje bruke mykje tid på dette når ein sjølv skal ta del i ei systemutvikling.
I tillegg er dette også ein type krav som, i mange tilfelle, er vesentlige for brukarane, og som dei kanskje difor tek for gitt er inkludert i det nye systemet. Som nemnt i innleiinga, forventar vi som brukarar av ulike system at dei skal vere mellom anna tilgjengelege, pålitelege og sikre. Sidan vi ser på desse eigenskapane som noko sjølvsagt, vil ein ikkje bruke mest tid på dette i analyse- og designfasane. Dersom utviklarane heller ikkje er opplærte til at desse krava må identifiserast og spesifiserast eksplisitt, kan ein risikere at desse krava ikkje får nødvendig fokus, og i verste fall ikkje vert teke omsyn til i det heile.
Dermed vert det eit problem at bøkene ikkje går nærmare inn i kva krava går ut på, korleis ein skal kome fram til dei og korleis dei skal vere med på å utforme systemet slik brukarane ønsker. Lesaren av desse bøkene er den framtidige utviklaren. Når ikkje-funksjonelle krav berre vert nemnt som noko ein må ta omsyn til ved utviklinga, får ikkje lesaren ei god nok forståing av omgrepet og kor viktig del av systemutviklinga det er. Dersom ein ikkje får forklart korleis noko skal utførast, vert det ofte enklare å bruke meir tid på dei delane av utviklinga som er betre forklart.

Alt dette viser at kravspesifisering er ein prosess som krev mykje arbeid og kanskje også strengare retningslinjer. Dette gjeld mellom anna kva krav ein skal sjå på og kven som skal vere med under kravspesifiseringa.

5.3.1. Kva krav skal ein sjå på?

Kvart utviklingsprosjekt er noko nytt. Det er ingen slike prosjekt som er identiske. Eit informasjonssystem utvikla for ein bank, vil ikkje direkte kunne brukast i ein annan bank. I tillegg vil også dei ulike faktorane som skal takast med i betraktning endre seg, både frå prosjekt til prosjekt og undervegs i eit prosjekt. Ein får individuelle situasjonar, og dermed også ulike relevante ikkje-funksjonelle krav. Difor er det ikkje mogleg å seie kva av dei ikkje-funksjonelle krava som skal leggast vekt på. Det som

er klart er at ein bør ta med dei ikkje-funksjonelle krava som ein del av analysen og kravspesifiseringa.
Til å hjelpe med denne delen av utviklinga kan ein ha sjekklister som ein går gjennom for å finne dei relevante krava for kvart prosjekt. Her kan modellen av ikkje-funksjonelle krav som vert presentert i teorikapittelet vere til hjelp.
Ikkje-funksjonelle krav er med på å bringe kvalitet inn i systemutviklingsprosjekt. Dersom ein ser på ISO9126 sin kvalitetsmodell over kva som bør vere med i eit godt datasystem (sjå vedlegg 3), er både 'portability', 'maintainability', 'efficiency' og 'reliability' del av dei ikkje-funksjonelle krava.

Arbeidet med denne oppgåva har vist at det ikkje vert fokusert så mykje på ikkje-funksjonelle krav i dei bøkene som oppgåva har sett på. Likevel er det nok ikkje slik at ein i dag *ikkje* tek omsyn til denne typen krav ved utvikling av nye informasjonssystem i dag. Hadde dette vore tilfelle, ville brukarvenlege og fungerande system vore ei mangelvare. Det er heller slik at ein som utviklar er usikker på korleis ein skal gå fram for å sikre at ein får avklart dei ulike krava som kunden har, og teke nok omsyn til dei ved utviklinga. Oppgåva har allereie nemnt sjekklister som ei hjelp i dette arbeidet. I tillegg kjem det stadig fleire rammeverk som kan nyttast for å finne, bruke og ta omsyn til dei ikkje-funksjonelle krava i utviklinga av eit system. Dette tyder på at kravspesifiseringa vert ein stadig viktigare del av systemutviklinga.

For å kome fram til dei ulike krava må ein, som utviklar, snakke med dei systemet skal utviklast for. Kommunikasjon og samarbeid vert difor ein viktig del av eit utviklingsprosjekt. Utviklaren må kommunisere og samarbeide med dei som har interesser i systemet, for å kunne lage eit system som brukarane treng og som passar inn i organisasjonen. Eit spørsmål vert då kven er desse aktørane, og kven av dei skal vere med å utforme krav?

5.3.2. Kven skal vere med under kravspesifisering og utvikling?

Ingen av bøkene i datamaterialet er veldig klare på kven som skal vere med på å utforme dei ikkje-funksjonelle krava. Nordbotten (1985) vil ha med brukarane ved utarbeiding av krav. SSADM - boka (1995) seier at "det er eit grunnleggande prinsipp i SSADM at brukarane er involvert i, og dedikerte til, utviklinga av deira system frå

eit tidleg tidspunkt (Goodman & Slater 1995:3, mi oversetting). Avison & Fitzgerald (2003) seier at dei ”meiner det er ønskeleg at alle interessentar i systemet vert involvert i utviklingsprosessen” (2003:14). Ein må dermed gå lenger enn til leiing og programmerarar når ein skal samle inn ønskjer og krav til det nye systemet.
Kven som skal vere med under kravspesifiseringa er eit tema som nok vil variere frå eit utviklingsprosjekt til eit anna, men eg er av den oppfatning at det er viktig å legge nokre retningslinjer også på denne delen av utviklinga. Dette fordi ein då vil få ei auka bevisstgjering av kor viktig det er at krava som ligg til grunn for systemet er utarbeida av dei som kjenner organisasjonen og bruksområdet best.

Det at brukarane skal vere med, er det ikkje noko tvil om. Brukarane skal nytte informasjonssystemet i sin kvardag, anten på jobb eller fritid, og dei bør difor få vere med å avgjere korleis systemet skal utformast og kva eigenskapar det skal ha. Representantar frå kundeorganisasjonen si leiing bør også inkluderast. Desse er med på å legge organisasjonsmessige føringar på prosjektet, mellom anna når det gjeld kva standardar som skal nyttast eller kor mykje pengar ein kan bruke på prosjektet.
Etter at eit informasjonssystem er teke i bruk, er det driftspersonell som tek over ansvaret for at systemet skal kunne utføre sine oppgåver i samsvar med tenesteavtalen. Driftssituasjonen er den lengste delen av systemet si levetid. Når det gjeld krav som går på vedlikehald og generell drift, er det dette personalet som har den beste kunnskapen. Dermed vert det naturleg å inkludere driftarane som ein del av systemutviklingsarbeidet, og av kravspesifiseringa. På dette området er det manglar i datamaterialet, då dei fleste bøkene ikkje ser på utvikling og drift i samanheng. Eit eksempel på dette er SSADM - boka (1995) som ikkje seier noko om dei ikkje-funksjonelle krava som høyrer til under driftsfasen til systemet. Sjølv om dette er ei bok som konsentrerer seg om dei første fasane av systemutviklinga, kan ein likevel sette fokus på dei delane av systemutviklinga som skal skje etter at systemet er teke i bruk.

Ansvaret for at dei aktuelle personane får vere med i kravspesifiseringa ligg ikkje berre til utviklarane eller kunden. Her må det ei haldningsendring til hos alle som er involverte i systemet og utviklinga på ein eller annan måte. I tillegg til at alle skal ha lov til å vere med, må det også vere slik at ein ser det som si plikt å få til eit best mogleg produkt gjennom å delta aktivt med ulike innspel.

6. Konklusjon

Dette siste kapittelet i oppgåva oppsummerer arbeidet med oppgåva og resultata av dette arbeidet. I tillegg vert det sagt litt om interessante spørsmål for vidare forsking angåande ikkje-funksjonelle krav i systemutviklinga.

I starten av arbeidet med denne oppgåva, var min kunnskap om ikkje-funksjonelle krav mangelfull. Dette var eit omgrep som eg hadde høyrt om i forelesingar og lærebøker, men som hadde kome i skuggen av andre omgrep som systemfunksjonar, utviklingsmetodologiar og implementering. Etter å ha sett meg litt meir inn i kva ikkje-funksjonelle krav var, fekk eg lyst til å sjå nærmare på denne delen av systemutviklinga.
Det vart raskt klart at temaet ikkje var så oversiktleg som eg kanskje hadde ønska. Teori om emnet var ikkje lett å finne tak i, og det å finne ein beskrivande definisjon var enda vanskelegare. Ein stor del av oppgåva har dermed vore utarbeidinga av modellen av ikkje-funksjonelle omgrep i teorikapittelet, samt å klargjere og definere dei ulike omgrepa. Modellen knyt saman ikkje-funksjonelle krav frå både utviklings- og driftsdelen til eit informasjonssystem. Tanken bak dette er at driftspersonellet sine krav også skal takast omsyn til i utviklinga. Grunngjevinga for dette er at det er i driftsdelen av systemet at dei ulike krava vert oppfylt. Dermed vert det viktig at dei som skal oppfylle krava også får vere med å avgjere dei. På denne måten hindrar ein også at det vert sett urealistiske eller feil krav som gjer at systemet ikkje utfører sine oppgåver på ein så god måte som ein kunne ønske.

Når det gjeld datafunna var der nokre overraskingar i forhold til forventingar og forskingsspørsmål. Skiljet mellom bøkene i datamaterialet når det gjeld fokuset på ikkje-funksjonelle krav, var ikkje så klart som eg hadde forventa. Boka frå 1980-talet (Nordbotten 1985) nemner like mange av dei ikkje-funksjonelle omgrepa som dei to seinare bøkene. Denne boka nyttar imidlertid ikkje nemninga ’ikkje-funksjonelle krav’ om desse omgrepa, noko både SSADM-boka (1995) og boka til Avison og Fitzgerald (2003) gjer.
Det var også vanskelegare å lese gjennom bøkene og hente ut dei ikkje-funksjonelle krava enn eg hadde venta. Eg brukte mykje tid før eg verkeleg fekk med meg alle interessante data, spesielt sidan datamaterialet er skrive på engelsk.

Sett vekk frå desse mindre overraskingane har arbeidet med oppgåva har gått om lag som forventa.

Oppgåva vart lagt opp som eit litteraturstudie, der eg tok føre meg fire bøker innan systemutvikling. Meininga med denne oppgåva var ikkje å vurdere kvaliteten på bøkene, men å nytte dei som representantar for si tidsperiode når det gjeld ikkje-funksjonelle krav. Kartlegginga av litteraturen innanfor eit felt gjer at ein får eit inntrykk av kva som er den rådande teorien. Funna i denne oppgåva viser at dei ikkje-funksjonelle krava ikkje har noko stor plass i datamaterialet. Bøkene viser likevel at denne typen krav har fått ein større plass i systemutviklingsbøkene i løpet av dei tiåra som oppgåva ser på. Det er også klart at kravspesifisering generelt, som ein del av analyse og designfase, har ein større plass i utviklingsprosjekt i dag.

Nokre vil kanskje hevde at fire bøker er eit lite utval å trekke bastante konklusjonar frå. Dette kan vere tilfelle. For å sikre best mogleg resultat har eg forsøkt å finne bøker som var gode representantar for sitt tiår og den systemutviklingsteorien som var rådande då. Alle dei fire bøkene har blitt nytta i undervisning, noko som tilseier at det var dette studentane skulle lære på den tida boka vart brukt.
Eit anna spørsmål er om resultatet hadde blitt det same dersom eg hadde valt ut bøker frå t.d. BI, ein høgskule eller eit anna universitet i Noreg som underviser i systemutvikling. Motargumentet vert at ein går ut frå at undervisningsinstitusjonar vel dei beste bøkene for bruk i undervisninga. Og dermed også dei som best representerar systemutviklinga i det tiåret dei høyrer til. På dette grunnlaget vil eg hevde at bøkene i datamaterialet er representative og at datafunna gjev eit godt bilete på den faktiske situasjonen for ikkje-funksjonelle krav i systemutviklinga.

Gjennomlesing av bøker kan sjåast på som eit subjektivt arbeid. Tekstar kan forståast på ulike måtar alt etter kva innfallsvinkel ein har til dei. Eg har forsøkt å sikre at mi lesing og tolking av tekstane vart så objektiv som mogleg. Dette vart gjort ved å utarbeide ein modell av ikkje-funksjonelle krav som eg har arbeida etter. Kvart av omgrepa i modellen vart beskrive slik at eg kunne finne fram til omgrep sjølv om dei i bøkene hadde eit anna namn. Eksempel på dette er Norbotten (1985) sin bruk av ordet ’constraints’ på det som eg vil kalle ikkje-funksjonelle krav. Gjennomlesinga av bøkene har skjedd systematisk, og eg har teke notat av interessante data undervegs.

Eit informasjonssamfunn, som det vi lever i, krev gode system som kan forvalte all informasjonen. Dermed vert det også viktig at alle delane av systemet - funksjonar og eigenskapar - er på plass. Resultatet av arbeidet med oppgåva er ei stadfesting av at systemutviklingslitteratur har lite stoff om ikkje-funksjonelle krav. Eg vil hevde dette på tross av at SSADM - boka (1995) brukar mykje tid på ikkje-funksjonelle krav som del av kravspesifiseringa. I dei andre bøkene kjem det ikkje klart nok fram kva som er ikkje-funksjonelle krav, korleis ein skal kome fram til dei og korleis ein skal ta omsyn til desse i utviklinga. Etter å ha lese bøkene vil ein sitte igjen med ei kjensle av at ikkje-funksjonelle krav berre er eit av mange omgrep innan systemutviklinga, og ikkje så viktig som eg og denne oppgåva vil vise.
Arbeidet med oppgåva har også resultert i ein modell av ikkje-funksjonelle krav presentert i teorikapittelet. Denne kan nyttast både ved definering av ikkje-funksjonelle krav og som sjekkliste ved utvikling av informasjonssystem.

Dette temaet har mykje stoff for vidare forsking. Eit problem er at det manglar felles terminologi for denne type krav (jf. teoridelen), og for kva som høyrer til under dette. Dette gjer at utviklarar manglar ein felles referanse for korleis krav passar inn i utviklinga. I dette ligg det ikkje at all systemutvikling skal skje på same måten, men at ein skal ha nokre felles rammer som ein arbeider innanfor. Her er det rom for vidare arbeid. Det hadde også vore interessant å finne ut om ikkje-funksjonelle krav vert lagt vekt på i praktisk systemutvikling. Ei spørjeundersøking av kva systemutviklarar legg vekt på ved utvikling av nye informasjonssystem er eit alternativ for vidare arbeid.
Eit anna punkt som oppgåva har vore innom er samarbeidet, eller kanskje heller det manglande samarbeidet, mellom utvikling og drift ved systemutviklingsprosjekt. Dette er også eit område som er interessant å studere nærmare.

Systemutvikling er vanskeleg. Som utviklar må ein forholde seg til endringar i samfunnet, stadig ny teknologi og dei ulike behova, forventingane og krava frå organisasjonar og brukarar. Dette er ikkje alltid like enkelt. Dei mange faktorane gjer det enda viktigare å få på plass gode retningslinjer for korleis utviklinga skal skje. Ein må også bruke tid på å kome fram til dei momenta som er avgjerande for å få eit godt system. Kva som er eit godt system er ei subjektiv vurdering frå brukarane, basert på

kor godt systemet oppfyller deira forventingar, behov og krav. I dette ligg det både krav til funksjonar, til brukarvenlegheit og til ikkje-funksjonelle eigenskapar.
Mykje tid vert brukt på å få på plass funksjonar til eit system. Brukarvenlegheit har også vore i fokus dei siste åra. Denne oppgåva har sett på dei ikkje-funksjonelle krava, som er den delen med minst merksemd.
Framleis er det utviklingsprosjekt som slår feil, ved at ein får system som ikkje passar inn i verksemdene. Ved å fokusere meir på dei ulike krava frå brukarane, og gje dei ikkje-funksjonelle krava ein større del av dette, vil ein kanskje sikre at ein får betre informasjonssystem.

Litteraturliste

Andersen, E.S. (1989). *Systemutvikling*. Bekkestua: NKI Forlaget

Avison, D. E. & Fitzgerald, G. (2003). *Information Systems Development – Methodologies, Techniques and Tools* (3. utgåve). New York: McGraw Hill

Boehm, B. W., Brown, J. R., Kaspar, J. R., Lipow, M. L. & MacCleod, G. (1978). *Characteristics of Software Quality*. New York: American Elsevier.

Boehm, B. (2006). A View of 20th and 21st Century Software Engineering. *Proceeding of the 28th international conference on Software engineering*
Shanghai, China, 12 – 29. Henta 09.januar 2007 frå
http://portal.acm.org/citation.cfm?id=1134288&dl=acm&coll=&CFID=15151515&CFTOKEN=6184618

Christensen, G.E., Grønland, S. E. & Methlie, L.B.(1994). *Informasjonsteknologi - Strategi, organisasjon, styring*. Oslo: Bedriftsøkonomens Forlag

Davis, A. (1993). *Software Requirements: Objects, Functions and States*. New Jersey: Prentice-Hall International.

Dysthe, O., Hertzberg, F. & Hoel, T. Løkensgard. (2004). *Skrive for å lære – skriving i høyere utdanning* (4. opplag). Oslo: Abstrakt forlag

Evans, I. & MacFarlane, I. (Editors). (2001). *A Dictionary of IT Service Management – Terms, Acronyms and Abbreviations*. Reading: itSMF Ltd

Gurholt, G. & Hasle, T. E. (2003). *Grunnleggende Systemutvikling***.** Oslo: Cappelens Forlag AS

Gentikow, B. (2005). *Hvordan utforsker man medieerfaringer – kvalitativ metode.* Kristiansand S: IJ – Forlaget

Goodland, M. & Slater, C. (1995). *SSADM – a Practical Approach*. Berkshire: McGraw Hill

Hansen, T. B. & Hjertø, G. (2003). *Kvalitet og Programvareutvikling*. Oslo: Gyldendal Norsk Forlag AS

Hertroys, P. (Editor). Baron, A., Clarke, B., Hertroys, P., van Oosterom, N. & Hinley, D. (writers). (2002). *Application Management.* London: The Stationary Office

Langefors, B. (1973). *Theoretical Analysis of Information Systems* (4.utgåve). Philadelphia: Auerbach

IEEE Computer Society. (1990). *IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990.* The Institute of Electronical and Electronics Engineers, Inc.

ISO/IEC. (2001). *Software Engineering - Product quality - Part 1:Quality Model.* ISO/IEC 9126-1.

Mylopoulos, J., Chung. L., Nixon, B. A. & Yu, E. (2000). *Non-Functional Requirements in Software Engineering.* Massachusetts: Kluwer Academic Publishers

Nordbotten, J. C. (1985). *The Analysis and Design of Computer-Based Information Systems.* Boston: Houghton Mifflin Company

Ringdal, K. (2001). *Enhet og mangfold*. Bergen: Fagbokforlaget

Skagestein, G. (2005). *Systemutvikling – fra kjernen og ut, fra skallet og inn* (2.utgåve). Kristiansand: Høgskoleforlaget

Sommerville, I. (2004). *Software Engineering* (7.utgåve). Essex: Pearson Education Ltd

Trienekens, J.J.M., Bouman, J.J. & Van Der Zwan, M. (2004). Specification of Service Level Agreements: Problems, Principles and Practices. *Software Quality Journal, 12*, 43-56. Henta 25.april 2006 frå http://www.springerlink.com/media/2gurtlxuyk5ve6tvxceg/contributions/g/1/l/7/g1l7328304157836.pdf

Van Bon, J. (editor), Pieper, M. (editor) & van der Veen, A. (editor) (2004). *IT Service Management based on ITIL – an Introduction* (2.utgåve). Van Haren Publishing

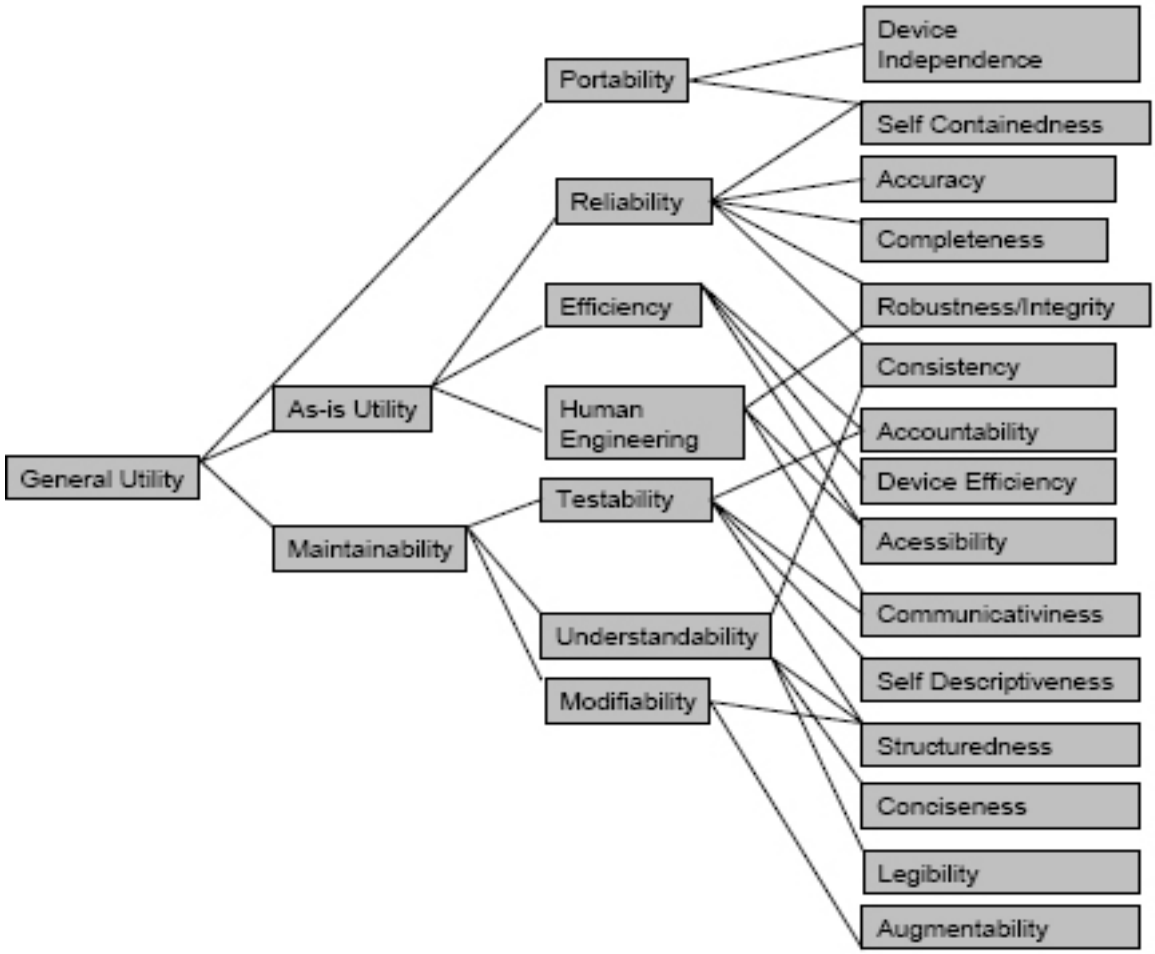
Whiteley, D. (2004). *Introduction to Information Systems: organizations, applications, technology and design*. New York: Palgrave Macmillan

Østbye, H., Helland, K., Knapskog, K. & Hillesund, T. (2002). *Metodebok for Mediefag* (2.utgåve). Bergen: Fagbokforlaget

Vedleggsliste

1. Barry Boehm sin modell av kvalitetskrav (1978)
2. Mylopoulos et. al si liste over ikkje-funksjonelle omgrep (2000)
3. ISO 9126 Quality Model (2001)
4. Somerville sin modell av ikkje-funksjonelle krav (2004)

1. Barry Boehm sin modell av kvalitetskrav



Boehm 1978

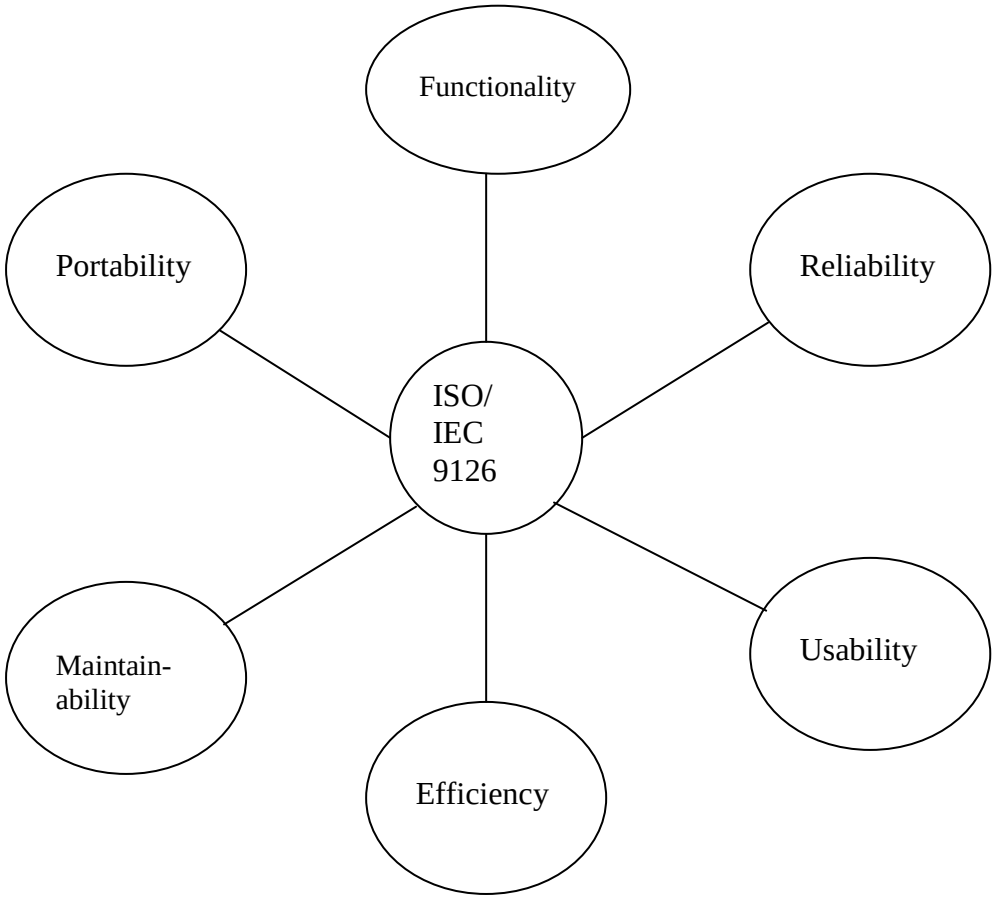
2. Mylopoulos et. al si liste over ikkje-funksjonelle omgrep

Ikkje-funksjonelle omgrep				
Accessibility	Accountability	Accuracy	Adaptability	Additivity
Adjustability	Affordability	Agility	Auditability	Availability
Buffer space performance	Capability	Capacity	Clarity	Code-space performance
Capacity	Clarity	Code-space performance	Cohesiveness	Commonality
Communication cost	Communication time	Compatibility	Completeness	Component integration cost
Component integration time	Composability	Comprehensibility	Conceptuality	Conciseness
Confidentiality	Configurability	Consistency	Controllability	Coordination cost
Coordination time	Correctness	Cost	Coupling	Customer evaluation time
Customer loyalty	Customizability	Data-space performance	Decomposability	Degradation of service
Dependability	Development cost	Development time	Distributivity	Diversity
Domain analysis cost	Domain analysis time	Efficiency	Elasticity	Enhanceability
Evolvability	Execution cost	Extensibility	External consistency	Fault-tolerance
Feasibility	Flexibility	Formality	Generality	Guidance
Hardware cost	Impact analyzability	Independence	Informativeness	Inspection cost
Inspection time	Integrity	Inter-operability	Internal consistency	Intuitiveness
Learnability	Main-memory performance	Maintainability	Maintenance cost	Maintenance time
Maturity	Mean performance	Measurability	Mobility	Modifiability
Modularity	Naturalness	Nomadicity	Observability	Off-peak period performance
Operability	Operating cost	Peak-period performance	Performability	Performance

Planning cost	Planning time	Plasticity	Portability	Precision
Predictability	Process management time	Productivity	Project stability	Project tracking cost
Promptness	Prototyping cost	Prototyping time	Reconfigurability	Recoverability
Recovery	Reengineering cost	Reliability	Repeatability	Replaceability
Replicability	Response time	Responsiveness	Retirement cost	Reusability
Risk analysis cost	Risk analysis time	Robustness	Safety	Scalability
Secondary-storage performance	Security	Sensitivity	Similarity	Simplicity
Software cost	Software production time	Space boundedness	Space performance	Specificity
Stability	Standardizability	Subjectivity	Supportability	Surety
Survivability	Susceptibility	Sustainability	Testability	Testing time
Throughput	Time performance	Timeliness	Tolerance	Traceability
Trainability	Transferability	Transparency	Understandability	Uniform performance
Uniformity	Usability	User-friendliness	Validity	Variability
Verifiability	Versatility	Visibility	Wrappability	

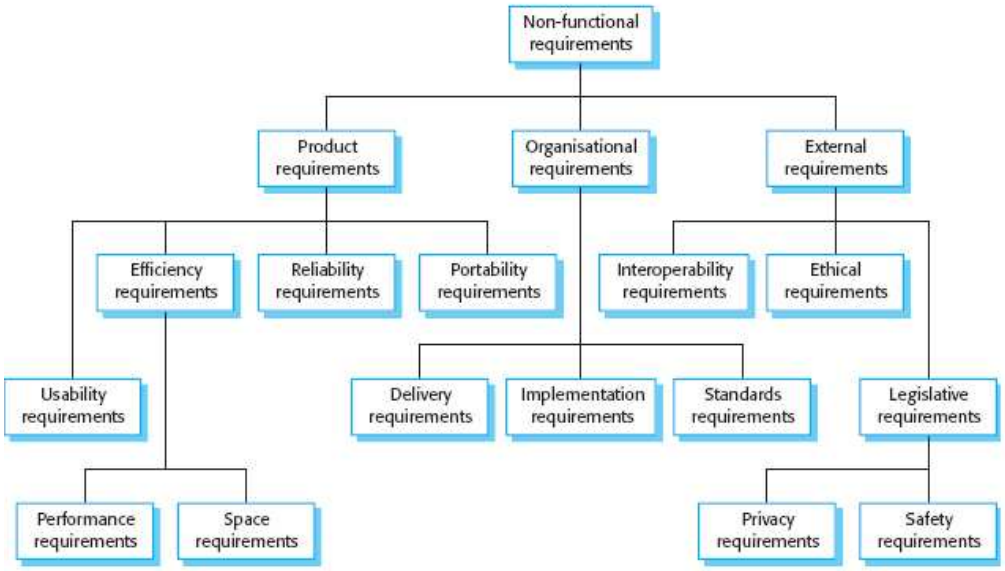
Mylopoulos et. al. 2000:160

3. ISO9126 Quality Model



ISO/IEC 2001

4. Somerville sin modell av ikkje-funksjonelle krav



Sommerville 2004