Jan Frode Haugseth

Forenklingens logikk

En studie av reproduksjonen av
profesjonell identitet i IKT-organisasjoner

Avhandling for graden philosophiae doctor

Trondheim, mai 2012

Norges teknisk-naturvitenskapelige universitet
Fakultet for samfunnsvitenskap og teknologiledelse
Institutt for sosiologi og statsvitenskap

NTNU
Det skapende universitet

NTNU
Norges teknisk-naturvitenskapelige universitet

Doktoravhandling for graden philosophiae doctor

Fakultet for samfunnsvitenskap og teknologiledelse
Institutt for sosiologi og statsvitenskap

ISBN 978-82-471-3514-3 (trykt utg.)
ISBN 978-82-471-3515-0 (elektr. utg.)
ISSN 1503-8181

Doktoravhandlinger ved NTNU, 2012:117

Trykket av NTNU-trykk

Sammendrag – Forenklingens logikk

En studie av reproduksjonen av profesjonell identitet i IKT-organisasjoner

I de siste 30 årene har informasjonsteknologier og programvare spilt en viktig rolle for vekst og innovasjon i moderne økonomier. Framveksten av "informasjonsalderen" – det "postindustrielle samfunn" eller "kunnskapssamfunnet" – har fått mye oppmerksomhet fra akademikere som har forsøkt å beskrive og teoretisere denne utviklingen. Et sentralt argument i mange fremstillinger er at informasjonsalderen endrer de sosiale forutsetningene for arbeid og organisasjon. Ulike forfattere har forklart kunnskapsarbeid som preget av en nettverkslogikk, assosiert med store, uformelle nettverk, aktivitet, kommunikasjon og konstant tilpasning. Selv om nettverksmetaforer er beskrivende for mange aspekter ved moderne kunnskapsarbeid, har denne typen arbeid også viktige aspekter som ikke kan relateres til idéen om nettverket. I løpet av 15 år har programmeringsfaget gått fra å åpne for fleksible arbeidsformer og mye overtid til å bli kontrollert og resultatfokusert. Dette har skjedd uten streiker eller at fagforeninger har vært involvert. Endringene har imidlertid sammenfalt med at det har vokst fram kritikker av IKT- og kunnskaparbeid fra minst tre ulike retninger: industriell effektivitet, kollektiv sikkerhet og velferd for arbeiderne og økonomisk vekst. Disse kritikkene har fått økt legitimitet blant profesjonelle utviklere. I kjølvannet av disse kritikkene har det vokst fram et nye profesjonelle idealer med referanser til et østlig prinsipp: *forenkling*. Ifølge dette prinsippet fremstår *nettverket*, det å *dele* arbeidsoppgaver, innhold og oppmerksomhet og bidra til å *øke informasjonsflyten* som lite verdifullt. I stedet skapes *kompleksitet*, *ineffektivitet* og en *uheldig balanse* mellom arbeid og privatliv. Profesjonell verdi skapes derimot av å *fokusere* oppmerksomheten, å *konsentrere* og å *slå av* e-post, sosiale medier og mobiler. Jeg kaller idealene som knyttes til dette prinsippet *forenklingens logikk* eller *den ukomplekse verdiverden*. Denne logikken legger noen føringer for hva som anses som rettferdig, og er et middel for å rangerere kolleger, relasjoner, handlinger og objekter. Ved å knyttes til forenkling som et øverste prinsipp, får industrielle idealer om effektivitet, ytelse og planlegging igjen legitimitet i IKT-organisasjoner. I denne avhandlingen viser jeg hvordan og hvorfor dette skjer gjennom to kvalitative studier. Den første er basert på en analyse av 23 dybdeintervjuer med programvareutviklere, og den andre er basert på en diskursanalyse av tolv bøker som omhandler *smidig programvareutvikling* og *prosjekthåndtering*. I den første analysen viser jeg hvordan idealene som henviser til forenkling blir brukt og posisjonert i forhold til andre profesjonelle idealer, mens den andre analysen beskriver forenklingens logikk i detalj. Begge analysene trekker på og bidrar til det pragmatiske rammeverket og handlingsteorien som er utviklet av Luc Boltanski og Laurent Thévenot. Strukturelle analyser blir ofte kritisert for å ha funksjonalistiske eller mekaniske forståelser av sosial handling. Ved å forbinde aktørenes forståelse og bruk av logikker til verdighet, åpner rammeverket opp for å male et univers av kulturelle endringer hvor sosiale handlinger er dynamiske og menneskelige aktører er både moralske og høyst levende.

Abstract – The logic of simplicity

A study of the reproduction of professional identity in ICT organizations

For the past 30 years, information technologies and software have played an important role in growth and innovation in modern economies. The emergence of the "information age" — the post-industrial society or Internet society — has drawn much attention from academics who have attempted to describe and theorize this development. A key argument in these works is that the information age changes the social prerequisites for work and organization. Different authors have explained knowledge work as reflecting a network logic, characterized by large, informal networks, activity, communication and constant adaptation. Although network metaphors are descriptive of many aspects of modern knowledge work, such work also has important features that cannot be explained in terms of this logic. In the space of 15 years, the programming profession has changed from being associated with flexible forms of work and a great deal of overtime to being controlled and focused on results. This has taken place without strikes or trade unions involved. The changes have coincided with a strong growth in social criticism of ICT and knowledge work as it unfolded in the 1990s from at least three different directions: industrial efficiency, the collective safety and well-being of the workers, and the growth of markets. Such criticism has gained legitimacy among professionals. The compromise that has developed between these critiques has emerged as a professional ideal rooted in eastern culture: *simplicity*. According to this ideal, connecting to a *network*, *sharing* awareness and content and contributing to the *information flow* do not create value, but rather lead to complexity, followed by inefficiency and an unhealthy work/life balance. Professional value is on the other hand created through *focusing* attention and *concentrating*, *turning off* e-mail, social media and cell phones. I term this value system *the logic of simplicity* or *the world of simplicity*. This logic guides what is considered fair, and provides a means to rank colleagues, relationships, actions and objects. By using simplicity as a higher common principle, industrial ideals relating to performance management and planning is gaining legitimacy in ICT organizations. I show how and why this change is unfolding through two qualitative empirical studies. The first is based on an analysis of interviews with 23 software developers, and the second is based on a discourse analysis of twelve manuals in *agile software development* and *project management*. In the first analysis I show how ideals referring to simplicity are used and positioned in relation to other important principles of professional practice, while the second analysis describes the logic of simplicity in detail. Both analyses draw upon and contributes to the pragmatic framework of social action developed by Luc Boltanski and Laurent Thévenot. Structural analyses is often criticized for having functional or mechanical understandings of social action. By linking the logic of the actors' understanding to the concept of worth, this framework paint a universe of cultural change where social action is dynamic and human actors are both moral and highly vivid.

A wealth of information creates a poverty of attention

Herbert A. Simon (1971:40)

Innholdsliste

Forord

Jeg skal bekjenne at da jeg startet dette doktorgradsløpet i 2007 var jeg ganske fascinert av programvareutvikling og *hackere.*[1] For å flytte til Trondheim og begynne ved NTNU sa jeg opp jobben i et Bergensfirma som blant annet solgte annonseplasser på Internett. Jeg hadde forskjellige arbeidsoppgaver og en viss grad av frihet i arbeidet. På eget initiativ lagde jeg et dataprogram som kunne fordele annonsetrafikk automatisk alt etter antallet annonser som var solgt, mellom en profesjonell, dyr leveringstjeneste i Nederland, og en fri programvareløsning hvor firmaet bare betalte for en server som lå i Norge. Dette var på ingen måte high skill-utvikling.[2] Men løsningen fungerte og den var langt billigere enn den opprinnelige. Personlig var utviklingen av dette programmet noe av det morsomste jeg så langt hadde gjort i arbeidslivet.

Da jeg fikk tilslaget på en åpen doktorgradsutlysning ved ISS, NTNU, var dette med en prosjektskisse om hackerkulturer og gaveøkonomier i moderne organisasjoner. Himanen og Castells beskrev i *The Information Society and the Welfare State: The Finnish Model* konseptet hackeretikk og hva dette betød for Finland. Jeg ville finne ut mer, og studere hvordan hackerkulturer manifesterte seg i profesjonelle organisasjoner. Jeg skulle sammenligne norske og finske organisasjoner. Da jeg gjorde intervjuene mine i norske organisasjoner fant jeg imidlertid ikke noen hackerkultur. Jeg møtte noen få mennesker med en grunnholdning som lignet idealtypen som Himanen beskrev, lidenskapelige og idealistiske. Men disse var overhodet ikke representative for utvalget mitt, og jeg ble vitne til noe jeg selv syntes var ganske sørgelig: I fortellingene sine beskrev de en fagverdighet og en arbeidskultur som var nesten umulig å realisere i organisasjonene de jobbet i. Historiene deres handlet om en identitet som var i oppløsning.

Jeg så samtidig at arbeiderne som trivdes i disse bedriftene hadde en lettere tilnærming til arbeidet, med mindre forventninger til at arbeidet skulle være så viktig og morsomt. Mange av dem realiserte seg på fritiden. De fleste brukte ordet "hacker" for å beskrive noe negativt. Mange

1 Med hacker mener jeg her personer med et lidenskapelig forhold til programmering, koding og datamaskiner.

2 Løsningen var hovedsaklig skrevet i programmeringsspråket PHP, og databasen var basert på lagring direkte i filsystemet.

av dem fortalte om noe de kalte *smidig utvikling*,[3] som jeg ikke hadde hørt om før. IKT-bransjen var i en periode preget av mye overtidsjobbing blant programmerere, men i utvalget mitt var det omtrent bare lederne som jobbet overtid. Jeg satt igjen med et inntrykk av at den typiske programvareutvikleren i mitt utvalg spilte golf, syklet den store styrkeprøven eller drev med seiling. Svært få drev på med programvare hjemme på kveldene, og svært få var opptatt av at de hadde et lidenskapelig forhold til jobben. Hvordan kunne dette forklares?

Mange personer har bidratt til at denne avhandlingen har blitt til. Jeg vil spesielt takke veilederen min Bente Rasmussen, som har holdt meg i tøylene, og bidratt til å utvikle og realisere prosjektet fra begynnelsen. Jeg vil takke Gisle Andersen, for kjappe, men svært verdifulle kommentarer til tidlige utkast av analysene og gode tips til viktig lesestoff. Jeg vil også takke Rune Sakslind, min gamle veileder fra da jeg tok hovedfag ved UiB, som for lenge siden hjalp meg å utvikle skissen til dette prosjektet. Jeg vil takke Gunhild Tøndel som har bistått med korrekturlesing i sluttfasen av skrivingen. Ingeborg Grønning og Børge Haugset var med på å skrive en tidlig analyse om informantene som kalles Fridtjof og Anders, og Børge hadde en del litteraturtips hva angår smidig utvikling. Jeg vil også takke bedriftene jeg besøkte, og informantene som viste meg tillit og delte sin kunnskap med meg.

Ellers har mange mennesker kommet med tilbakemeldinger på prosjektet i ulike fora iløpet av disse fire årene. Jeg har mistet oversikten – men takk skal dere ha, alle sammen! Til slutt vil jeg takke kona mi Maren – som gir meg alt jeg trenger, inkludert en del motstand. Aller sist vil jeg takke Maja (4) som har skrevet inn navnet sitt selv, og som hver dag minner meg om hva lidenskap, eksperimentering og indre drivkrefter faktisk betyr for mennesket.

3 Smidig utvikling er et samlenavn på ulike metodologier knyttet til IKT-yrker.

1. En innledning om forenkling

Idet jeg skriver denne innledningen midt i august 2011, foregår det en stor tørke i hele den østlige regionen av Afrika. Tørken har skapt en akutt matkrise i Somalia, Etiopia og Kenya. Verst har det gått utover Somalia. FN erklærte hungersnød i to regioner sør i Somalia 20. juli. Den 3. august ble ytterligere tre regioner erklært rammet. Mange flyktninger har tatt seg inn i Kenya og Etiopia, hvor det også mangler mat og sanitæranlegg. Så langt er det flest somaliere som har omkommet, og dødeligheten er størst blant barn under fem år. Omfanget av krisen er ventet å øke utover høsten (Fewsnet 2011).

Det er første gang på 30 år at FN har erklært hungersnød. På disse 30 årene har verden forandret seg med hensyn til hvordan informasjon om denne typen kriser kan formidles. Flere ledende sosiologer har postulert at verden i dag ligner et nettverk, og at informasjon og kommunikasjon foregår kontinuerlig (Dijk 2005; Castells 2009). Internett og mobile tjenester har gjort at det er enklere å gi individuell, økonomisk støtte til mennesker i humanitære katastrofer via frivillige organisasjoner enn før. Man kan for eksempel sende en SMS til Røde Kors eller ringe til Care, og et bidrag blir automatisk øremerket formålet og trukket fra telefonregninga. I Norge sammenfaller imidlertid de nye mulighetene til å gi med at den generelle interessen for bistandsspørsmål har avtatt (Wilhelmsen 2010).

Det største sosiale nettverket i verden per i dag heter Facebook. I juli 2011 rapporterte Facebook at de hadde 750 millioner aktive medlemmer på verdensbasis (Rao 2011). Selv om jeg er Røde Kors-fadder og har personer som jobber for Røde Kors på kontaktlista mi på Facebook, ser jeg svært sjelden oppdateringer om Afrikas Horn på veggen[4] min. Dette er ikke fordi mine kontakter ikke poster informasjon om dette (jeg har sjekket), men fordi innholdet jeg får se *filtreres* etter *personlig relevans*. Google, Yahoo og mange andre aktører gjør det samme med søketreff. Dette betyr at jeg, hvis jeg som bruker ikke aktivt endrer innstillingene, bare får se informasjonen som tjenestetilbyderne mener er mest interessant for meg. Filtrene som beregner hva som er relevant er helautomiserte. Det jeg får se eller ikke se beregnes ved hjelp av algoritmer på tjenestetilbyderens servere. Innholdet på Facebook-veggen og søkeresultater blir beregnet ut

4 Facebook-veggen er som en digital oppslagstavle, hvor beskjeder, innlegg, bilder og annet innhold blir vist.

fra hva brukeren vanligvis klikker på, hvor lang tid man bruker per lenke og mange andre variabler (Pariser 2011). Hvis man vanligvis liker å diskutere lenker eller dele og kommentere bilder, er sjansen mindre for at man får opp informasjon om humanitære katastrofer. Det samme gjelder mennesker som til vanlig foretrekker å se musikkvideoer. Disse digitale filtrene kan sees i lys av en uttalelse av gründer, programvareutvikler og konsernsjef i Facebook, Mark Zuckerberg:

> *A squirrel dying in front of your house may be more relevant to your interests right now than people dying in Africa (Kirkpatrick 2010:296).*

Ved å gi brukerne mer av det de liker å se, håper han at brukerne skal trives så godt at de kommer tilbake til Facebook igjen senere. Pariser kaller resultatet av slike filter en "filter bubble" – teknologien skaper en virtuell boble rundt brukerne. Slik skjermes de for synspunkt som kontrasterer det innholdet de kjenner fra før. Når Zuckerberg legitimerer disse filtrene anvender han en logikk som tilsier at det å filtrere og fjerne informasjon er til brukernes beste. Her anvender Zuckerberg det jeg kaller en *ukompleks logikk.* I denne avhandlingen argumenterer jeg for at verdisystemet bak denne logikken kan forklare viktige aspekter ved samhandling, koordinasjon og identitetsformasjon i moderne IKT-organisasjoner.

Det moderne samfunnet er preget av et overskudd av informasjon. Informasjonsvolumet i sosiale medier, i mediebildet og samfunnet generelt, skaper en kollektiv tilstand av det Simon (1971:40) omtalte som "poverty of attention". Mennesker *kan* ikke ta stilling til alt. Den ukomplekse logikken hevder at ved å fokusere og fjerne mindre relevant informasjon, økes verdien på budskapet som står igjen. Logikken er knyttet til en sentral, kognitiv begrensning hos mennesket: Sinnet har ikke kapasitet til å håndtere alle inntrykk likt. Når vi passerer en grense blir selve informasjonsmengden et problem – som krever filtrering eller en aller annen håndtering.

I flere tilfeller opp gjennom historien har idealer som fremhever det ukomplekse blitt brukt som et *verdi- og normsystem* for å legitimere at noen typer praksis er mer verdifull enn annen praksis. Et eksempel på dette er læren forbundet med Buddhismen. Ifølge Buddhismen kan en person komme nærmere frelsen (Nirvana) gjennom *jhāna*; å fjerne selve behovet for ting, anerkjennelse og objekter gjennom meditasjon (Armstrong 2001:70-72). Religiøs askese fordrer ofte et liv i åndelig og materiell enkelhet, og opp gjennom historien har mange hevdet at spirituell inspirasjon viste veien mot en enklere (og derfor bedre) livsstil. Frans av Assisi, Leo Tolstoj og Mahatma Gandhi er noen av disse (Shi 2001). Henry Thoreau (2008) forfektet en sekulær versjon av det enkle og bærekraftige livet i boka *Walden*, etter å ha bodd i en lengre

periode i en hytte langt unna resten av sivilisasjonen. Thorstein Veblen (2004) advarte mot faren ved å gjøre økt forbruk[5] til en symbolsk markør for rikdom i *The Theory of the Leisure Class*, og argumenterte for nødvendigheten av, og *verdigheten* i, en ukompleks tilnærming til livet. Richard Gregg (1936) skrev om verdien av "voluntary simplicity" i *The value of voluntary simplicity*, hvor han trakk grenser mellom ufrivillig fattigdom og det han beskrev som en berikende, frivillig, ukompleks tilværelse. Økonomen Ernst Schumacher (1973) brukte lignende perspektiver i essaysamlingen *Small is beautiful*, hvor han tok utgangspunkt i at den moderne økonomien ikke var bærekraftig på sikt. Gjennom dette problematiserte han blant annet det å måle velferd ved å se på en nasjons BNP, og foreslo at det i et lengre perspektiv vil være riktigere å se maksimum velferd ut fra et minimum av forbruk.

Sentralt i denne avhandlingen står det *pragmatiske* rammeverket som ble utviklet av Boltanski og Thévenot, og som som tar hensyn til rollene *verdighet*, *sosial kritikk* og *legitimering* spiller i utviklingen av det moderne samfunnet. For å mobilisere, overbevise, eller vinne diskusjoner, er aktørene nødt til å forme argumenter med referanse til noe som flere oppfatter som allment godt. Det finnes imidlertid et *mangfold* av "gode" idealer, og aktørene står fritt til å henvise til så mange idealer som de ønsker, eller sette dem sammen til ulike konfigurasjoner. Aktørene kan realisere idealene på både mikro- og makronivå. En konsekvens av dette rammeverket er forståelsen av at det moderne samfunnet aldri kan forklares ut fra én dominant ideologi, én produksjonsmodus eller én delt fellesskapsfølelse i Durkheimsk forstand, eller én sosial offentlighet som hos Habermas. Boltanski og Thévenot (2006) argumenterer derimot for at det eksisterer minst seks adskilte verdiverdener, som utgjør modernitetens legitime evalueringsrepertoar.[6] Disse kalles den *inspirerte*, den *hjemlige*, den *kollektive*, verdenen for *berømthet*, *markedsverdenen* og den *industrielle* verden. Det finnes ikke noe endelig antall verdiverdener, men disse seks blir trukket fram som spesielt stabile, historiske konstruksjoner i kildene, som er skrevne manualer, guider og håndbøker for ledelse (Boltanski og Thévenot 2006:149-158). Ut fra en diskursanalyse av management-bøker gitt ut på 60- og 90-tallet, forklarer Boltanski og Chiapello (2005) 90-tallets kapitalisme som preget av en syvende verdiverden, den *prosjektorienterte* verdenen (the projective city) – også kalt *nettverkslogikken*. Denne er kjennetegnet av at verdi regnes ut fra tilknytningen til store, uformelle nettverk og evnen til å hele tiden være i aktivitet, til å kommunisere og stadig tilpasse seg.

Jeg argumenterer for at den prosjektorienterte verdenen allerede siden midten av 90-tallet – i

5 Begrepet han brukte, *conspicuous consumption*, betyr direkte oversatt prangende eller synlig forbruk.

6 Evalueringsrepertoar består av ulike kulturelt betingede forståelser, verdier og idealer. De kan være lokale og globale, og bestå av flere ulike klassifiseringskategorier (hos Boltanski og Thévenot typisk kalt verdiverdener).

hvert fall i IKT-organisasjoner – har blitt begrenset av en åttende verdiverden. *Den ukomplekse verdiverdenen*, eller *forenklingens logikk*, består av en egen grammatikk, og en egen særegen måte å rangere og bedømme subjekter og objekters *verdighet* på. I den ukomplekse verdiverden er det høyeste prinsippet basert på *enkelhet* og *klarhet*. Prinsippet knyttes igjen til et allment gode gjennom at man *frigjør* plass eller tid, eller gjør at noe annet blir *enklere*, *klarere*, *mer tilgjengelig* eller *gjennomførbart*. Verdiverdenen utfyller universet av verdiverdener som er beskrevet av Boltanski og Thévenot (2006) og Boltanski og Chiapello (2005).

2. Avhandlingens argument og struktur

2.1 Innledning

De siste 50-års sosioøkonomiske endringer er beskrevet som et brudd med tidligere tiders produksjonsformer. Ulike navn har blitt brukt: Postkapitalisme (Dahrendorf 1959), postindustrialisme (Bell 1974), informasjonssamfunn (Stonier 1983; Bell 1976), kunnskapsøkonomi (Reich 1991), "informationalism" og nettverksamfunn (Castells 2009). Til felles for beskrivelsene er at de nye produksjonsformene bryter med Fordisme og Taylorisme, og at de er mer inspirerende og tilfredsstillende enn samlebåndsarbeidet (Baldry et al. 2007:27–28). Det er ikke bare produksjonsformene som er endret, selve *meningen* med arbeidet har fått nytt innhold. Castells og Himanen argumenterer for fremveksten av en ny arbeidsetikk som de kaller *the hacker ethic*,[7] hvor arbeidet først og fremst forstås som kreativt, personlig motivert, inspirerende og tilfredsstillende (Himanen 2001; Castells og Himanen 2002). Denne forståelsen er igjen koblet mot at tradisjonelle byråkratiske modeller har måttet vike for nye, *fleksible* former for organisering i mange vestlige organisasjoner. Den viktigste motivasjonskilden for profesjonelle IKT-arbeidere på 90-tallet var ifølge Himanen idealer som i økende grad posisjonerte seg imot Webers beskrivelser av den protestantiske arbeidsetikken (2001). Ifølge Himanen dominerte denne gamle formen for arbeidsetikk fremdeles i Finland på 90-tallet. Den protestantiske etikken gjenspeilte seg ifølge Himanen i sju verdier: Penger, arbeid, fleksibilitet (at organisasjonener tilpasset seg og profitterte på den verdenen de var en del av), stabilitet, målorientering og ansvar for resultatet (Himanen 2001:111–129, 139). Selv om Finland allerede hadde blitt kjent for suksess på IKT-området, skrev

7 Meningen av ordet hacker er i dag omstridt. For enkelte innad i programmeringsbransjen har ordet positive konnotasjoner. Massemedia bruker som oftest hacker for å beskrive folk som har begått datainnbrudd eller datakriminalitet. I norsk akademia har hacker-begrepet blitt brukt som et utgangspunkt og et begrep innen kjønnsforskning og IKT (Berg 2000; Corneliussen 2002, 2003; Gansmo 2002a, 2002b; Langsether 2001; Håpnes and Rasmussen 1991; Stuedahl 1997). Lagesen (2005) skriver at begrepsbruken i norsk akademisk sammenheng minner om Sherry Turkles hackerdefinisjon fra 1984: En hacker er en asosial, mannlig ungdom, som er "besatt" av datamaskiner (Lagesen 2005:82, 111). Lagesen putter denne denne hackerdefinisjonen i samme bøtte som "nerd" og "geek".

Himanen (2001:125-126) at disse sju idealene til Weber også kunne beskrive nettverkssamfunnet som dette ble fremstilt i Castells (2009). Hackeretikken, som Himanen først og fremst observerte blant IKT-arbeidere og akademikere (Himanen 2001:6-8, 71), men som var tilgjengelig for hvem som helst som var motiverte av lidenskap og kreativitet i arbeidet, utgjorde en alternativ arbeidsetikk for det moderne samfunnet, en "spirit of informationalism". Himanen så for seg at hackerlogikken ville kunne utkonkurrere den protestantiske etikken i det moderne informasjonssamfunnet.

Hackeretikken gjenspeiler seg i sju idealer: Lidenskap, frihet, sosial verdi, åpenhet, aktivitet, omsorg (caring) og kreativitet (Himanen 2001:139). *Lidenskap* er definert til å bety at motivasjonen for arbeidet har en indre forankring, og gir glede når det realiseres. Hackere organiserer ikke livene sine etter rutiner og ideer om å optimalisere arbeid, og *frihet* oppnås gjennom å følge sin lidenskap. *Sosial verdi* og *åpenhet* motiverer for å delta i arbeidet, bidra til fellesskapet og dele kunnskap. Aktivitet betyr frihet til å uttrykke seg gjennom handlinger, og å trekke grenser mot passivt konsum til fordel for å forfølge ens lidenskap. I *omsorg* ligger det å hjelpe andre som trenger det, og ikke minst å hjelpe andre til å bli del av og å få utbytte fra nettverket (Himanen 2001:141). *Kreativitet* blir definert som å bruke av egne evner til å "overgå seg selv", og skape verdifullt innhold som kan gis til "verden" (Himanen 2001:139-141). Himanens beskrivelser av arbeidsetikk i samtiden var ikke enestående. Nettverket, snarere enn firmaet, er det som karakteriserer 90-tallet, hevder Castells (2009). I dette bildet blir byråkratiske styringsmodeller sett på som utdaterte og rigide, og i stedet erstattet av "kulturell koordinering", internalisert dedikasjon og selvdisiplin (Thompson 2003). Arbeideren opplever mer ansvar og kontroll over sin egen arbeidsutøvelse. Frenkel et al. (1999:71, 141) fant at dette også gjaldt såkalte front-line-arbeidere, som jobber i skjæringspunktet mellom kunnskapsformidling og service. Standardisering og direkte kontroll er mindre utbredt enn tidligere, og ledere legger mindre vekt på å måle ytelsen til arbeiderne.

Boltanski og Chiapello (2005) leverte med sin bok *The New Spirit of Capitalism* sitt bidrag til å forstå denne transformasjonen. De argumenterer for at overgangen til informasjonsalderen krevde en økt innsats og investering fra arbeiderne som gjenspeilte seg med hensyn til varighet og intensitet i arbeidet. Når alle ble mer involvert, ble det nødvendig å utvikle et språk og et verdisystem sånn at man kunne vurdere sin egen og sine kollegers innsats, og ikke minst rettferdiggjøre overfor andre at man brukte mye tid på jobben. Da kunnskapsarbeiderne fikk mer ansvar, fikk de også en større dedikasjon og en følelse av forpliktelse i forhold til arbeidet og prosjektene de var en del av – endringene var ikke påtvunget arbeiderne (Boltanski og Chiapello 2005:8). Hackeretikken, "the spirit of informationalism", (Himanen 2001; Castells og Himanen

2002) alluderer til "the new spirit of capitalism" (Boltanski og Chiapello 2005) gjennom at subjektet fremstilles som individualisert, personlig motivert, kreativt og prosjektorientert. Førstnevnte ånd er imidlertid mer normativ, mens sistnevnte er del av rammeverket som jeg benytter gjennom avhandlingen, og som jeg presenterer i detalj i kapittel 4.

Forståelsen som ligger til grunn for *The New Spirit of Capitalism* er basert på at kapitalismen er et ustabilt system med ubegrenset behov for akkumulasjon, et grunnsyn Boltanski og Chiapello deler med Marx. Men i motsetning til marxister er ikke Boltanski og Chiapello opptatt av at kapitalismen opprettholdes gjennom klassekamp og klassemotsetninger, men heller at det er aktørenes egne legitimeringer som opprettholder og vedlikeholder systemet: Ved at kapitalismen har både tålt og tilpasset seg i tråd med kritikk fra f.eks. nasjonalstatene hver gang systemet har vært i krise, har det kapitalistiske systemet blitt gjort legitimt – dermed forstås også systemet som legitimt. Dette perspektivet står ikke i opposisjon til marxistiske eller bourdieuske kritiske sosiologiske handlingsmodeller; man kan heller si at de utfyller hverandre (Benatouil 1999; Silber 2003; Wagner 2001). For å forbli attraktiv, må kapitalismen få legitimitet fra eksterne ressurser – forståelser, verdier og idealer – som i kritiske historiske situasjoner og kriser kan bidra til å forandre og endre den (Boltanski og Chiapello 2005:20). Gjennom dette rammeverket åpnes dermed muligheten for å forstå den sosiale transformasjonen av modernitetens produksjonsformer som en kontinuerlig offensiv preget av et mangfold ulike forståelser, ikke som et brudd.

Sosiale situasjoner som krever legitimering kan ha mer eller mindre tydelige referanser til en generell forståelse av et felles gode eller en rettferdighet som hevder å ha universell gyldighet. Disse ulike forståelsene av hva som er godt, er i Boltanski og Thévenots rammeverk modellert som ulike verdiverdener. Henvisningen til verdiverdener gjelder på ulike handlingsnivåer, og inkluderer derfor også sosiale system av størrelsesordenen *kapitalismen*. Etter hvert som gamle legitimeringer og forståelser av kapitalismen ble oppfattet som utdaterte, bidro kritikere til å lansere nye prinsipper som styrte hva som ble oppfattet som viktig og ikke i samfunnet. Boltanski og Chiapello beskriver framveksten av en ny, prosjektorientert verdiverden, med utgangspunkt i epoken som begynte på 70-tallet. Kildematerialet hos Boltanski og Chiapello besto av to tekst-korpus på tilsammen 126 mindre håndbøker fra 60- og 90-tallet. I sistnevnte er den overordnete formen for verdi basert på aktivitet, prosjektinitiativtaking og nettverk. Verdige subjekter er ledere, disponenter og innovatører. Objekter er datamaskiner, informasjonsteknologier og nye organisatoriske innretninger. Det overordnede talentet en verdig person bør ha er å koble sammen punkt (mennesker, prosesser, ting) i nettverket.[8] Kritikkene som denne verdiverdenen er

8 For en komplett liste over grammatikken i prosjektverdenen, se metodekapitlet s. 83.

basert på, er formet rundt det sosiohistoriske tankegodset som spesielt kom til uttrykk hos 68-generasjonen. Arbeidsgivere på 90-tallet brukte blant annet språket til motkulturene på 1968-tallet for å legitimere nye arbeidspraksiser og skape mer subtile og suksessfylte former for å utnytte arbeidskraften. 1968-ernes *artistiske* kritikk av industrialismen og samlebåndsarbeidet gikk på at systemet var uverdig og urettferdig, fordi det var uestetisk, kjedelig og gjorde mennesker triste og ulykkelige, samtidig med at respekten for autoritet og hierarki minket. Den artistiske kritikken fikk støtte fra en industriell kritikk, som hevdet at epokens arbeidsorganisering ikke lenger var optimal. Fordismen var rett og slett ikke lenger spisbar. Kritikkene bidro til å endre systemet innen noen klarte å lansere en kritikk som forsvarte denne typen produksjonsregimer.[9] Svaret ble en ny type prosjektorientert logikk, utviklet i et *kompromiss* mellom den artistiske og den industrielle verdenen, med løsninger som fleksibel spesialisering, mer autonomi til arbeiderne og prosjektorganisering. Siden den "nye kapitalistiske ånden" hadde tilpasset seg mye av den artistiske kritikken som ble rettet mot kapitalismen på slutten av 60-tallet, virket kritikk som henviste til størrelser som individuell frihet, autonomi og autentisitet malplassert og ute av kontekst (Boltanski og Chiapello 2005:419).

Argumenter med røtter i den prosjektorienterte verdiverdenen blir gjerne brukt for å legitimere en virkelighet hvor man har mange svake bånd til subjekter i et stort *nettverk* av kontakter. Verdighet reguleres ikke av profesjonell eller faglig stolthet i seg selv, men gjennom deltakelse, evne til å knytte kontakter og danne større nettverk. Ifølge prosjektverdenen mister subjekter verdi når de ikke har nettverkspotensiale, når deres eget nettverk ikke lenger gjør dem interessante. Samtidig med at prosjektverdenen har fått økt gyldighet, har verdigheten i familiære bånd og lojalitet mistet mye av sin kraft (Boltanski og Thévenot 2006:164). I Boltanski og Chiapellos framstilling er denne nye, prosjektbaserte "kapitalistiske ånden" brukt for å legitimere akkumulasjonsprosessen i kapitalismen, og samtidig for å begrense den og gi den en retning (Boltanski og Chiapello 2005:xx).

2.2 Mitt argument

Selv om hacker- og nettverksmetaforer er beskrivende for mange aspekter ved moderne kunnskapsarbeid, er det også vesentlige trekk ved det moderne arbeidet som ikke kan forklares av denne logikken. Mange har påpekt at denne nye "ånden" ikke er utbredt på generell basis i samfunnet (se f.eks. Ritzer 2004) eller blant alle kunnskapsarbeidere (Baldry et al. 2007). Mitt

9 Det skal riktignok bemerkes at samlebåndsproduksjonen gradvis ble flyttet ut av vesten og til fattigere land etter hvert som de "fleksible" moderniseringsoffensivene ble gjennomført. Boltanski og Thévenot modell tar høyde for at den kritiserte relasjonen eller det kritisert objektet skjules – og orden gjenopprettes. *Out of sight, out of mind.*

poeng er imidlertid at logikken som preger beskrivelsene av informasjonssamfunnet, den prosjektorienterte verdiverden eller nettverkslogikken, har begrenset legitimitet i IKT-organisasjoner og IKT-faget. Hvordan skal vi for eksempel forstå den raske framveksten av metodikker for regulering og resultatorientering i den kreative programvarebransjen[10] – midt i knutepunktet for alle verdens nettverk? Hvorfor blir muntlig kommunikasjon regnet som overlegen skriftlig blant profesjonelle IKT-arbeidere, og hvorfor er planleggingssystemene som brukes i verdens største IKT-organisasjoner basert på post-it-lapper?[11]

På 15 år har programmeringsfaget endret seg fra å være forbundet med fleksible arbeidsformer og mye overtid, til å bli mer regulert og resultatorientert. Dette har foregått uten tydelig eksternt press fra ledelse eller eiere, uten sosialt opprør, streiker eller involverte fagforeninger. Metodologier og nye styringssystemer spiller en del-rolle, men de største endringene er at sosiale kritikker av IKT- og kunnskapsarbeid slik dette utspilte seg på 90-tallet har vokst seg sterk og skapt et krav om disiplinering av yrkesutøverne. Dette kan observeres i nyere management-litteratur, men også i organisasjoner hvor de nye metodologiene ennå ikke er utbredt. I kompromisset som oppstår mellom disse kritikkene har det vokst fram et profesjonelt ideal med røtter i østlig kultur: *Enkelhet*. Å knytte seg til et *nettverk*, *dele* oppmerksomhet og *innhold* samt bidra til *mer informasjonsflyt* gir ikke verdi ifølge dette idealet, snarere motsatt. Verdi skapes gjennom *balanse*, *sykluser*, å *skru av* e-post, mobil og sosiale medier, *å konsentrere* seg om arbeidet og *fokusere*. Jeg kaller verdisystemet som følger idealet om enkelhet *forenklingens logikk* eller *den ukomplekse verdiverdenen*.

Ekvivalensprinsippet i den ukomplekse logikken – det øverste prinsippet som alt og alle kan måles etter – knyttes alltid til et allment gode gjennom at man frigjør plass, eller at ting blir *tilgjengelig* og *gjennomførbart*. Høy verdi tillegges dermed mennesker som fatter seg i korthet, og som klarer å bryte ned komplekse sammenhenger til enkle "sannheter", teknologier eller prosesser som gjør livet mindre komplisert. Logikken kan som regel forstås relasjonelt, *enklere* eller *mindre komplisert* enn noe annet. Det enkle er også et mål i seg selv hvis ting kan gjøres *selvforklarende*. Den ukomplekse logikken åpner for en legitim mulighet til å isolere, stenge alt annet ute, om nødvendig både helheten og detaljer – sånn at man kan fokusere på nåtiden. Å ha oppmerksomheten på det nære trekkes fram som essensielt for å kunne eksistere på sikt. Mens

10 Her sikter jeg til utbredelsen av smidig utvikling (Laanti, Salo, og Abrahamsson 2011; Giblin, Brennan, og Exton 2010), brukt av blant annet Google, Nokia, Microsoft mfl.

11 Her sikter jeg til planleggingskonseptet brukerfortellinger (user stories/kanban) (Beck og Andres 2004; Poppendieck og Poppendieck 2009). Jeg presenterer systemet i detalj som del av analyse 2 i kapitlet som begynner på side 171.

andre fremstillinger av forenkling av arbeidet (for eksempel Taylorisme) gjerne representerer forståelser knyttet til management, styringssystemer eller eksterne prosesser som er innført for å disiplinere arbeiderne, representerer forenklingens logikk en måte å forklare hvordan og hvorfor arbeidstakere selv knytter prinsipper om forenkling til verdighet og stolthet i fag- og arbeid.

2.2.1 Forskningsprosess og utvikling av forskningsspørsmål

I den opprinnelige beskrivelsen av dette prosjektet, kalt *Kunnskap og kreativitet i IKT-samfunnet: En ny digital moral?*, planla jeg en komparativ intervjuanalyse av Norge og Finland i et kultursosiologisk perspektiv. Problemstillingene mine dreide seg rundt definisjoner av god, profesjonell fagutøvelse og hvordan hackeretikk, fri programvare og gaveøkonomier manifesterte seg i organisasjonskulturer av ulik størrelse og på ulike felt. Jeg var kjent med Boltanski og Thévenots rammeverk fra arbeidet med hovedoppgaven, og jeg hadde tenkt å bruke rammeverket i analysen. Grunnen til dette var en idé om at dette rammeverket ville kunne gi ny sosiologisk innsikt i forhold til feltet, at profesjonelle idealer trenger å bli gylddiggjort eller legitimert for å kunne bli utbredt. På forhånd antok jeg at programvareutvikling var et fag i "profesjonell utvikling", et fag med mange spenninger. Jeg anså det som sannsynlig at dette ville være et felt hvor det foregikk mange "forhandlinger" – hvor det ikke var gitt hva som ble ansett som god eller verdig jobbpraksis. Intervjuguiden ble skrevet med tanke på å få informasjon om grensetrekkinger og definisjoner av god fagpraksis, samt spørsmål om gaveøkonomier, forpliktelse og biografi.

Jeg hadde ikke hørt om *smidig utvikling* før jeg intervjuet Anders, den første informanten jeg møtte. Jeg oppdaget iløpet av de femten første intervjuene at utviklerne som ofte henviste til hackeridealer eller hackeretikk uttrykte et savn. Samtidig fant jeg at idealene jeg nå kaller *ukomplekse* preget mange refleksjoner hos informantene, og de personene som trakk frem viktigheten av disse så ut til å være både suksessfulle og fornøyde. Dette var uventede funn, som isolert sett ikke kunne forklares ut fra teorier om hackeretikk eller de sju beskrevne verdiverdenene i Boltanski og Thévenot (2006) og Boltanski og Chiapello (2005). Jeg forkastet ideen om å reise til Finland i samråd med veileder, og jobbet ut fra en hypotese om at det eksisterte et verdisystem eller en logikk som eksplisitt verdsatte forenkling, isolasjon og det å stenge ute. Underveis i analyseprosessen framsto det som nødvendig å finne ut om disse idealene var så tydelige at de tok form som et koherent verdisystem, og det kunne jeg ikke se tydelig nok ut fra intervjuene mine.

Jeg leste derfor tre bøker i smidig utvikling, og gjorde en manuell analyse som støttet hypotesen min. Senere utvidet jeg korpuset til 12 bøker, og gjennomførte diskursanalysen ved

hjelp av analyseverkøyet Tropes. Jeg vil anmerke at ikke alle informantene eller organisasjonene brukte såkalte smidige metoder på det tidspunktet jeg utførte intervjuene, men de kunne likevel referere til ukomplekse idealer. Disse idealene var altså allerede en del av deres profesjonelle evalueringsrepertoar.

Med avhandlingen søker jeg dermed å besvare to forskningsspørsmål. Hvilke idealer og verdier er viktige for å gi en pragmatisk forklaring på reproduksjonen av profesjonell identitet blant programvareutviklerne? For å kunne belyse dette første spørsmålet var jeg nødt til å finne ut hvor idealer med henvisning til et øverste prinsipp om forenkling kom fra. Mitt andre forskningsspørsmål er dermed: Hvordan ser et grammatisk oppsett av den ukomplekse verdiverden ut, og hvilken rolle har denne i fagutøvelse og dannelsen av faglig identitet? Den siste problemstillingen var ikke del av det opprinnelige prosjektet. Et gjennomgående tema i den første analysen er å vise hvordan ukomplekse idealer blir brukt og posisjonert i forhold til andre viktige prinsipper i fag- eller profesjonsutøvelse, mens den andre analysen rendyrker trekkene ved det ukomplekse.

2.2.2 Sammendrag av funn

Avhandlingen presenterer to originale bidrag knyttet til pragmatisk handlingsteori. Den første analysen studerer hvordan *profesjonell identitet* blir reprodusert blant programvareutviklere. Den andre analysen er et teoretisk-metodisk bidrag, identifikasjonen av *den ukomplekse verdiverden*, som utvider Boltanski og Thévenots univers av verdiverdener. Analysene er forankret i to empiriske undersøkelser, en kvalitativ analyse av intervjuer med 23 programvareutviklere og en diskursanalyse av tolv håndbøker i *smidig programvareutvikling* og *prosjektmanagement*. Studieobjektet er legitimeringer, argumenter, uttalelser og idealer som gjelder i fagutøvelsen samt biografier av profesjonelle IKT-utøvere. Analysene er nært knyttet sammen. I *Analyse #1 – den gode programvareutvikleren* utforsker jeg hvordan programvareutviklerne posisjonerer former for *faglig verdighet* opp mot hverandre når de snakker om arbeidspraksiser og betydningen av arbeid. Hva som er god og verdifull faglig praksis er ikke gitt, og verdigheten i arbeidet kan bare reproduseres gjennom at den tåler eller tilpasses kritikk. Analysen bidrar til en dynamisk forståelse av profesjonell eller yrkesfaglig identitet, som tar hensyn til aktørenes rolle i dens utvikling, samt betydningen av endringer i teknologi, organisasjon og samfunnsbetingelser. Analysen tegner opp fire faglige verdigheter som inneholder henholdsvis *hackeridealer*, *ingeniøridealer*, *hybride idealer* og *smidige* idealer. I sistnevnte er forenkling, fokus og det å isolere viktig, og jeg viser at disse idealene ser ut til å reprodusere en *faglig verdighet* som spiller på lag med både organisasjonenes behov og mange arbeidstakeres behov for et liv utenfor jobben.

Verdien av indre motivasjon, lidenskap og sosialt nettverk på jobben nedtones, og det blir lagt mer vekt på metoder og det å ikke spre produksjon utover for mange team.

I *Analyse #2 – den ukomplekse verdiverdenen* setter jeg sammen idealene knyttet til forenkling til en hittil ubeskrevet verdiverden, den ukomplekse verdiverden, tydeliggjort gjennom kritikker av IKT-faget. Jeg viser at forenkling er et viktig prinsipp som brukes for å realisere en ny, og "bedre" praksis i utvalget, og skisserer hvilke kritikker denne støtter seg på, og hvordan denne ser ut i henhold til Boltanski og Thévenots grammatiske oppsett.

Avslutningsvis viser jeg hvordan «ukomplekse» idealer, i hvert fall i IKT-organisasjoner, bidrar til å begrense den prosjektorienterte verdiverdenens (Boltanski og Chiapello 2005) gyldighet gjennom et kompromiss med den industrielle verdiverden, som jeg kaller det "smidige" kompromisset. Ved at forenkling brukes som et øverste ekvivalensprinsipp, har industrielle idealer knyttet til blant annet målstyring og planlegging igjen fått legitimitet i IKT-organisasjoner. Det kan kanskje virke trivielt å hevde at idealer som fremhever forenkling preger moderne organisasjoner, men poenget er imidlertid at ukomplekse prinsipper legger noen føringer for hvilke legitimeringer som regnes som verdige/gyldige i organisasjonskulturen. De ukomplekse idealene regulerer hva som regnes som rettferdig, og de gir mulighet for å rangere mennesker, relasjoner, handlinger og ting. Idealene gir for eksempel preferanser for hvilke type arbeidere som foretrekkes, som kan trives og ha suksess. *Fagidioter* og *nerder* er lite verdige i tråd med ukomplekse idealer. Personer med stort *nettverk*, med mange *jern i ilden*, som *sprer seg for tynt utover* har også liten verdighet. Personer som *begrenser* seg, som ikke har et lidenskapelig forhold til arbeidet og som først og frem bidrar til å gjøre systemet *enkelt* og *bærekraftig* – de har høy verdi.

2.2.3 Forbehold

Hva innebærer det at jeg viser at forenklingens logikk legger føringer i IKT-organisasjoner? Jeg vil ta noen forbehold rundt analysene og betydningen av det teoretiske og metodiske bidraget mitt.

(1) I analysen som beskriver den ukomplekse verdiverdenen er jeg ikke opptatt av om programvareutviklerne har det godt i jobbene sine, eller om de smidige metodologiene har den effekten som loves ut på baksiden av bøkene. Jeg er kun opptatt å vise hvordan en hittil ubeskrevet verdiverden – et system av verdier og idealer hvis høyeste verdi kan måles i enkelhet – eksisterer, og hvordan denne er sentral i et utvalg management-litteratur for IKT-organisasjoner gitt ut etter årtusenskiftet. Det grammatiske oppsettet av den ukomplekse verdiverden er et teoretisk-metodisk bidrag som kan brukes for å vise og beskrive lignende endringer på mikro- og makronivå i samfunnet, både historisk og mellom ulike kulturer.

(2) Det er ikke min intensjon å påvise områder hvor den ukomplekse logikken har "brister", åpenbare mangler eller formallogiske problemer av noe slag. Jeg søker heller å vise hvordan dens gyldighet blir implisert i utvalgene mitt, gjennom at den settes opp mot andre former for logikker og brukes til å understøtte legitimeringer og danne kompromisser. Gjennom Boltanski og Thévenots rammeverk tar jeg hensyn til at aktørene selv har en egen kritisk sans, og jeg studerer hvordan denne blir anvendt for å si noe om hvilke idealer som har gyldighet i faget.

(3) Den ukomplekse verdiverden representerer verken idémessig eller kognitivt sett noe nytt. Idealene som opphøyer det enkle har for eksempel røtter i østlig filosofi. Det første eksempelet på enkelhet som et ideal, er så vidt jeg har kunnet spore Buddhismen, med ideen om frelse gjennom å *fjerne* selve behovet for ting, anerkjennelse og objekter gjennom meditasjon. Prinsippet er også viktig i kristendommen, hvor det gjerne posisjoneres mot former for prestisje og status. Tanker om verdien av det enkle i en sekulær vestlig kontekst går som jeg nevnte i innledningen tilbake til Thoreau (2008), Veblen (2004), Richard Gregg (1936/1983) og økonomen Ernst Friedrich Schumacher (1973/1989). Her fremstilles ideen om det enkle livet som som en reaksjon på det komplekse. Forenkling har også vært et viktig prinsipp i forbindelse med arbeid og organisasjon ved masseproduksjon, som forstått gjennom Taylorisme og Fordisme. Det som er nytt er at jeg organiserer hvordan det ukomplekse tar form som et øverste ekvivalensprinsipp i grammatikken til Boltanski og Thévenot, og jeg viser hvordan dette verdisystemet har vært viktig for å koordinere arbeid og aktiviteter i IKT-organisasjoner og legitimere en pågående *transformasjon* av IKT-faget fra å være en disiplin forbundet med lek, lidenskap og ingeniører til å bli forbundet med stabilitet, styring og pålitelighet.

(4) Jeg hevder ikke at den ukomplekse logikken er universell eller dominerende i kunnskapsorganisasjoner eller blant programvareutviklere i vesten. Jeg forsøker heller ikke å gi en helhetlig beskrivelse av det ideologiske mangfoldet i håndbøkene, eller spenninger som måtte oppstå mellom konkurrende fremstillinger eller perspektiver i bøkene. I de aller fleste moderne programvarebedrifter er det fortsatt høyere produktivitet og lavere utgifter – å skape profitt – som er hovedmotivet. Dette går også klart fram i korpuset. Cohn (2009:230) skriver for eksempel:

> *The primary selection system that should govern organizational evolution is long-term market success. Products that generate profits should displace products that do not; team structures that lead to profitable products should drive out those that do not.*

For at bedriftene skal kunne oppnå profitt, må arbeiderne imidlertid støtte seg på verdier som kan skape tilknytning, engasjement og en profesjonell stolthet – en mening i arbeidet. Som

jeg viser i *analyse #1*, spiller flere idealer en viktig rolle, ikke minst idealer med røtter i hackeretikken (Castells 2001, Castells og Himanen 2002) og den prosjektorienterte verdiverden (Boltanski og Chiapello 2005), som med sine verdier knyttet til nettverk, prosjekt, aktivitet og kommunikasjon. Den ukomplekse logikken er imidlertid viktig i begge utvalgene mine for å begrense gyldigheten av både *hackeretikk* og den *prosjektorienterte* verdiverden i utvalget mitt, og gjøre idealer fra den industrielle verdiverden attraktive igjen. Det er ikke urimelig å tro at forenklingens logikk kan spille en lignende rolle i andre moderne organisasjoner. Avhandlingen gir i så måte et utfyllende bidrag for å lete etter forenklingens logikk i andre kontekster, som tar hensyn til hvordan aktører selv bidrar til å forme en profesjon i utvikling og danne seg et arbeid som de ikke bare skal leve av, men knytte til faglig identitet, mening og stolthet.

2.2.4 Logikk og verdighet

Med logikk mener jeg et sett av idealer og verdier som kan settes sammen til et *koherent* verdisystem, og som vedlikeholdes og utvikles av individer. Når jeg hevder at verdisystemet er koherent mener jeg at folk, relasjoner og ting kan vurderes i forhold til idealer som kan organiseres systematisk ut fra de blir ansett som gyldige og relevante i forhold til et øverste evalueringsprinsipp om forenkling eller klarhet. Dette representerer en egen verdighetstilstand, som åpner for rangeringer av typen *X er enklere enn Y*, *A er klarere enn B*. Verdien av dette prinsippet kan settes opp mot verdien av prinsippene om inspirasjon og effektivitet, og representerer et eget måleprinsipp. Det enkle er ikke alltid det mest effektive eller inspirerende. Vurderingene uttrykkes gjennom et vokabular, som også er ladet med verdi og mening. For eksempel er verbene *å redusere* og *å fokusere* viktige. For at en logikk skal kunne kalles en verdiverden, må den tilfredsstille seks kriterier – det mest sentrale er at det er mulig å oppnå for alle mennesker, gitt at de investerer riktig.[12]

Verdien av *å forenkle* vil i en *gitt kultur* kunne bli gyldiggjort som viktigere enn verdien av høy effektivitet i produksjon, eller viktigere enn å forfølge enhver inspirasjon eller ethvert innfall. Denne typen logikker har imidlertid ikke allmenn gyldighet, tvert i mot er de omstridte og deres mening og relevans i ulike situasjoner er gjenstand for forhandlinger. Ved å flytte oppmerksomhet fra å studere dominerende logikker eller legitimeringsstrategier til å studere det dynamiske mangfoldet av verdiverdener og kompromisser, åpnes muligheten for å analysere kultur i endring (Blokker 2011; Jagd 2011). Strukturelle analyser blir gjerne kritisert for å ha funksjonalistiske eller mekaniske forståelser av menneskelig handling. Ved å knytte logikkene til aktørenes forståelse av verdighet, åpnes det for forståelser hvor menneskene er høyst levende og

12 En nærmere presentasjon av disse kriteriene finnes på side 41.

menneskelige.

2.3 Ukomplekse idealer i andre faglige kontekster

Det jeg kaller ukompleks logikk – som i sin reneste form hevder at "så enkelt som mulig" alltid vil være til det beste for alle – finnes i mange former i ulike kontekster og fagområder. Reilly et. al. fant for eksempel at for leger ga det å rangere hjerterisiko etter ett enkelt flytskjema sikrere vurderinger enn individuelle profesjonelle vurderinger (Reilly, Evans, Schaider, og Wang 2002; Reilly, Evans, Schaider, Das, mfl. 2002). Ophir et al (2009) fant at å skifte oppmerksomheten raskt mellom ulike medier går utover konsentrasjon, oppmerksomhet og ytelse. Mennesket har en kognitiv kapasitetsbegrensning. Å skjerme seg for inntrykk i en produksjonssituasjon vil i så fall kunne gi høyere produktivitet. Dette idealet finnes også i andre mer generelle metoder for å organisere personlig produktivitet. *Getting Things Done* er en organiseringsmetode basert på at kunnskapsarbeidere må flytte oppgaver "ut av hodet" ved å skrive dem ned. På denne måten er sinnet fritt til å konsentrere seg om det det skal gjøre i øyeblikket (Allen 2002). Ukompleks logikk benyttes også i håndbøker for å lage gode powerpint-presentasjoner, hvor det å samle seg om enkle budskap, og bruke få ord og visuelle elementer blir fremhevet som en styrke (se f.eks. Reynolds 2007; Duarte 2008). Evidensbasert management (Armstrong 2009; Cokins 2009) kan sees i lyset av ukompleks logikk – enkle målinger og tall er noe de fleste kan forstå og samle seg om, og blir dermed et godt verktøy for styring av organisasjoner. På de vitenskapelige feltene matematikk og geometri er også forenkling og reduksjon et svært viktige prinsipp, og det samme gjelder til en viss grad informatikk.

2.4 Struktur for resten av avhandlingen

I kapittel 3. introduserer jeg studiefeltet, og gir en begrunnelse hvorfor dette er et viktig felt å studere. Jeg presenterer litt IKT-historie, som både er nødvendig for å forklare hvordan jeg strukturerte utvalget mitt, og som forklarer betydningen noen av informantene tillegger fri programvare. I kapittel 4. presenterer jeg rammeverket til Boltanski og Thévenot. Jeg presenterer bakgrunn, forskningsprogram og sentrale konsepter og knytter innholdet delvis til egne eksempler. I kapittel 5. presenterer jeg det analytiske verktøyet jeg bruker for å belyse reproduksjonen av profesjonell identitet. Dette er i hovedsak konstruert rundt Boltanski og Thévenots forestilling om tester, men er utvidet til å være sensitivt for det jeg kaller personlig og kulturell identitetsformasjon i tråd med profesjonsteori. I kapittel 6. presenterer jeg utvalg og metode. Kapittel 7. består av den første grenseanalysen min. Her beskriver jeg fire fagverdigheter,

og undersøker hvordan disse blir reprodusert gjennom legitimeringer og praksis i organisasjonene. I kapittel 8. går jeg videre og utdyper bakgrunnen for noen av de mest uventede funnene i den første analysen. Jeg presenterer det historiske grunnlaget som gjorde at kompromisset som skulle bli *den ukomplekse verdiverden* kunne vokse fram, deretter presenterer jeg innholdet i håndbøkene jeg har studert, skisserer den "nye" verdenens moralfilosofiske røtter, før jeg presenterer denne i sin helhet. Til sist i kapitlet relaterer jeg den ukomplekse verdiverdenen til noen av de andre verdiverdenene hos Boltanski og Thévenot, spesielt den industrielle og den kollektive. I kapittel 9 sammenfatter jeg analysene, og viser hvordan den prosjektorenterte verdiverden blir begrenset, og at personlige motivasjoner og det mange forstår som profesjonsmoral krever legitimering og gyldiggjøring for å kunne danne en profesjonell identitet. Avslutningsvis antyder jeg mulige generaliseringer, og peker mot andre felt det metodisk-teoretiske bidraget mitt kan bidra til å belyse.

3. Introduksjon av studiefeltet

3.1 Hva er programvare?

Informasjonsteknologier (IKT) og programvare har spilt en sentral rolle for vekst og innovasjon i moderne økonomier de siste 20 årene. I løpet av disse årene har det foregått store teknologiske endringer i samfunnet, som blant annet har gjort det lettere å kommunisere, dele og lagre informasjon. Mange tenker på PC-er når de hører ordet programvare. I 1985 hadde 9 % av norske husstander tilgang på PC, i 2009 hadde dette tallet steget til 92 % (SSB 2010a). Produktene av *programmering* finnes imidlertid i dag overalt i samfunnet. I mediene, hos kommune og stat, i Nordsjøen og i stadig større grad også i hjemmet, for eksempel i komfyren. Former for programvare finnes i alle digitale apparater og maskiner, og ulike typer programvare utgjør tilsammen entiteten som alle kjenner, men ingen har full oversikt over: Internett.

3.2 Hva gjør en programvareutvikler?

Programvareutvikling er derfor en egen yrkesretning. Programvareutviklere *lager* og *vedlikeholder* programvaren som gjør at vi kan bruke fjernkontrollen, kjøre biler produsert etter 1999, sende e-post og bruke mobiltelefonen til mer enn å bare ringe med. Programvareutviklere har også programmert operativsystemene, svære nettverkstjenester som f.eks. Facebook, og alle programmene som finnes på PC-er rundt omkring. Programvareutviklere arbeider både i konsulentfirmaer, i egne avdelinger i andre bransjer (f.eks. olje/finans/medier) og i egne dedikerte programvarefirmaer. Mens det blir stadig vanligere at programmerere har 5-årig informatikk- eller ingeniørutdannelse, hersker det fremdeles uenighet hvorvidt fagutøvelsen kan kalles en profesjon. Jeg er heller ikke opptatt av dette spørsmålet.

Det er vanskelig å dele inn programvareutviklernes arbeid i spesifikke sektorer for å tegne en statistisk skisse. OECD og SSB deler inn IKT-sektoren i to hovedgrupper, kalt IKT-industri og IKT-service. IKT-service er igjen delt inn i tre bransjer, IKT-varehandel, IKT-telekommunikasjon og IKT-konsulentvirksomhet. I sistnevnte gruppe er *system og programvare* en av sju undergrupper, og det er antakeligvis i denne gruppen programvareutviklere burde vært samlet. Men inndelingen

gir ikke et riktig bilde, fordi det er umulig å avgrense programvareutvikling mot andre sektorer og grupper. Bedrifter i olje og gass-sektoren har ofte egne utviklere. Det samme gjelder helsesektoren. Disse blir dermed ikke med i statistikken over IKT-sektoren. De blir i stedet registrert under andre næringer. Telekom og IKT-industrien har gjerne også egne avdelinger med utviklere, samtidig som de har produksjonsavdelinger og salgsavdelinger. Dette gjør at det er vanskelig å anslå hvor mange programvareutviklere det egentlig er som jobber i Norge.

Mange av IKT-arbeiderne i Norge jobber dessuten hovedsakelig med å tilpasse programvare for kunder. Dermed er det ikke først og fremst det å utvikle programvareløsninger som er hovedproduktet, men heller relativt enkel konfigurasjon og mindre tilpasninger. Antallet som tok en mastergrad i teknologi/sivilingeniør i skoleåret 08/09 lå på 998 – 760 menn og 238 kvinner (SSB 2010b). Men igjen: Ikke alle disse er programvareutviklere. Internasjonalt har Norge en relativt stor IKT-industri som består av små og middels store firmaer og foretak (Jordfald og Olberg 2002). Likestilling har kommet langt i forhold til mange andre land, og Norge skiller seg ut positivt i forhold til ansatterettigheter i forbindelse med foreldreskap og fagorganisering. Likevel er kvinneandelen i bransjen relativt lav. Det foregår flere kampanjer for å øke rekrutteringen til faget (Lagesen 2007; Lagesen og Sørensen 2009). For ti år siden var det bare 25 % kvinner i IKT-konsulent, men det var også en tendens til at menn i bransjen hadde høyere utdanning enn kvinner.

3.3 Hvorfor er det interessant å studere programvarefaget?

1. Det er interessant å studere IKT- og programvare fordi fagfeltet er viktig både økonomisk og kulturelt, samtidig som det fremdeles er i utvikling. Dermed kan fagfeltet inneholde flere konflikter og spenninger. Spenningene mellom ulike faglige forståelser utgjør et interessant studieobjekt, og dette er utgangspunkt i analyse #1.
2. Feltet er videre interessant fordi programvareutvikling er et *arbeid* som utføres i *organisasjoner*. Begge er sentrale studieobjekt for sosiologien helt fra Marx og Webers dager og fram til i dag.
3. Organisasjonene i utvalget mitt er også interessante fordi de befinner seg i kjernen av det som mange kaller kunnskapsarbeid. IKT-arbeidere er tidligere omtalt som en gruppe som det kan være verdt å studere fordi de tydeliggjør endringer i moderne kunnskapsarbeid (Scarbrough 1999; Glover, Currie, and Ackroyd 2000).
4. Til slutt er feltet interessant å studere fordi det ligger i kjernen for mange sosiologiske samtidsdiagnostiske prosjekt, for eksempel Bell (1976), Castells (2009) og Himanen

(2001).[13] Alle disse tre forfatterne argumenterer for at moderniteten og produksjonsmodellene er preget av at nye idealer knyttet til individualisering bryter med eldre idealer assosiert med tidligere former for arbeid og produksjon. Gjennom denne avhandlingen forklarer jeg utviklingen i dette feltet på en alternativ måte. Denne avhandlingen tar imidlertid ikke mål av seg å være en samfunnsdiagnose, analysene begrenser seg til IKT-faget.

3.4 En kort historisering av fri- og produsenteid programvare

På sekstitallet var datamaskinene store, sjeldne og svært dyre. De datamaskinene som fantes tilhørte gjerne store organisasjoner og universiteter. Den gang var det vanlig å *dele* både kildekode[14] og programvare blant studenter og brukere av datamaskinene. Når folk fra andre universitet skrev og spurte om de kunne få kildekode de kunne videreutvikle selv, var det vanlig å gi dem denne uten vederlag (Stallman 1999). Selv om perioden tidlig i programvareutviklerhistorien har elementer som minner om dagens fri programvarekultur, var det likevel to åpenbare forskjeller: For det første var ikke maskinvaren standardisert, sånn at tilgjengelig programvare gjerne bare fungerte på en gitt (og ofte sjelden) type arkitektur, og for det andre manglet Internett. Distribusjon av programvare var dermed dyrt og vanskelig. Utover syttiårene-årene oppfordret mange dataselskaper til deling av kildekode som ledd i kompetanseutvikling og rekruttering rundt deres eget system. Dette var forløperen til *fri* programvare. På åttitallet ble maskinvaren i større grad standardisert, og markedet for programvare vokste. Den kommersielle bransjen begynte å selge programvaren som binær kode (uten kildekode) – i stedet ble det lagt ved strenge lisenser som forhindret modifisering. Uten kildekode kunne ikke konkurrenter, utviklere eller andre interesserte lenger studere eller endre programmene (Von Hippel og von Krogh 2003).[15] Utover 80-tallet oppsto det en motkultur i de universitetsbaserte programmeringsmiljøene, med Richard Stallmann som en av frontfigurene. Muligheten til å studere kildekode – både som amatør og profesjonell – hadde ifølge Stallmann vært sentral for innovasjon i programvareutviklingen fram til 80-tallet, og denne verdien måtte

13 Med samtidsdiagnose forstår jeg analyser som er begrepslig formidlet, empirisk testbare og som gir en helhetlig modell av samtiden i tråd med Aakvaag (2011:263).

14 De aller fleste dataprogram er opprinnelig skrevet som ren tekst, i en eller flere syntakser avhengig av programmeringsspråk. Denne teksten skrives, leses og forstås av mennesker, og kalles kildekode. En god analogi for kildekode, er kakeoppskrift. Kildekoden er oppskriften, programmet er en ferdig stekt kake.

15 Dette hindret dog ikke at kommersiell programvare uten tilgjengelig kildekode ble *reverse engineered* – at man skrev ny kode som hadde de samme funksjonene som annen programvare. Linux ble opprinnelig skrevet som en reverse engineered kopi av Unix.

beskyttes.

Stallmann tok initiativet til Free Software Foundation og skapte GPL – en *fri* lisens som fastslo at kildekoden alltid *måtte* distribueres sammen med programvaren (Stallman 1999; Weber 2004). Industrien vegret seg for å bruke GPL-lisenser til sine produkter, og to utviklere – Perens og Raymond – bestemte seg for å lage en lisens som var litt mindre streng enn Stallmanns. Denne lisensen kalte de *open source.* Både free software og open source gir full rett til å endre og distribuere egne modifikasjoner av kildekoden basert på originalen. Rettighetene er altså ganske radikale hvis man sammenligner med rettighetene som tredjeparter får gjennom tradisjonell opphavsrett (Weber 2004). Forskjellen mellom free software og open source ligger i førstnevntes krav til å måtte distribuere derivater av programvaren under *samme vilkår* som originalen. Dette begrenser mulighetene for å benytte kodebiblioteker og kode som er basert på andre lisenser enn free software, siden disse ofte vil være inkompatible med den originale lisensteksten. Moderne operativsystem må bruke patentbeskyttet kode for å fungere med avspilling av musikk, film og være kompatible med for eksempel nettbankløsninger. Mens dette ble vanskelig å tilby med free software, mangler open source-lisenser denne begrensningen. Open source er altså en pragmatisk løsning med noe større fleksibilitet enn free software (DiBona, Ockman, og Stone 1999). Selv om open source-lisenser gir noe mindre intellektuell beskyttelse, blir de ofte valgt foran free software-lisenser på grunn av at lisens-fleksibiliteten vil gjøre det enklere for andre å ta i bruk programmene og inkludere teknologien i sine prosjekter. Dette skillet er imidlertid ikke viktig for argumentene i avhandlingen, og jeg bruker i resten av avhandlingen termen *fri programvare*, som da referer til både free software og open source.

Selv om det finnes mange ulike lisenser for fri programvare, inneholder de aller fleste følgende aspekter: (1) Enhver kan endre kildekoden, (2) enhver kan redistribuere programvaren i både modifisert og umodifisert form, (3) og at opphavspersonen og kode-forfatterne ikke har noe ansvar hvis du velger å bruke koden. Forskjellene i lisensene koker ned til kompabilitet med produsenteide eller andre frie lisenser, behovet for kreditering av kodeforfattere eller opphavsperson, beskyttelse av varemerker eller firmanavn, og til slutt beskyttelse av "artistisk integritet". I praksis vil Free Software-programvare for det meste lisensieres under en GNU General Public License (GPL), og Open Source-programvare vil hovedsakelig lisensieres under en MIT License. GPL er i dag den mest utbredte lisensen for fri programvare-prosjekter (Fogel 2005).

Fri programvare kan kontrasteres til produsenteid programvare. Kildekoden til produsenteid programvare blir ikke delt mellom andre enn de produsenten velger, og utviklerne som jobber med den. For så og si alle store produsenteide programvareprosjekter vil den juridiske opphavspersonen være en organisasjon eller en bedrift som lønner de ansatte gjennom enten salg

av et programvareprodukt, ved å selv benytte produktet som utvikles, eller ved å bruke produktet til promotering av *andre* produkter. Den juridiske opphavspersonen til et fri programvare-prosjekt er ofte en organisasjon eller et privat initiativ, som kan enten lønne eller ikke lønne programvareutviklerne. Noen ganger vil enkelte utviklere være fast ansatt, mens andre er hyret inn på kortere eller lengre kontrakter. I de senere årene har vi i stadig større grad sett at fri programvare-utviklere er ansatte i store, profesjonelle organisasjoner (Lerner og Schankerman 2010).

3.5 Den privat-kollektive innovasjonsmodellen

Innovasjoner og gode idéer har vært forutsetningen for sosial progresjon gjennom historien. I vesten er det vanlig å forstå utvikling ut fra to ulike innovasjonsmodeller. Den *private investeringsmodellen* forutsetter at investoren får avkastning gjennom salg av varer og tjenester gjennom effektiv håndheving av intellektuell eiendom (Demsetz 1967; May 2000). Den *kollektive handlingsmodellen* forutsetter at innovatører samarbeider for å utvikle et kollektivt gode, uansett om markedet belønner det eller ikke. Kollektive goder kjennetegnes av at de er verken er ekskluderbare eller rivaliserende. Dette betyr at alle kan gjøre nytte av godene samtidig. Under dette regimet tjener ikke innovatøren mer enn andre brukere, noe som åpner opp for *gratispassasjerer* (Olson 1971). Et eksempel på et kollektivt gode er for eksempel offentlig finansiert grunnforskning – funnene blir åpent publisert og alle kan gjøre nytte av dem (Sandler 1992). Innen akademia, i hvert fall tidligere, sikres innovasjon gjennom statlig finansiering og muligheten for at institutt/enkeltpersoner kan få anerkjennelse (Merton 1979).

Både produsenteid og fri programvare har vært sentrale i utviklingen av IKT-samfunnet de siste 30 årene. Produsenteid programvare produseres med grunnlag i en privat investeringsmodell – men fri programvare blir *ikke* produsert med grunnlag i en kollektiv handlingsmodell. Individer og selskaper investerer derimot sine private ressurser i fri programvare. Dette har Von Hippel og Von Krogh (2003) kalt en *privat-kollektiv innovasjonsmodell*. Både enkeltindivider og selskaper investerer i utviklingen av fri programvare, gjennom å arbeide, betale arbeidere eller konsulenter for å utvikle programvare. Private aktører investerer altså i utviklingen av kollektive goder (Stuermer, Spaeth, og Von Krogh 2009; Spaeth, Stuermer, og Von Krogh 2010).

Hvorfor investere penger i noe man kan slippe å betale for? Det kan virke som et paradoks at mange investerer i fri programvare og at sektoren vokser. Antakelig er de private fordelene ved å investere i fri programvare større enn fordelene ved å være gratispassasjer. Fogel lister opp fem

grunner til at organisasjoner velger å investere i fri programvare (2005):

- *For å dele produksjonsutgifter med andre* – ulike organisasjoner kan spare penger ved å investere i ett sentralt prosjekt i stedet for å utvikle hver sin løsning som gjør akkurat det samme.
- *For å forbedre servicen* – hvis en bedrift selger tjenester som baserer seg på fri programvare, vil man sikre seg at programvaren forblir oppdatert og utvidet.
- *For å støtte salg av maskinvare* – verdien av maskinvare (f.eks. pc-er og mobiltelefoner) henger direkte sammen med programvaren, og hva den kan gjøre.
- *For å underminere konkurrenter* – noen firma støtter fri programvare-prosjekter for å ta markedsandeler fra en konkurrent.
- *Markedsføring* – å ha firmaet assosiert med populær programvare er rett og slett god reklame.

I 2006 donerte IBM kode estimert til en verdi av 40 millioner dollar, da ved å frigi kildekoden til programvareutviklingsverktøyet Eclipse (Fitzgerald 2006). Ved å donere kildekode øker IBM sjansen for at framtidige fri programvare-aktører vil benytte nettopp deres plattform og produkter i framtiden.

Fri programvare er selvfølgelig ingen gylden oppskrift for suksess, og historisk sett har svært mange fri programvareprosjekter strandet. Urealistiske kvalifiseringskrav, vage spesifikasjoner, svak ressursstyring, utilfredsstillende designfaser og sviktende finansiering er noen av grunnene til dette. Produsenteide programvareprosjekter vil i mange tilfeller potensielt kunne gjennomføres raskere, være lettere å administrere, og koste mindre enn et fri programvare-prosjekt. I tillegg er potensialet for kortsiktig (og muligens også langsiktig) profitt større i produsenteide prosjekter (Fogel 2005).

Fri programvare har også fordeler. Man slipper å bruke tid på å utvikle en funksjon eller kode som allerede finnes. Fri programvareproduksjon er en kumulativ prosess, som bygger videre på tidligere arbeid. Fri programvare-kode er i praksis etterprøvbar for alle – kildekoden er tilgjengelig for testing, kontroll og eksperimentering. På sikt vil dermed produkter bygd på fri programvare potensielt kunne være mer robust, siden de er bygget opp over lengre tid, og flere øyne har vært innom og kontrollert koden (DiBona mfl. 1999). Et godt fri programvare-prosjekt vil heller ikke lide hvis opphavsperson eller prosjektledelsen skulle finne det for godt å trekke seg ut – kildekoden er tilgjengelig, og andre kan fortsette der utviklingen stoppet. Produksjonen av fri

programvare vil også i sin natur være desentralisert og globalisert, noe som gjør at utviklingen ikke stopper på grunn av uforutsette strukturelle endringer i f.eks. en nasjons eller et firmas økonomi (Weber 2004). Oversettelse til flere språk vil også gjøres enklere.

Mens fri programvare er preget av en kollektiv logikk med deling av utgifter til fordel for alles beste, har produsenteid programvare røtter i en "tradisjonell" kapitalistisk eiendomsmodell. Spenningsfeltet mellom fri og produsenteid programvare kan sees på som drevet av to helt ulike måter å forstå, utnytte og legitimere intellektuell eiendom på. Den kollektive innovasjonsmodellen har gitt mange robuste produkter, for eksempel serverprogramvaren Apache (som over halvparten av verdens webservere fremdeles benytter) og nettleseren Firefox, OpenOffice og Ubuntu. Men utviklingen og utbredelsen av produktene er kumulativ og langsom. Produsenteid programvares utvikling og utbredelse er i større grad gjenstand for store investeringer, markedsføringskampanjer og trender.

4. Det pragmatiske rammeverket

I de siste årene har tilnærminger som ser organisasjoner som karakterisert av et mangfold av logikker og rasjonaliteter bidratt til å øke vår forståelse av produksjonen av orden og endring i organisasjoner (Jagd 2011). Ulike forestillinger har blitt brukt for å beskrive den samtidige eksistensen av konkurrerende logikker eller rasjonaliter. Disse inkluderer institusjonell pluralisme (Kraatz og Block 2008), institusjonelle logikker (Reay and Hinings 2009; Thornton 2004; Thornton and Ocasio 2008), konkurrerende rasjonaliteter (Cloutier and Langley 2007) og pluralistiske kontekster (Denis, Langley, and Rouleau 2007; Jarzabkowski and Fenton 2006). Boltanski og Thévenot (Boltanski og Thévenot 1999, 2006; Thévenot 2001a, 2007) bidrar til dette feltet gjennom sitt teoretiske rammeverk om et mangfold av verdiverdener – eller logikker (Jagd 2011:344). Det som skiller Boltanski og Thévenot fra de andre tilnærmingene er perspektivet, eller handlingsteorien, som understreker viktigheten av *verdighet*, *kritikk* og *legitimering* gjennom begrepet *virkelighetstester* for å forstå sosial stabilitet og endring. I denne avhandlingen bruker jeg Boltanski og Thévenots rammeverk for å undersøke hvordan ulike *meninger med arbeidet*, det vi kan forstå som ulike *profesjonelle verdigheter*, konkurrerer i IKT-organisasjoner, og bidrar til å utforme og reprodusere en profesjonell eller yrkesfaglig identitet. Denne identiteten kan gjennom dette rammeverket forstås som aktørprodusert, men likevel forankret i både fagmiljøet og det moderne samfunnet, gjennom at den i stor grad deler språk og verdier, og er utformet i tråd med hvilke logikker og institusjoner som finnes, og hvilke oppgaver som regnes som legitime og viktige i samfunnet.

4.1 Introduksjon

Gjennom det meste av avhandlingen følger jeg det teoretiske rammeverket hos Boltanski og Thévenot (2006) og Boltanski og Chiapello (2005). Denne teoretiske retningen har blitt kjent som fransk pragmatisk sosiologi (Benatouil 1999; Nachi 2006). "Pragmatisk" referer ikke til amerikansk pragmatisme, selv om en indirekte inspirasjon kan spores til teoretiske perspektiv som symbolsk interaksjonisme og etnometodologi. Snarere refererer begrepet til lingvistisk pragmatisme, som understreker aktørenes bruk av grammatiske ressurser for å legitimere handling i ulike

situasjoner (Jagd 2011:345). Boltanski og Thévenots (2006) rammeverk kan forklare hvordan *forståelsen* og *betydningen* av arbeid og økonomi transformeres på ulike nivå gjennom *kritikker* og *kompromisser* med referanse til allmenne oppfatninger av hva som er *godt* eller *verdig*. Boltanski og Thévenots rammeverk forutsetter at aktørene har *kritisk kompetanse*, det vil si at de har mulighet for å forstå og påvirke profesjonell og faglig utvikling gjennom *engasjement*. Rammeverket anerkjenner aktørers evne til å formulere kritikker som blir drivkrefter for sosial endring. Et sentralt premiss for rammeverket er at aktørene som engasjerer seg i en diskusjon eller en argumentasjon er opptatt av å definere det *gode*. For at et argument skal kunne bli oppfattet som godt og få oppslutning i en offentlighet, må aktørene utforme det på en måte som gjør at det appellerer til andres oppfatning om det gode.

Jeg bruker rammeverket til å gi en *aktør-orientert* forklaring på fremveksten og endringene av IKT-organisasjoner og programvarefaget. Selv om rammeverket tar mål av seg å beskrive sosial endring som drevet fram av aktører gjennom legitimering, tar det hensyn til en *strukturell* komponent: for å vinne fram må kritikker formuleres i tråd med allmenne idealer og virkelighetsoppfatninger – som allerede er godt etablert i samfunnet. Sosial endring basert på kritikker og kompromiss er dermed langsom og krever bred mobilisering.

Når aktører deltar i diskusjoner eller debatter realiserer de sin *kritiske kompetanse*. Denne kritiske kompetansen er noe aktører benytter både på det partikulære nivået (i mindre sammenhenger, f.eks. privat eller blant kolleger) og på det generelle nivået (offentlig, i avisene og ved offentlige høringer m.m.). Verdiverdenene kan benyttes for å analysere og sammenligne både mikro- og makrososiologiske dimensjoner (Blokker 2011:257; Boltanski og Chiapello 2005:32). I dette kapitlet vil jeg presentere rammeverket ved hjelp av ulike eksempler. Det teoretiske rammeverket er utviklet som en generell kultur- og handlingsteori, så det er ikke alltid de beste eksemplene er fra organisasjonslivet. I kapittel 5. beskriver jeg hvordan jeg bruker dette rammeverket for å undersøke reproduksjon av profesjonell identitet.

4.2 Utviklingen av en moralsosiologi

Hva styrer menneskelig handling? I hvor stor grad er våre handlinger styrt av sosialisering og strukturene som omgir oss, og i hvor stor grad er vi autonome? Dette er et av de sentrale ontologiske spørsmålene innen sosiologien. Durkheim (1965, 1992) og Mauss (1970) var de første som interesserte seg for å studere kognitive kategoriseringer eller symbolske grensedragninger for å forsøke å svare på dette spørsmålet, og forståelsene disse skapte har spilt en viktig rolle for både Pierre Bourdieus sosiologi (f.eks. 1984), Mary Douglas antropologiske studier (f.eks. 1966) og

Abbots (1988) teorier om utviklingen av profesjoner. Samtidig har de handlingsteoretiske implikasjonene av dette vært begrenset gjennom at både kollektivene og de kognitive kategoriene har blitt forstått som reprodusert ved at aktørene har *ønsket* eller *arvet* medlemskap i ulike sosiale kollektiv (Thévenot 2007). Dette har betydning for hvorfor og hvordan Boltanski og Thévenot utviklet sitt rammeverk. Et sentralt spørsmål hos Boltanski og Thévenot var å finne ut hvordan kognitive kategorier reproduseres på tvers av klasser og disposisjoner. Svaret ble utviklingen av en moralsosiologi som i stor grad var posisjonert mot Bourdieus handlingsteori. Boltanski og Thévenots tilnærming står imidlertid også i skarp kontrast til Rational Choice, som forutsetter at handlinger først og fremst er rasjonelle og kalkulerbare. Boltanski og Thévenots handlingsteori viser at evalueringsrepertoarene i et samfunn har en høy grad av historisk kontinuitet. Ikke minst markerer rammeverket et punkt hvor rettferdiggjøring og ekvivalens ikke lenger har relevans.

Bourdieu tok utgangspunkt i at aktører først og fremst kunne sees som medlemmer i sosiale grupper, som reproduserte visse disposisjoner gjennom *tillærte* tanke-, adferds- eller smaksmønstre (habitus) eller *forventninger* til en kultur (klasse-etos). Aktørenes handlingsbetingelser og forståelser blir følgelig betinget av at aktørene ønsker fortsatt medlemsskap i disse gruppene. I *Distinction* (1984) viser Bourdieu at logikken bak legitimering av klasseforskjeller kan utvides til å gjelde smak og livsstil, og at klasseprivilegier blir reprodusert gjennom symbolsk klassifisering. Dominante grupper, hevder Bourdieu, klarer å legitimere sin egen kultur og sitt levesett som moralsk høyverdig. Dette gjøres gjennom at det dras symbolske grenser mellom høyverdige og lavverdige kulturelle trekk, gjennom beskrivelser som for eksempel distingvert/vulgær, estetisk/praktisk og ren/uren. Gjennom begrepet *habitus* forklarer Bourdieu hvordan kulturelle praksiser medfører uunngåelige og ubevisste klassifiseringer som også påvirker samfunnets inndeling i sosiale klasser. Ideen om klassifiseringskamper (med den hensikt å vinne fram med sitt eller sin gruppes verdensbilde) på ulike *felt* er tilsvarende viktig for Bourdieu. Et felt er forstått som et nettverk av sosiale relasjoner, hvor for eksempel penger eller prestisje er en knapp (og dermed konkurranseutsatt) ressurs. Gjennom å fremstille sitt syn som eksklusivt og å være intolerant ovenfor nye kulturelle impulser, kan en dominant klasse beholde eller forsvare makt på et gitt felt. Bourdieus begrep om kulturell kapital er viktig i hans handlingsteori for å forklare sammenhengen mellom kultur og ulikhet. I hans handlingsteori ble de øvre klassene sett på som forvaltere av høykulturell kapital. Dette bidro til at de øvre klassenes kultur ble ansett som koherent og eksklusiv. Bourdieu postulerte her at det er en sammenheng mellom en aktørs posisjon i det sosiale rom og de kognitive kategoriene en person bruker for å forstå og uttrykke seg i verden. Bourdieus kritiske sosiologi har dermed, i hvert fall i *The distinction*, en strukturell forståelse av reproduksjonen av klasse og kultur (Lamont 1992; Vallas

2001; Boltanski og Thévenot 2006).

Boltanski og Thévenot var først studenter av Bourdieu, men utviklet etter hvert en egen handlingsteori, hvor handling blir forklart ut fra praktiske situasjoner – ikke klasse, gruppetilhørighet eller konsekvenser. Aktørene i pragmatisk sosiologi er aktive, bevisste og refleksive, og den sosiale sfæren blir ikke forstått ut fra dominans og hegemoni, men som et større offentlig rom hvor det er plass til et mangfold av konflikter, kritikker, lokale avtaler og kompromisser (Jagd 2011:345–6). Den pragmatiske aktøren kan være ærlig eller strategisk – han/hun kan referere til ulike typer logikker fordi de tror på det, eller fordi det gir dem fordeler. Hvis et argument skal vinne fram, må det imidlertid formes slik at andre aktører oppfatter det som gyldig og med referanse til virkeligheten.

Det sosiologiske rammeverket til Boltanski og Thévenot ble utviklet på åttitallet i skjæringspunktet mellom lingvistisk pragmatisme og sosial rettferdighetsteori. *De La Justification* kom i første utgave i 1987, og møtte mye kritikk. 1991-utgaven, oversatt til engelsk i 2006, inneholdt svaret på mye av denne kritikken. Akkurat som Elster (1993) fokuserer Boltanski og Thévenot på at rettferdighet kan ha et mangfold av former, og disse er organisert etter "worth" eller verdi – men i motsetning til Elster begrenses ikke disse formene til utelukkende å gjelde situasjoner ved fordeling av goder og byrder i et samfunn. Snarere kan alle henvise til hva som er verdifullt i ulike situasjoner, for å løse en konflikt, fremme en kritikk og oppnå et kompromiss. Det handlingsteoretiske grepet er at individer besitter en kritisk kompetanse til å vurdere hva som er riktig og galt i ulike situasjoner. Når mennesker betrakter eller er del av en situasjon, blir disse vurderingene konstruert og delt med andre gjennom språk og handling. Her er den strukturelle reproduksjonen forlatt til fordel for en handlingsteori hvor aktørene er utstyrt med evne til å vurdere, forhandle og danne ulike kategoriseringer og forståelser når de skal begrunne sine argumenter, handlinger eller valg. Disse kategoriseringene er imidlertid ikke løsrevet fra samfunnet eller kulturen aktørene er del av. Tvert i mot er kategoriseringene kulturelt betingede historiske konstruksjoner som må ha gyldighet i en større kontekst hvis aktøren skal regne med å få aksept for en legitimering. Gjennom historien vil disse konstruksjonene endre seg, ettersom aktørene knytter sine argumenter til nye praksiser og samfunnsforhold. Noen kategoriseringer vil tape terreng, andre vokser fram. Derfor er det sosiologisk interessant å se på legitimering – måtene aktører bruker språket for å begrunne sitt engasjement med henvisning til ideer og moralske argumenter.

Boltanski og Thévenots rammeverk representerer en klassifiseringsteori som henter mye inspirasjon fra kognitiv grenseanalyse. Analyseobjektet er derimot ikke bare symbolsk språk, men også verdier og idealer satt i relasjon til materielle objekter og praksis, hvor kategorier også kan

være naturaliserte. Rammeverket utfyller andre klassifiseringsteorier (Foucault 1994; Desrosières 1998; Hacking 2000:125–162) gjennom at det tar hensyn til det Star og Griesemer (1989) kalte "boundary object" (grenseobjekt), som beskriver hvordan konsepter og forklaringer brukes på ulike måter i ulike kulturer. Grenseobjekt er akkurat så plastiske at de lar aktører bruke dem i lokale situasjoner, men også så robuste at de kan overføres og brukes i flere situasjoner. De kan være både abstrakte og konkrete, men representerer en "logikk" som kan overføres og oversettes, og som dermed spiller en rolle i å opprettholde mening og koherens på tvers av ulike sosiale verdener (Star og Griesemer 1989:393). Boltanski og Thévenot utvider dermed konseptet med grenser til ikke bare å brukes i forbindelse med eksklusjon og separasjon, men også for å kommunisere, dele meninger, bygge broer og inkludere (Lamont and Molnár 2002:180-1).

I norsk sosiologi ble Boltanski og Thévenot for alvor kjent gjennom artikkelen "Den 'legitime kulturens' moralske forankring" av Ove Skarpenes (2007), knyttet til forskningsprogrammet *Kunnskap og kultur i den øvre middelklassen i Norge* ledet av Rune Sakslind ved UiB. Artikkelen beskriver blant annet hvordan den norske middelklassen vegrer seg for å bruke kulturelle grensedragninger uten først å la disse bestå en moralsk test (Boltanski og Thévenot 2006:133). Skarpenes konkluderer med at den norske middelklassen i stor grad, og til forskjell fra en del andre europeiske land, uttrykker motvilje mot å rangere kulturuttrykk hierarkisk. Artikkelen ble møtt med svært ulike reaksjoner. På den ene siden ble Skarpenes tildelt Universitetsforlagets tidsskriftspris, på den andre siden fikk artikkelen mye motbør fra sosiologer i både akademiske og tradisjonelle medier. Dette vitner om at de teoretiske og metodiske perspektivene som ligger til grunn for Boltanski og Thévenot oppfattes som kontroversielle i norsk sosiologi (Andersen og Mangset 2012). Min framgangsmåte ligner imidlertid lite på framgangsmåten hos Skarpenes. Han bruker forestillingen "test" for å vise at informantene begrenser sine kulturelle grensetrekkinger i intervjusituasjonen ved ikke å la disse passere en moralsk prøve. Jeg følger i stedet Boltanski og Chiapello (2005), som jeg først og fremst bruker for å beskrive endringer i forhold til profesjon/fag. Jeg bruker forestillingen om *test* for å vise hvordan den profesjonelle identitetens formasjon og transformasjon er både drevet fram av aktører, og forankret i samfunnet og de lokale organisasjonene aktørene er en del av (se side 45 og 61).

4.3 Handlingssfærer eller handlingsregimer

Boltanski og Thévenots rammeverk er hovedsakelig utviklet for å analysere rettferdiggjøring, det vil si situasjoner hvor to eller flere aktører forsøker å løse en uenighet, disputt eller konflikt gjennom argumentasjon. Sosial koordinering i organisasjoner foregår selvsagt også i

handlingsregimer hvor rettferdiggjøring er mindre viktig. Jeg skiller mellom fire handlingssfærer eller handlingsregimer i organisasjoner. Disse er kategorisert etter konfliktnivå og refleksivitetsnivå. Inspirert av Boltanski i Wagner (1999:349) og Thévenot (Thévenot 2001b:61–71) kaller jeg disse det *familiære* regimet, regimet for *tilpasning og målrettet handling*, regimet for *makt og motstand*, og regimet for *rettferdiggjøring* (Tabell 1). Boltanski og Thévenots regimer er generelle, jeg har derimot prøvd å vinkle beskrivelsene mot handlinger i organisasjoner. Det pragmatiske rammeverket beskjeftiger seg hovedsakelig med regimet for *rettferdiggjøring*, som er konfliktfylt men samtidig styrt av at aktørene må etablere en form for ekvivalens – likeverdighet – for å kunne diskutere et problem.

	Lav refleksivitet / non-ekvivalent	*Refleksivitet / ekvivalent*
Fredelige	Det familiære	Tilpasning / målrettet handling
Konfliktfylte	Makt / motstand	Rettferdiggjøring

Tabell 1: Handlingsregimer

Mange av de sosiale aktivitetene mennesker foretar seg på jobben, foregår i det Boltanski kaller det familiære handlingsregimet. Her er handlinger regulert av vaner, avtaler og konvensjoner som tas for gitt. Handlinger i dette familiære handlingsregimet kan kalles nonrefleksive og fredelige, siden de svært sjelden trenger rettferdiggjøring. Rutinemessig arbeid og vanlige sosiale interaksjoner faller inn i denne kategorien. Når man først har lært seg å utføre en rutine eller en teknikk, kan man fortsette å reprodusere denne uten å være bevisst på den. I mitt utvalg er det mange som har rutiner for å sjekke e-post, eller sjekke test-status på koden de programmerte dagen i forveien. Andre handlinger, ikke direkte relatert til arbeidet, foregår også i dette regimet. Om personer er ekvivalente eller har ulik status spiller liten rolle når man omgås ved skriveren, kaffemaskina eller toalettet. Både ledere, konsulenter og sekretærer kan oppholde seg i samme rom uten å måtte forklare seg eller forhandle om sin verdighet. Deres stillingstitler spiller liten eller ingen rolle for handlingene som utføres. Internaliserte normer og verdier, som arbeideren tar for gitt og som er uttrykk for den generelle forståelsen av arbeidet, hører hjemme i dette regimet. Smidige metodologier – og andre modeller som innføres i organisasjoner, retter seg blant annet inn på å få de ansatte til å reflektere mer over de mindre refleksive handlingene sine.

Når en aktør skal koordinere aktiviteter og gjøre avtaler med andre, befinner de seg i et refleksivt regime som forutsetter fredelig samhandling, og en avklaring av ekvivalens. Opplæring av nytt personell, mentorfunksjoner og møtevirksomhet befinner seg ofte i dette regimet. Man

legger planer, rapporterer og informerer, men det er ikke tilfeldig hvem som sier hva eller hvem som har autoritet til å uttale seg. Status og ulike roller og stillinger fyller ulike funksjoner, men alle samarbeider for å dra lasset i samme retning. Mange teorier om ledelse og styring av organisasjoner er opptatt av problemstillinger innad i dette handlingsregimet. Arbeidsavtalen kan forstås som en fredelig regulering av det ulike styrkeforholdet mellom partene i et arbeidsforhold.

Det finnes også et regime for non-ekvivalente, konfliktfylte handlinger. Jeg har kalt dette regimet for *makt/motstand*. Her foregår alle strategiske handlinger og planer som ikke legitimeres, Det er anerkjent at subjekter ikke er ekvivalente. I organisasjoner kan det oppstå uenighet om prioriteringer, om mål, metoder, eller mer generelle frustrasjoner, som gjør seg utslag i direkte motstand. Eksempler på dette er sabotasje, tyveri, løgn, unngåelse eller former for vold (Giacalone og Greenberg 1997). Arbeidere som blir frustrerte kan bevisst gå inn for å sabotere ledelsesstrategier ved å motarbeide og aktivt eller passivt underminere mål. Dette kan foregå diskret, for eksempel gjennom kontroll og manipulasjon av ressurser som andre trenger, eller mer åpenlyst gjennom å stadig uttale seg nedsettende om andre personer- eller gruppers mål- eller arbeid. Denne typen handlinger kan være en indikator på legitimitetsproblemer i en organisasjon.

En studie av motstand mot organisasjonsendringer foreslår at Boltanski og Thévenots verdiverdener kan brukes for å analysere denne typen handlinger (Fronda og Moriceau 2008). Fronda og Moriceau deler inn motstand i tre typer: *revolusjon*, som er direkte og konfronterende, *tilbaketrekking*, som innebærer å trekke seg tilbake til den mer private sfære hvor identiteten er tryggere, og *diskret motstand,* som f.eks. utvist blant ledere på mellomnivå som har ansvar for å gjennomføre endringer i en organisasjon, men som verken klarer å legitimere endringene ovenfor arbeiderne eller mobilisere gode argumenter til ledelsen om hvorfor de ikke ønsker å gjennomføre endringene. På makronivå vil fusjoner og oppkjøp potensielt være konfliktfylte og nonekvivalente.

Det fjerde handlingsregimet er forbeholdt legitimering og rettferdiggjøring. Dette er en aktivitet som er konfliktfylt og som krever ekvivalens – og det er dette regimet som vanligvis regnes som mest interessant i forhold til Boltanski og Thévenots teoretiske rammeverk. Her foregår det komplekse, konfliktbaserte sosiale forhandlinger som krever rettferdiggjøring og kritikk, og som har potensiale til å skape kompromisser i organisasjoner (Jagd 2011). Hvis en part inngår i en debatt med en annen part, anerkjennes i utgangspunktet en ekvivalens – og man må diskutere for å komme til en avgjørelse og en mulig løsning på problemet. Her spiller verdighet basert på f.eks. titler, profesjonell kunnskap eller erfaring en stor rolle. Profesjonelle identiteter møtes og bryner seg mot hverandre for å finne ut hvem som har mest rett i ulike situasjoner.

Svaret er ikke gitt på forhånd, diskusjon kan bidra til en foreløpig avklaring. På makronivå foregår denne typen disputter mellom for eksempel stat, fagbevegelse og organisasjoner. I moderne vestlige stater løses mange disputter på makronivå mellom aktører med etablert ekvivalens, men gjennom historien kan institusjoner miste så mye legitimitet og tillit at kritikken har flyttet seg over i et non-ekvivalent handlingsregime (hos meg kalt regimet for makt og motstand). Kjente eksempler er for eksempel de voldelige opptøyene i Paris i mai 1968 og Athen 2010-2011. Voldelige hendelser kan også forstås som kritikker som det etablerte systemet må forholde seg til, og som kan drive fram en systemendring. Resten av avhandlingen vil i hovedsak ta utgangspunkt i regimet for rettferdiggjøring.

Alle større debatter, for eksempel om profesjonell styring av universitetene, kan forstås ut fra dette handlingsregimet for rettferdiggjøring. I denne typen disputter settes ulike idealer og målestokker opp mot hverandre. I tilfellet profesjonell styring blir ofte prinsippet om *effektivitet* satt opp mot prinsippet om *faglig og profesjonell autonomi*. Partene kan imidlertid ikke bare fremføre argumentene sine som perler på en snor. De må i tillegg argumentere for eksistensen og gyldigheten av sine prinsipper og idealer, og koble disse til objekter i virkeligheten og vise at denne sammenstillingen leder mot en bedre virkelighet for alle. Jeg tar utgangspunkt i at forhandlinger om *meningen med arbeidet* og *profesjonelle grenser* og *identitet* også til en viss grad foregår i dette handlingsregimet. For at nye management-idealer og praksiser skal bli institusjonalisert, må de først passere hos arbeiderne og bli akseptert som til det felles beste. Ofte er det umulig å bli enige om hva som er det felles beste, og man blir enig om å møtes i et kompromiss. Hvis man er del av en levende arbeidsplass med mange diskusjoner, debatter og grenseforhandlinger, er det kanskje lettere å utvikle et felles språk og en profesjonell identitet og stolthet. Nye fagfelt – som for eksempel programmeringsfaget – har gjerne flere faglige spenninger og konflikter enn for eksempel medisinfaget, de er mindre "satt", og er derfor også interessante studieobjekter å analysere i et legitimeringsperspektiv.

4.4 Et mangfold av verdier og idealer

Mangfoldet av idealer, tolkninger og verdenssyn som eksisterer i samfunnet er utgangspunktet for den sosiale kritikkens uunngåelighet og potensial. Boltanski og Thévenot utviklet rammeverket sitt for å analysere måter legitimering blir brukt for å forhandle om mening og oppnå en enighet eller et kompromiss. Sentralt for dette rammeverket er seks verdigrammatikker eller verdioffentligheter.[16] Disse skiller seg fra verdiverdener, gjennom at de er

16 I den engelske oversettelsen av De La Justification brukes forestillingen *Polities of worth*. Polity er imidlertid vanskelig å oversette direkte til norsk, det måtte i så fall ha blitt *stat*, *styre* eller *kommune* – ingen av dem gir en god

moralske, rene former – ikke ulikt måten Platon så for seg at en perfekt sirkelform bare kunne eksistere i idéverdenen. Det at verdier kan organiseres i et verdisystem, betyr ikke at de nødvendigvis hører hjemme i en logikk med sterke grenser mot andre logikker eller verdiverdener. Hva som kan regnes som en verdioffentlighet eller en verdilogikk er begrenset ved at det øverste prinsippet må kan organiseres i henhold til seks aksiomer: Det må være mulig å identifisere og inkludere alle i en *felles menneskelighet* (common humanity) (a1), og det må gå an å differensiere mellom ulike *nivåer* (states) (a2) av en *felles verdighet* (a3). Nivåene må videre kunne *rangeres* (a4). Det må finnes en *formel for investering* (a5) og en klar definisjon på hva som regnes som *allmengyldig godt* (a6). Resultatet er et rettferdighetsprinsipp som kan regnes som universelt og meritokratisk innenfor en grammatikk eller offentlighet, og som kan brukes for å redistribuere verdighet mellom "store" og "små" i tråd med begrensninger og muligheter i denne (Boltanski og Thévenot 2006:74–77). Boltanski og Thévenot identifiserte disse seks verdigrammatikkene fra kanoniske, moralfilosofiske tekster om hva som regnes som godt. Den inspirerte verdenen ble utledet fra St. Augustin, den hjemlige verdenen fra Bossuet, verdenen for berømthet fra Hobbes, den kollektive verdenen fra Rousseau, markedsverdenen fra Adam Smith og den industrielle verdenen fra Saint-Simon. Disse moralfilosofiske verdioffentlighetene er abstrakte, og for formelle til å kunne brukes i sosiale situasjoner. Men når de settes i sammenheng med objekter fra virkeligheten, danner de konkrete verdiverdener (Boltanski og Thévenot 2006:43–123; Jagd 2011:346).

Håndbøkene og manualene som utgjorde analyseobjektet i *The Justification* og *The New Spirit of Capitalism* er skrevet for å gi råd og veiledning til ledere, managere og ansatte, og valgt ut fordi bøkene viser eksplisitte problemer og foreslår ulike "verdige" løsninger på disse (Boltanski og Chiapello 2005:62–63; Boltanski og Thévenot 2006:151). Bøkene er hovedsakelig fra 60-tallet og 90-tallet. Utvalget av management-bøker i *The Justification* ble gjort fordi tekstene fokuserte på en av de seks verdenene som forfatterne ville dokumentere, og litteraturen beskrev hvordan man kunne danne situasjoner eller settinger med minst mulig spenning: Hvordan stimulere menneskers kreativitet, hvordan kultivere passende relasjoner i bedriften og gjøre et godt inntrykk på sjefen, på underordnede, for kolleger, besøkende eller klienter; hvordan fremme bildet av en bedrift, en person eller et produkt, og hvordan bruke eget image til din fordel i public relations, hvordan organisere rettferdige valgsystem for å velge ut kandidater til ulike funksjoner i bedriften, hvordan organisere en bedrift for inntjening eller hvordan gjøre bedriften mer effektiv. Tekstene viste praktiske og konkrete råd på problemer, knyttet til prinsipper i de

forståelse av hva begrepet innebærer. Jeg bruker derfor ordet grammatikk og verdioffentlighet for å beskrive disse rene formene.

kanoniske tekstene – men helt uten referanse til politisk filosofi eller noen av forfatterne (Boltanski og Thévenot 2006:152). Alle metodebøkene har forbindelser til den industrielle verden, gjennom at de i seg selv skaper en metodologi. Både på 60-tallet og på 90-tallet var det anerkjent at profitt i seg selv ikke er et veldig inspirerende mål (Boltanski og Chiapello 2005:63). Mange av forfatterne, både på 60-tallet og på 90-tallet, var derfor opptatt av å definere en mening med arbeidet, eller kapitalistisk ånd – som formulert av Boltanski og Chiapello (2005:64).

Det er umulig å se for seg en organisasjon som aldri endrer seg, hvor arbeiderne jobber uten å reflektere, tvile eller noen gang være i brytninger med kolleger. På denne måten går det an å forstå den sosiale verden som ustabil og alltid under konstruksjon. Dette skaper en usikkerhet, en sosial uro. Ulike måter å engasjere seg med omverdenen, institusjoner eller semantiske strukturer er måter å håndtere denne usikkerheten på (Blokker 2011:253; Boltanski og Chiapello 2005:30). Å engasjere seg i debatter med legitimering blir da en måte å forsøke å skape en felles forståelse og mening, og hvis man ikke klarer å komme til enighet kan man i hvert fall komme nærmere å forstå hva man er uenig om. *Verdiverdenene* utgjør i dette perspektivet et repertoar av evalueringer som aktørene bruker for å legitimere argument og handling i virkeligheten. Verdiverdenene som er presentert av Boltanski og Thévenot er den *artistiske* verden, den *hjemlige* verden (domestic), verdenen for *berømthet*, den *kollektive* verdenen, *markedsverdenen*, den *industrielle* verdenen og den *prosjektorienterte* verdenen. Alle disse er koblet til blant annet en generell form for verdighet, en formel for investering og et øverste prinsipp som subjekter og objekter kan måles etter (Tabell 2). Tabell 2 er for øvrig en svært forenklet modell av verdiverdenene. For en komplett oversikt over den industrielle og den prosjektorienterte verdiverden se tabell på side 80 og 83, eller Boltanski og Thévenot (2006:127-211) og Boltanski og Chiapello (2005:107-128).

Verdiverdener er kognitive kategoriseringer hos aktører, som kan ha en viss gyldighet på mikro- og makronivå. De uttrykkes gjennom språk, men representerer samtidig mer enn bare et vokabular. Verdiverdener består av prinsipper, relasjoner, og referanser til verdighet, subjekter og objekter som aktørene tillegger viktighet og som gir mening for dem. Dette betyr ikke at aktører uten videre kan forvente å få forståelse for å benytte kritikker fra en hvilken som helst av disse verdiverdenene i enhver situasjon, de må først bestå en *kvalifisering* eller *gyldiggjøring*.[17]

17 Boltanski og Thévenot bruker ordet "qualify" i engelsk oversettelse (som jeg oversetter med gyldiggjøring/kvalifisering) til å beskrive (a) hvordan noe kan beskrives gjennom sammenligning (å vurdere ekvivalens), (b) hvordan dette utgjør en kategorisering som også forutsetter (c) inskribering av en performativ kode og (d) referanse til et felles beste (Boltanski og Thévenot 2006:1, 359).

Verdiverden	*Øverste prinsipp*	*Verdighet*	*Investering (hva ofres)*
artistisk	Det inspirerte	Kjærlighet, lidenskap, kreativitet	Rutiner og stabilitet
hjemlige	Tradisjon, generasjon, hierarki	Vaner, karakter, å ha naturlig autoritet	Egoisme, selviskhet (i stedet påtar man seg plikter)
berømthet	Offentlig anerkjennele	Å bli gjenkjent, å nyte respekt	Anonymitet og hemmeligheter
kollektiv	Den kollektive vilje, alle	Sivile rettigheter, deltakelse	Personlig vinning
marked	Konkurranse, rivalisering	Behov, egoisme, kjærlighet til ting	Sympati, emosjonell distanse
industriell	Effektivitet, ytelse	Arbeid, energi, å utnytte et potensial	Innsats, investeringer (i tradisjonell forstand)
prosjekt	Aktivitet	Å knytte sammen subjekter (og objekter)	Alt som vanskeliggjør tilgjengelighet eller bevegelse

Tabell 2: Verdiverdener (forenklet). Etter Boltanski og Thévenot (2006) og Boltanski og Chiapello (2005).

Innholdet i en verdiverden både *muliggjør* og *begrenser* ytringer og handlinger. Det er dermed aktører som skaper, reproduserer og endrer grensene mellom verdiverdener, og som kan gyldiggjøre hvilke prinsipper som regnes som logiske og verdifulle. Verdisystemet i bunn av en verdiverden eller logikk er koherent på den måten at både innholdet og det øverste prinsippet for evaluering må gjenspeile en egen *verdighetstilstand.* Kunnskap om hvilken verdighetstilstand som er gyldig i ulike situasjoner besitter aktørene selv, og dette utgjør menneskers kritiske kompetanse (Boltanski og Thévenot 1999).

Det finnes ikke noe prinsipp som blir regnet som gyldig i enhver situasjon. Selv den mest inspirerte artisten vet at hun ikke kan la inspirasjon styre sosial koordinasjon i alle sosiale sammenhenger. For ikke å bli oppfattet som gal må hun stå i kø på postkontoret som alle andre (Boltanski og Thévenot 2006:216). Det vil imidlertid være legitimt i de fleste situasjoner, både blant kunstnere, kunstkritikere og lekpersoner, å begrunne en uttalelse om et maleri med henvisning til egne *følelser*, og snakke om for eksempel *inspirasjon* og *kreativitet* – forhold som regner verdi ut fra en ekstern, abstrakt kilde som ikke er konkret målbar. På samme måte vil det ikke oppfattes som legitimt å kvalifisere en kunstkritikk ut fra en industriell verden, hvor verdien først og fremst hentes fra målbare kriterier som produksjon og effektivitet. Man kan ikke *måle* at kunst er *effektivt*, å ytre noe sånt vil bli regnet som dumt eller usaklig. En kritikk må altså fremføres på en *gyldig* måte hvis den skal bli tatt seriøst (Boltanski og Thévenot 2006:227). Kritikken må ha altså en gyldig syntaks, som gjenspeiler verdenens overordnete verdi (ekvivalensprinsipp). I den artistiske verdenen er dette ekvivalensprinsippet *inspirasjon*. Kunstens

verdi kan da regnes ut fra hvor mye *kjærlighet* eller *lidenskap* som er lagt ned i arbeidet, eller hvor *orginalt* arbeidet er. Ofte vil det også regnes som helt legitimt å henvise til andre *kunstnere* eller eventuelt *ånder* (at kunstneren var besatt), gjerne i form av konkrete referanser til *genier*. Objekter som regnes som gyldige å vise til er knyttet til *kropp* eller *sinn*, man kan for eksempel referere til *følelser*, *drømmer* og *rus* (Boltanski og Thévenot 2006:160–161).

Ekvivalensprinsippet i den industrielle verdenen er derimot basert på *produksjon*, *effektivitet* og *ytelse*. Her måles verdighet etter *energi* og *potensiale*, mens subjekter man ofte henviser til er *eksperter* og *ingeniører*. Objekter som regnes som gyldige å vise til er *verktøy*, *oppgaver*, *sannsynlighet*, *lister*, *kriterier* osv. En verdiverden begrenser i den forstand at man ikke uten videre kan blande for eksempel ekvivalensprinsipp, subjekter og objekter fra forskjellige verdiverdener i én uttalelse. Dette er den kritiske kompetansen som alle aktører eller medlemmer i fellesskapet må besitte for ikke å dumme seg ut og potensielt tape verdighet. Det vil neppe bli regnet som kvalifisert kritikk hvis noen sier "Det bygget der er så vakkert, det er tydelig at det ble malt av en formann som hadde gode verktøy." Likeledes går det ikke an å skrive i en rapport "Målstyringsmodellen gir tydelig mer effektiv undervisning; sinnet vårt frigjøres og vi føler oss mer levende". Kritikken aktørene bruker, er formet ut fra gitte prinsipper, og de må ha en koherent syntaks for å bli oppfattet som gyldige. Det er imidlertid ingenting i veien for at man kan kritisere forhold ut fra flere verdiverdener til samme tid. Arkitektur kritiseres ofte ut fra både en artistisk og en industriell verden. En bygning kan være konstruert sånn at den *føles* behagelig, men også sånn at den er *energieffektiv*. Mange ganger kan kritikker henvise til flere evalueringsprinsipp til samme tid. Hvis kritikk fra flere verdiverdener får moment, vil kritikken kunne stå sterkere og appellere til flere.

Bruken av evalueringsrepertoar er *kulturelt* betinget. Franskmenn har andre kategorier og definisjoner av det gode enn USA-amerikanere (Lamont and Thévenot 2000). På samme måte har profesjoner og organisasjoner gjerne et lokalt repertoar. Roch (2004) fant at ved å bruke de samme prinsippene for legitimering, kan to organisasjoner nærme seg og integreres raskere enn ved å dele den samme teknologien eller det samme distribusjonsnettet. Over tid kan verdiverdener miste sin legitimerende kraft i gitte situasjoner, bli endret eller erstattet av nye verdiverdener.

4.5 Virkelighetstester

Kritikker har den egenskapen at de trykker på sosiale systemers "ømme punkter", samtidig som de mobiliserer forslag til forbedringer. Hvis de skal ha en effekt, må imidlertid idealer,

legitimeringer og kritikker relateres til virkeligheten gjennom *virkelighetstester*. Alle sosiale relasjoner kan analyseres som tester av ulike variabler som må bestå for at relasjonen skal *reproduseres*. Testen har kraft til å transformere hvilke legitimeringer en relasjon er bygd opp av, hvis det skulle være nødvendig. Det er dette test-rammeverket som er originalt for Boltanski og Thévenot. Rammeverket gjør det mulig å se sosiale prosesser i lys av et mangfold av verdier, bygd på både legitimering eller styrke/vold, og er mulig å anvende både på aktør-nivå og for å forklare reproduksjon av større kulturer. Tester henter sin legitimitet med referanse til verdier, subjekter eller objekter som hører hjemme i verdiverdenener. Rettferdiggjøring har da den effekten at den tester – og hvis testen blir bestått – reproduserer en relasjon som blir oppfattet som ekvivalent. Vold kan også reprodusere en relasjon, men da er relasjonen i så fall non-ekvivalent. Jeg vil i det følgende forklare dette testrammeverket. Jeg begynner med å beskrive en relasjon mellom to aktører, og utvider gradvis til å beskrive testene på et kulturelt plan. Til slutt viser jeg hvordan den sosiale relasjonen *trafikk* blir kritisert og utfordret med referanse til verdighet, og hvordan denne lar seg reprodusere på en relativt ordnet måte.

Et vestlig parforhold mellom to mennesker kan ofte forstås som en relasjon som blant annet består av en test av *tillit*, og denne testen må fornyes for at relasjonen skal kunne fortsette. I verdiuniverset til Boltanski og Thévenot regnes tillit som verdig i den hjemlige verdiverden. Gjennom en lengre relasjon vil imidlertid denne testen av tillit bli satt på prøve. I utgangspunktet er det gjerne sånn at parforholdet er avhengig av at denne testen av tillit reproduseres. Hvis testen feiler, må den endres og re-testes til den består på nytt for at parforholdet skal kunne vare. I testen ligger også muligheten for at parforholdet kan transformeres, og bli basert på nye typer legitimitet. Et nytt subjekt (et barn) eller et objekt (en hytte som krever mye arbeid) – kan skape nye tester av tilliten. Eventuelt, hvis det skjer noe som gjør at forholdet ikke lenger kan basere seg på tillit, må relasjonen finne ny legitimitet med henvisning til andre prinsipper for å kunne la seg reprodusere. Hvis den ene har vært utro må paret muligens finne ut om relasjonen kan fortsette ved hjelp av en annen test, som kanskje tar hensyn til barna eller det å holde en utvendig fasade. Relasjoner kan også forstås som tester som er stabile og nonekvivalente, f.eks. ved at en part betaler en annen for at relasjonen skal fortsette.

Virkelighetstester kan ta form av legitimitetstester eller styrketester. Begge testene måler styrke, men en test er en legitimitetstest bare så lenge den forsøker å hente legitimitet ut fra prinsippene i en verdiverden. I sosiale situasjoner vil en legitimitetstest være begrenset av prinsippene i en eller flere verdiverden(er), mens en styrketest kan brukes for å trumfe gjennom et argument uten å referere til verdiverdener.

4.5.1 Rangeringer

Legitimitetstester har rom for rangeringer med referanse til ulike verdiverdener. I den industrielle verdiverdenen er det høyeste ekvivalensprinsippet effektivitet – det avgjørende spørsmålet er om X er mer effektivt enn Y, eller visa versa. Ut fra dette prinsippet kan man teste hvem (subjekt) som er mest effektiv i arbeidet. Denne logikken åpner også for å måle subjekter mot objekter gjennom tester som er *interne* innen verdiverdenen – man kan vurdere om en robot flytter kassene mer effektivt enn to menn, eller gjøre seg opp en mening hvorvidt ulike former for relasjoner mellom subjekter og ting er mer eller mindre effektive. I den inspirerte verden er det høyeste idealet "the outpouring of inspiration" - det som inspirerer. Siden *inspirasjon* altså kan anta form som et høyeste ideal, kan vi forstå at Gud tillegges mer viktighet enn for eksempel industriell effektivitet i enkelte miljøer, eller at å røyke hasj eller spille dataspill med venner potensielt blir oppfattet som av høyere verdi enn å ta en utdanning. Det gir rett og slett mer glede og inspirasjon, det kjennes bedre ut, man kjenner det i *hjertet*. Hvis en spiller ytrer at "*dataspill gir meg inspirasjon og glede, men min bror kan ikke oppleve dette fordi han er en overarbeidet streber*" i et sosialt fellesskap av dedikerte spillere, vil de andre sannsynligvis bekrefte testen og dermed forsterke dette virkelighetssynet. Ifølge den prosjekt-orienterte verdenen er det høyeste prinsippet å være i aktivitet og være del av ett større nettverk. Virkelighetstester her måler for eksempel i hvilken grad subjekter klarer å gå fra prosjekt til prosjekt på en verdig måte. Det å forlate et suksessfylt prosjekt som man har bidratt til å øke verdien på, og så integrere seg (suksessfylt) i et nytt prosjekt, og å gi andre mennesker et større nettverk bidrar til økt verdi for alle og bestått test (Boltanski og Chiapello 2005:125).

Tester kan også være *rene styrketester* – og henvise til verdier som ikke er gyldige i en verdiverden. Alle legitimitetstester begynner faktisk som en styrketest, før testen kan institusjonaliseres og til slutt bli anerkjent som mer rettferdig. Mange stater hadde for eksempel innført stemmerett mot slutten av 1800-tallet, men denne gjaldt bare menn. Mange mente derfor at demokratiet egentlig ikke hadde legitimitet, det var ikke rettferdig, det kunne ikke kalles ekte. Hvis demokratiets legitimitet skulle bestå denne testen, måtte kvinner inkluderes. Forkjemperne argumenterte for dette ut fra en kollektiv verdiverden med referanse til likhet og menneskeverd. Andre anerkjente ikke hensyn til at likhet og menneskeverd var gyldig i tilfellet kvinner. Mange måtte engasjere seg og raffinere kritikkene, og understreke viktigheten ved hjelp av demonstrasjoner og aksjoner. Til slutt klarte tilhengerne å presse gjennom dette enkle poenget, som i dag er helt legitimt og tatt for gitt når det kommer til stemmerett: At demokratiet ikke har legitimitet hvis halve befolkningen ikke får lov å stemme.

Over tid vil styrketester kunne transformeres til legitimitetstester og begrense mulighetene aktører har til å bruke styrke for sin egen vinnings del. I et samfunn hvor det eksisterer mange legitimitetstester, har de sterke mindre makt. Begrensningene som ligger i legitimitetstestene vil gjøre det vanskeligere for sterke å investere denne styrken på tvers av verdiverdener. Eksempel fra dagens samfunn som begrenser rike personer, er at forfattere ikke kan betale en litteraturkritiker for å bli anerkjent som en inspirert forfatter, eller at man ikke kan få jobb i kommunen bare fordi man kjenner ordføreren (Boltanski og Chiapello 2005:31–32).

Verdiverdenene legger føringer for sosiale relasjoner, som igjen regulerer hva som regnes som akseptable og verdige handlinger innenfor en test. En kunstner som ofte lar et image om at han/hun er styrt av "ren inspirasjon" styre handlingene sine i offentligheten, kan ikke forbli i en inspirert tilstand og svare godt for seg hvis han/hun blir konfrontert med skatteunndragelse. Hvis kunstneren forsøker å bestå rettssystemets test ved å snakke om *drømmer*, *aura* og *smerten ved å være et geni* (som alle hører til i den inspirerte verdenen) vil testen feile.

4.5.2 Korrigerende og radikal kritikk av testen

Framføringen av *kritikk* og *tester* henger nært sammen gjennom at de kan utfordre den eksisterende orden, og så tvil om statusen til de/det kritiserte – før testen (eventuelt) reproduseres. Men tester er igjen mulig å kritisere. Dette er kjernen i disputter. Virkelighetstester settes opp mot hverandre og kritiseres ut fra idealer i verdiverdener. Det er ikke gitt på forhånd hva som skal regnes gyldig. Det er også to måter å kritisere tester på, *korrigerende* og *radikal*.[18] Korrigerende kritikk tar testene på alvor, men holder seg innenfor samme verdiverden (Boltanski og Chiapello 2005:32–3). Blant to ingeniører kan det oppstå en disputt hvorvidt en ny rutine virkelig er mer effektiv eller ikke, men begge er enige om at prinsippet for effektivitet er det som avgjør testen. To hasjrøykere kan være uenig om hvilken type hasj som gir den beste følelsen, men begge vil møtes i prinsippet om at hasjrus er det overordnete målet som kan og skal gi en god følelse. To teologer kan være uenige om en tolkning i en bibelpassasje, men enige i at Gud er den eneste gud. I en disputt innenfor en verdiverden, spiller også subjektenes status, eller verdighet, en rolle. Ingeniøren som har den høyeste tittelen vil muligens ligge an til å kunne vinne en disputt gjennom statusen dette gir. Hasjrøykeren som skaffer sin egen dop vil ligge an til å vinne en disputt mot en røyker med mindre erfaring. Men tester – jo strengere de er – gir subjekter med mindre status en mulighet til å bryne seg mot kjempene – og man kan møtes til strid, regulert av like kriterier.

Det finnes mange etablerte rettferdigghetstester i samfunnet, ikke direkte knyttet til en

18 Som igjen henger sammen med reformerende og revolusjonær kritikk (Boltanski og Chiapello 2005:33).

verdiverden, men gjerne mulig å analysere i lys av en eller flere verdiverdener. Dette kan være for eksempel politiske valg, eksamener, idrettskonkurranser og felles forhandlinger mellom parter i arbeidslivet – alle med klare, strenge reguleringer for hva som skal regnes som gyldig og ikke, og hvor de med mindre status har en sjanse til å konkurrere på like vilkår mot personer med større verdighet. Jo strengere testene er, jo mer rettferdige er de i forhold til å utjevne statusforskjeller blant subjekter. I idrettskonkurranser og ved eksamener er reglene så klare, at de kan anses som legitimitetstester som det vil være umulig å ignorere resultatet eller gyldigheten av. Og de fleste er enige om at resultatet av testen er rettferdig, eller i hvert fall så rettferdig som det kan bli innenfor testpremissene.[19] Men selv denne typen tester kan kritiseres. Hvis en korrigerende kritikk blir gjentatt mange nok ganger kan den ikke ignoreres, og testene blir nødt til å komme med en respons på kritikken. Responsen kan forsøke å hevde at kritikken er feil, at den ikke vil føre til større rettferdighet – og må da knytte dette svaret til en gyldig referanse fra virkeligheten. Eventuelt kan man gjøre testen enda strengere, sånn at kritikken ikke lenger blir gjeldende.

Et eksempel fra virkelighetens verden er eksamener ved universitetene – som holdes årlig, og som hvert år får kritikker om at de ikke er rettferdige med referanse til dagens moderne teknologi. Mange studenter skriver aldri for hånd til vanlig, og er vant til å gjøre eksamener på PC fra videregående. Når de kommer til universitet er de imidlertid nødt til å skrive besvarelsene for hånd, på papir. En av kritikkene som kom opp våren 2011 ved UiB var at å skrive for hånd er

19 Dette er noe av det vakre med arrangementer som Birken og Oslo maraton i mine øyne. Privatpersoner kan konkurrere med de etablerte, sponsede idrettsheltene på samme vilkår i testen. Resultatet vil være sant, trygt, sikkert – og ingen kan betvile resultatet uten å bevege seg utenfor testrammene. Hvis en deltaker på forhånd følte en uro, eller at deres posisjon i verden var utrygg kan de i hvert fall få bekreftet at deres posisjon relativ til subjekt X (med høy verdighet) bare er Y antall plasser bak, og dessuten foran de fleste andre. Dette kan skape mening og trygghet hos sarte sjeler, uten at det er den eneste motivasjonen for å delta. Denne forståelsen er for øvrig kompatibel med det synet at mange personer som deltar på denne typen arrangement er i livskriser (typisk: 30-, 40- eller 50- års-krise). Forklaringen gir en grunnleggende annerledes fortolkning på motivasjonen for å delta i denne typen konkurranser enn typiske bourdieuske forståelser som dominerer i mediene (se f.eks. Hjelseth 2011), hvor motivasjonen for å delta blir forstått som at aktørene (mer eller mindre bevisst) ønsker å være medlemmer i en sprek/sunn klasse. Den pragmatiske forståelsen foreslår at det kan være mange grunner til at aktørene investerer i trening (hvis vi vil vite hvorfor må vi nesten spørre dem), men ved å *teste* kan aktørene få en *sikker* bekreftelse på hvorvidt investeringen har vært god. En pragmatisk forklaring på at mange fra intelligentsiaen (som i Hjelseths tekst gjelder skribenter, forfattere, kunstnere, professorer og journalister) trener og deltar i konkurranser, vil imidlertid også kunne åpne for at arbeidet, som de en gang trivdes med og var stolte av før, ikke lenger gir mening, trygghet eller verdighet – i forhold til hvor mye de investerer. Å skrive en artikkel eller kronikk som raskt blir glemt i mediemangfoldet reproduserer kanskje ikke mening i dagens medievirkelighet, men å løpe i skau, mark og på landevei – det gir mening. Da opplever kanskje "intelligentsiaen" at de får noe igjen for investeringen, uansett om dette er bedre helse, endorfinkick, vennskap, prestasjonsfølelse eller tid til meditasjon.

frustrerende fordi skriften blir stygg, man får muskelkrampe og det er svært uvant for elevene å skrive uten å kunne redigere (Svendsen og Agdesteen 2011). Dette er en variant av en kritikk som studentene har fremmet helt siden det ble vanlig å bruke hjemmedatamaskin, og i dag blir den også fremmet og forsterket av ulike studentorgan. Universitet i Bergen hevder at å tillate bruk av PC vil gjøre testene mer urettferdige, siden det blir lettere å jukse. Viserektor Kuvvet Atakan sier imidlertid også at han forstår kritikkene av testen, og sier at UiB har planer om å løse problemet med å opprette et eksamenssystem med egne, kontrollerte PC-er for eksamenformål. I mange år har UiB ignorert kritikken, men etter hvert har kravet om å få slippe å bruke penn og papir blitt mer og mer legitimt – og UiB må antakelig tilpasse seg studentenes krav snart.

Kritikken kan også være radikal, ved at den helt ser bort fra de etablerte testkriteriene, og søker å erstatte testen med en helt ny test. Denne kan låne legitimitet fra andre verdiverdener eller henvise til noe helt nytt. For å gå tilbake til eksempelet med et parforhold, så kan dette skje hvis den ene parten får Alzheimers, og gradvis begynne å glemme. Testen av tillit vil ikke bestå på sikt, og må erstattes med en ny test hvis forholdet skal ha en sjanse til å fortsette i et likeverdig forhold (og til slutt vil det bli umulig å møtes som ekvivalente). Relasjonen kan imidlertid fortsette en stund til hvis den lar seg reprodusere gjennom etablering av tester basert på tålmodighet og omsorg.

4.5.3 Tester av verdighet i komplekse sosiale system - trafikk

I subkulturer kan lokale evalueringsrepertoar gjelde, og for medlemmene kan de interne verdiverdenene ha større legitimitet enn etablerte tester i samfunnet. Jeg vil nå forklare hvordan en subkultur kan utfordre et større samfunns etablerte test ved hjelp av korrigerende og radikal kritikk.

Motorferdsel på det offentlige veinettet – trafikken – er en kontinuerlig test på trygghet/likhet for loven i forhold til en kollektiv verdiverden. Testen er godt etablert i Norge, med ganske detaljerte regler som gjelder for aktører uansett status. Det vil si at de samme trafikkreglene gjelder, uansett hvor rik sjåføren er eller hvor rask bilen er. Hvis alle følger reglene er testen som regel selvbekreftende. Det finnes imidlertid en subkultur (må ikke forveksles med gruppa som kalles rånere, dette gjelder langt flere) som mener at de er *berettiget* til å overprøve denne testen. Her spiller verdighet en viktig rolle. Den subjektive følelsen av verdighet i denne subgruppa er gjerne knyttet til objekter som en "fin" bil, som regulerer evalueringer som markerer hvilke typer dekk, felger, merke, motortype eller andre egenskaper som skal telles som verdige. Enkeltpersoner i miljøet kan få en høy status gjennom at de behersker – ikke bare det riktige språket – men et helt sett av verdier og idealer. Dette utgjør et evalueringsrepertoar

bestående av lokale verdiverdener og kompromiss, som bare delvis alluderer til verdiene og idealene som finnes i resten av samfunnet. Siden høystatuspersoner kjenner repertoaret, har de gjerne også en bil med mange gyldige symbolske markører. Hva som er gyldig er ikke gitt, det er åpent for forhandling. Nedfelt i verdisettet ligger det imidlertid også forståelser som skiller mellom verdige og uverdige subjekter, som gjør at enkeltpersoner føler seg berettiget til å komme med legitim kritikk av testen (av rettferdighet) som sier at alle er like for veitrafikkloven. De ønsker å marginalisere viktigheten alle tillegger denne testen. Personene kan *ytre* en kritikk som hevder at at fartsgrensene generelt sett er for lave, at alle veier burde ha en høyere standard, og at syklister og andre myke trafikanter ikke har noe på en bilvei å gjøre uansett fartsgrense. Samtidig kan de komme med en radikal kritikk, som forsøker å erstatte testen om trygghet/likhet for loven med en styrketest, hvor ekvivalensprinsippet om likhet ikke lenger gjelder. Styrketesten erstatter likhet-for-loven med aktuelle spørsmål som *hvem har den raskeste bilen*, *hvem våger* mest. Dette fører til at medlemmer som konkurrerer om verdi i subkulturen bryter fartsgrenser, hindrer andres ferdsel, lager farlige situasjoner og unødig støy. Verdiene kan også internaliseres, og legitimere det synet at det er greit å kjøre fortere enn de "andre trafikantene" selv når ingen andre i subkulturen er tilstede. Å kjøre fortere enn fartsgrensa har blitt en praksis, som daglig reproduserer og strekker rekkevidden av testen – er egentlig veitrafikkloven lik for alle? Politi og veimyndigheter kan på sin side gjøre testen av trygghet/likhet for loven strengere, ved å sette ned fartsgrenser, øke kontroller og sette opp fysiske midtrabatter som begrenser sjåførene som mener de er mer verdt enn andre. Internt i subgruppen kan alle forsøk på å utfordre legitimitetstesten føre til en økt status, inkludert hvis man får bot – med et brått fall i status hvis man får ubetinget fengsel. Sentralt i dette verdisettet er eksplisitte forestillinger om andre subjekters verdighet, regulert ut fra biltype, hodebekledning og kjønn – i tråd med symbolsk grenseteori (Epstein 1992; Lamont 1992, 2000). For folk utenfra er det vanskelig å forstå og i hvert fall utfordre denne verdigheten. Politi og myndigheter kan på forhånd ikke utfordre disse personene. Uniformerte tjenestemenn og offentlige etater tilhører ikke subjekter med høy status.

4.5.4 Oppsummering

Det er ikke alltid at den fremførte kritikk av tester fører til endring, eller at den i det hele tatt rettes mot det eller den som kritiseres. Men hvis en kritikk oppfattes som legitim, vil den potensielt resonnere med flere, gjentas, forsterkes gjennom ytterligere testing og til slutt kunne føre til sosial endring. I dette lyset trenger alle sosiale system sine fiender og sine kritikere for å finne løsningen på problemer, innarbeide mekanismer som oppfattes som rettferdige og gjenvinne moralsk støtte i tider med endringer. Kritikk kan føre til endring på minst tre måter.

For det første kan tidligere idealer og prinsipper *miste gyldighet*, og gjøre plass for nye. Det er dette som skjer når et parforhold endrer seg og blir basert på for eksempel et felles huslån i stedet for lidenskapelig begjær. For det andre kan det kritiserte systemet endre seg i retning av noe som oppfattes som mer *rettferdig*, og det er dette som skjer når det blir mulig etter hvert å bruke PC for å svare på universitetseksamener. For det tredje kan det kritiserte systemet *tilpasses* til å omgå kritikken, og finne løsninger som verken er optimale eller mer rettferdige, men vanskeligere å kritisere (Boltanski og Chiapello 2005:27–29). Det er dette som skjer når samlebåndsarbeid som blir regnet som uverdig i forhold til inspirasjon (den artistiske verdenen) og lønnsomhet (markedsverdenen) i vesten, blir flyttet til land lenger sør og lenger øst.

4.6 Kompromiss

Handling og legitimering i virkelighetens verden er sjelden forankret i idealer plassert i rene verdiverdener. Rammeverket til Boltanski og Thévenot anerkjenner tre ulike måter for å forstå hvordan orden kan forenes når flere verdiverdener spiller inn. Den ene er *clarification in one*, hvor en verdiverden dominerer de andre verdiverdener. Den andre er *private avtaler,* som er lokale og mellom et fåtall aktører. Den tredje måten, som har fått mest oppmerksomhet i rammeverket, er *kompromisset*, rettet mot mer varige verdisystem med rot i flere verdiverdener. For at sosiale kritikker skal få så stor kraft at de kan føre til sosiale endringer, krever de bred mobilisering og evne til å gi et troverdig løfte om å tilfredsstille behov som det kritiserte systemet ikke kan. Et kompromiss vil imidlertid ikke være stabilt hvis aktørene forsøker å bli enige om hvilket av evalueringsprinsippene i de sammensatte verdenene som er det viktigste (Boltanski og Thévenot 2006:282-3, 336–8). Verdiverdenene kan bare danne et kompromiss hvis de kan møtes på trygg grunn. Stabile kompromiss bidrar til å rekonfigurere verdiverdenene, knytte dem til nye prinsipper. På lengre sikt kan kompromisset få kraft til å danne et grunnlag for en ny verdiverden. Prosjektverdenen i Boltanski og Chiapello (2005) er dannet i et kompromiss mellom den industrielle og den artistiske verdenen, og er analysert som en (relativt) stabil konstruksjon, med nye tester. Det at en arbeider eller leder skulle få økt verdi av å slutte i jobben og begynne i en ny organisasjon var uhørt på 50- og 60-tallet, og spesielt det at han/hun skulle få økt verdighet jo kortere og mer flyktige forbindelser som ble gjort. Det er denne forskyvningen, og institusjonaliseringen av en ny gyldig test, som Boltanski og Chiapello dokumenterer i *The New Spirit of Capitalism*.

5. Reproduksjonen av profesjonell identitet i organisasjoner

Verdiverdenene og rettferdiggjøringstestene kan brukes for å analysere alle sfærer som inneholder legitimering og ekvivalens. Jeg har i gjennomgangen av det teoretiske rammeverket forsøkt å tegne opp eksempler fra ulike situasjoner. I denne avhandlingen bruker jeg imidlertid dette rammeverket for å analysere kultur i organisasjoner. Boltanski og Thévenot argumenterer for at private organisasjoner (business enterprise) er interessante forskningsobjekter, fordi de er under konstante spenninger fra ulike verdiverdener (Jagd 2011:347). Verdiverdener hos Boltanski, Thévenot og Chiapello er for øvrig identifisert ved hjelp av management-tekster. Moderne organisasjoner og foretak er komplekse, styres av imperativer som stammer fra ulike former for idealer, verdier og generaliseringer. Konfrontasjonene mellom disse skaper spenninger og leder til mer eller mindre ustabile kompromiss. Organisasjoner kan dermed sees som "kompromiss-maskiner" (Thévenot 2001; Jagd 2011).

I dette kapitlet, som er spesielt relevant i forhold til den første analysen, vil jeg vise hvordan verdiverdenene danner et felles evalueringsrepertoar i organisasjoner, yrkesfag eller profesjoner, som aktører må bruke for å gjøre seg forstått og fremme egne argumenter i dialog med andre. Dette repertoaret er ikke låst til sosiale grupper, stillinger eller subkulturer, men kan i teorien realiseres av hvem som helst, hvor som helst. Det er videre en sammenheng mellom idealene som utgjør evalueringsrepertoarene i en organisasjon og mulige *tester* og *kritikker*. Repertoaret vil også overlappe med det generelle repertoaret i en større kultur, nasjon eller generasjon. En ytring vil imidlertid måtte struktureres i relasjon til hvordan andre forstår virkeligheten hvis det skal ha en sjans til å bli oppfattet og evaluert. Evalueringskategoriene aktørene bruker må dessuten fremføres på en semantisk gyldig måte; og være mer eller mindre semantisk koherent for at andre skal oppfatte argumentene som gyldige. Organisasjoner og faglige identiteter kan gjennom dette analyseres som sosiale system bestående av flere parallelle tester som reproduserer et mangfold av idealer gjennom kritikk og legitimering.

Jeg vil iløpet av dette kapitlet beskrive hvordan jeg har gått fram for å analysere intervjuene mine i lys av Boltanski og Thévenots rammeverk og symbolsk grenseteori. Sju poeng er sentrale:

1. *Grensetrekkinger* kan brukes for å forklare organisasjonskultur- og organisasjonsstruktur – inkludering er imidlertid like viktig som eksklusjon.
2. Strukturer i organisasjoner, samfunn og det tilgjengelige evalueringsrepertoaret legger *føringer* for hvilke grensetrekkinger, og dermed hvilke idealer som er mulige.
3. Virkelighetstester og kritikk er nødvendig for å *gyldiggjøre* forståelser av idealer og rettferdighet på arbeidsplassen, og gjennom dette kan verdien som tillegges personlige motivasjoner for arbeid og strukturer i organisasjonene forstås som *reprodusert* av aktører.
4. Profesjonelle idealer kan skape *interne logikker*, *verdigheter* eller *mikrokompromiss* i profesjoner og organisasjoner, og disse utgjør igjen basis for et evalueringsrepertoar som kan brukes og transformeres uavhengig av stillinger/roller/subkulturer.
5. Disse *interne logikkene* inneholder muligheter for å rangere subjekter, objekter og handlinger, og er sentrale for *identitetsformasjon* og *gruppedannelse*.
6. *Endringer* i organisasjoner og samfunnsbetingelser, samt framveksten av ny teknologi vil kunne reflekteres gjennom å muliggjøre nye virkelighetstester, som igjen påvirker hvilke kritikker og forståelser som er mulige.
7. De legitime, gyldige oppfatningene av hva som er god, profesjonell praksis vil potensielt kunne *forskyves*. Meninger med arbeidet og profesjonelle identiteter som før var gyldige kan bli ugyldiggjort.

Selv om den teknologiske utviklingen har vært enorm de siste 200 årene, foregår de kognitive endringene i språk og evalueringsrepertoar ganske langsomt. Ved å studere og klassifisere idealene som argumenter og legitimeringer knyttes til på et eget nivå, kan vi danne oss et bilde av samfunns- og organisasjonsendring i skjæringspunktet mellom aktør, kultur og teknologi, hvor aktørene selv åpenbart bidrar til å forme den verdenen de er en del av gjennom *virkelighetstester*, *kritikker* og *gyldiggjøring*. Denne typen analyser bryter med teknologisk determinisme.

5.1 Symbolsk grenseanalyse

Det er imidlertid ikke gitt hvordan forskeren kan observere *legitimering* i komplekse organisasjonskulturer. Jeg har i mitt opplegg tatt utgangspunkt i informantenes grensetrekkinger i intervjuene. Spesielt når informantene ble bedt om å definere for eksempel god programvareutvikling eller en god kollega, ble svarene som oftest begrunnet. Det å skille mellom godt og dårlig utgjør grunnlaget for hva vi forstår som den menneskelige *kritiske kompetanse*. Alle kvalitative forskningsopplegg forutsetter at informantene besitter en kompetanse til å forstå og

formidle fortellinger fra deres egen virkelighet, men i en symbolsk grenseanalyse er det aktørenes kritiske sans og grensedragning mellom det gode og det dårlige som driver analysen fram. Boltanski og Thévenots rammeverk er et klassifiseringsverktøy basert på analyse av symbolske og materielle grenser. Boltanski og Thévenot har imidlertid ikke utviklet noen spesifikk intervjumetodikk, og i utviklingen av intervjuguiden min og i analysen av intervjuene benytter jeg delvis framgangsmåter som er i tråd med Lamont (1992).

Symbolske grenser forstås av Lamont som konseptuelle distinksjoner som mennesker bruker for å kategorisere objekter, folk, praksiser, tid og sted (Lamont 1992). Det å kunne trekke en symbolsk grense forutsetter både *inkludering* av ønskelige egenskaper og *ekskludering* av det ikke-ønskelige (i tidlige forskningstradisjoner ble det ikke-ønskelige ofte omtalt som det urene) (Durkheim 1965; Simmel 1950). Symbolske grenser trekkes altså mellom *hva, hvem* og *hvilke egenskaper* man liker og ikke liker. Distinksjonene mellom hva som er godt og dårlig blir uttrykt gjennom normative interdiksjoner (tabuer), kulturelle holdninger og praksiser, og gjennom mønster av preferanser (Lamont 2001). Gjennom å dele konseptuelle systemer, følelser og forståelser for hva som regnes som høy- og lavverdig med andre, skaper mennesker en felles forhandlingsarena over hva som er viktig og hva som må gjøres i et fellesskap. Å skape og utvikle symbolske grenser i samhandling er dermed *en* måte for å utvikle og ivareta et gruppemedlemsskap (Epstein 1992). Symbolske grenser er en forutsetning for kategoriseringen av mennesker i klasser, grupper, kjønn og raser. Grensetrekkinger brukes også for å legitimere *ulikhet* – de er en forutsetning for at noen individer eller grupper, som typisk assosieres med høy verdighet eller spesielt gode egenskaper, får sosiale fordeler eller flere ressurser enn andre i et større fellesskap (Weber 1978). Bare når det hersker bred enighet om hva de symbolske grensene er i en gruppe, kan de anta en sosialt begrensende form og bli sosiale grenser. Sosiale grenser er objektifiserte former av sosial ulikhet, som er reelle i den forstand at de gir ulik tilgang til ressurser (materielle og immaterielle) og muligheter (Lamont og Molnár 2002). Symbolsk grensedragning har altså en konstitusjonell rolle i utviklingen av organisasjonspraksiser, rutiner og materielle inndelinger. Klassifikasjonssystemene kan utvikle seg til å bli *naturaliserte*, sånn at ingen lenger stiller spørsmålstegn ved dem. Det at kvinnelige programvareutviklere ofte blir tilbudt hybride stillinger (Guerrier mfl. 2009) kan være nettopp et sånt eksempel på et naturalisert klassifikasjonssystem. Kausalt sett kan symbolske grenser altså forstås for å være en av flere nødvendige forutsetninger for konstruksjonen av av sosial ulikhet (Lamont 1992). Lamont mener, akkurat som Boltanski og Thévenot, at det ikke kan være direkte sammenheng mellom klasse og disposisjoner. Hun forutsetter i stedet at kritiske distinksjoner blir konstruert gjennom at aktører bruker og utvikler kulturelle repertoar. Selv om sosioøkonomisk status ofte

korresponderer med holdninger, kan man ikke vite dette på forhånd, og hun mener derfor at det blir feil av forskeren å kategorisere aktører på forhånd etter f.eks. klasse/stilling.

Lamont tar utgangspunkt i nettopp det at symbolske grenser ikke er statiske eller permanente, og viser dette gjennom flere analyser av kulturelle og nasjonale variasjoner (Lamont 1992, 2000; Lamont og Thévenot 2000). I *Money, morals and manners* (1992) ser Lamont spesielt på *moralske*, *sosio-økonomiske* og *kulturelle* grensetrekkinger i den amerikanske og den franske middelklassen. Lamont finner at den amerikanske middelklassen i motsetning til den franske oftere brukte moralske og sosioøkonomiske grensetrekkinger. Hun konkluderer med at dette skyldes at intellektuelle i USA har en mindre framtredende rolle i samfunnet, at skolevesenet og media er desentralisert og at velferdsstaten er mindre utbredt. Dermed bruker aktørene i større grad markedsbaserte evalueringer når de vurderer andre. I *The Dignity of Working Men* (2000) finner Lamont blant annet at den hvite arbeiderklassen i USA og Frankrike deler oppfatninger om arbeidsmoral, men sistnevnte har en mye større klassesolidaritet med fattige og svarte.

5.1.1 Grenser i forhold til profesjon og identitet

Definisjonen av profesjon har vært gjenstand for mange diskusjoner gjennom tidene. De tidligste studiene så på profesjonalisering som en internalisert normativ verdi hos en utøver, med en stabiliserende funksjon i større sosiale system (Parsons 1951; Durkheim 1992). Noe senere forsøkte man i stedet å definere profesjon ut fra generelle trekk som skilte profesjoner fra andre yrker (Etzioni 1969; Pavalko 1971). På 70- og 80-tallet fulgte en periode med en mer konfliktorientert tilnærming til profesjonsbegrepet, og her var forskningsobjektet i større grad ideologiproduksjon, dominans og ekskludering gjennom grensetrekkinger. Profesjon ble ikke definert ut fra funksjon eller kunnskapstype, men ut fra hvordan den profesjonelle organiseringen ga utøverne institusjonell kontroll over tilgang, opplæring og evaluering av nye medlemmer. Kunnskap og ferdigheter ble gjort til knappe goder som ga økonomiske gevinster og prestisje. Yrker som klarte å forvandle sin særegne kompetanse til et knapt gode, kunne posisjonere seg og kjempe om monopol (Larson 1977). Parkin mener at monopoliseringen av retten til å utøve kunnskap som sosial grensetrekking, er et hovedtrekk ved moderne profesjoner (Parkin 2001).

Etter hvert ble problemstillinger forbundet med sosial interaksjonisme studert. Viktige spørsmål har vært hvordan eksperter og lekmenn forstår og forhandler om hverandres kunnskap og roller, og hvordan de tenker om kunnskap og praksis. Abbot (1988) gjorde en komparativ og historisk studie av profesjonenes framvekst i England, Frankrike og USA. Boka bidro blant annet til å endre fokus fra skillene mellom profesjonsinnehavere og outsidere, til de forhandlingene (og

av og til kampene) som foregår innen og mellom profesjoner. Abbot tok utgangspunkt i at profesjonene konkurrerte seg mellom, gjennom å forhandle om *juridiske* grenser og gyldigheten av de ulike typene *profesjonell kunnskap* i samfunnet. Grensetrekkingene kunne også forklare hvordan profesjonelt "urent" arbeid ble skilt ut til egne profesjonsmedlemmer, og i noen tilfeller til andre profesjoner (Abbot 1988:125). Abbots utgangspunkt var at privilegerte grupper ikke oppstår i et vakuum, men at de blir utviklet i samspill med andre institusjoner i samfunnet. De profesjonelle praksisene er knyttet til verdier og normer i samfunnet, og må derfor forstås i en større kulturell sammenheng. Abbott fremhever at grensetrekking er den eneste måten å etablere og utvikle en profesjonell legitimitet som kan få allmenn aksept i et samfunn eller i en kultur. For å utvikle profesjonelle privilegier må det dermed finnes et organisasjonelt "andre" som profesjonen kan utvikles i mot (Abbott 1995). Larson (1977) mener at de strategiene profesjonelle bruker for å definere og institusjonalisere grensene mot lekmenn, utgjør essensen i "profesjonaliseringsprosjektet". En nøkkeloppgave for symbolsk grenseteori ift. profesjoner er dermed å påpeke og beskrive de sosiale strategiene som privilegerte profesjonelle konstruerer for å *skille seg fra* de andre (Abbott 1995).

I tråd med dette har Vallas (2001) sett nærmere på hvordan disse distinksjonene mellom profesjonelle ingeniører og maskinarbeidere utspiller seg i papirindustrien. Han finner også at grenser utvikler seg og forhandles om, mellom profesjonelle og andre ansatte på arbeidsplassen. Spesielt kulturelle grenser i form av vitenskapelig og teknisk kunnskap blir regnet som gyldige for å legitimere sosiale grenser på arbeidsplassen. Vallas finner at symbolsk grensetrekking er en omstridt prosess innad i organisasjonene, og at utfallet av lokale profesjonskonflikter ofte er kontekstavhengig. For å legitimere en kvalitativ forskjell på profesjonell og lek-kunnskap, er det ikke nok å vise til utdanningstitler og diplom. Den profesjonelle kunnskapen, om hvordan man for eksempel bruker et verktøy, kan i betydelig grad overlappe med lekfolks kunnskap på området. Derfor må de profesjonelle også trekke grenser for kunnskapen sin i uformelle prosesser, som kan styrke og bekrefte nødvendigheten av etablerte privilegier. Spesielt viktig blir dette i turbulente organisasjonskontekster med hyppige endringer (Vallas 2001). De *uformelle* grensedragningene blir viktige når det skal skapes nye stillinger og ulike karrierestiger i organisasjonene. Grensetrekkinger beskriver ikke bare egenskaper informantene distanserer seg fra, men også idealer og egenskaper informantene strekker seg mot eller identifiserer seg med. Mens Abbot og Vallas er mest opptatt av grensetrekkinger som lukker de andre ute fra profesjonen, og definerer identitet ut fra eksklusivitet, har andre grenseteoretikere i stedet vist at det finnes ulike *nivåer* av eksklusjon, toleranse og kulturell bredde (Lamont 1992; Bryson 1996; Erickson 1996). Aktører bruker symbolsk grensetrekking for å inkludere seg selv i kulturelle fellesskap, og symbolske

grenser styrer også deres oppfatning av forpliktelse i forhold til andre grupper (Lamont 2000).

I mitt utvalg bruker informantene symbolske grenser – ikke bare for å håndheve, vedlikeholde, normalisere eller rasjonalisere sosiale grenser i en organisasjon, men for å *posisjonere* seg selv og sine meninger i tråd med hva man anser som *verdig*, og *fremme kritikker* mot andres posisjon og definisjoner av verdigheter. Symbolske grenser brukes også for å *bestride* og *rekontekstualisere* meningen med sosiale grenser (Lamont og Molnár 2002:186), for å *forstå og kategorisere* seg selv og andre, *skille* mellom god og dårlig kunnskap og godt og dårlig arbeid, *forhandle* om hva som er viktig i programvarefaget, og *legitimere ulike privilegier* mellom grupper av arbeidere i organisasjonene. Kombinert med referanse til verdiverdener spiller grensetrekkinger en stor rolle for å legitimere både *gruppemedlemsskap* og ulike *praksiser* i organisasjonene.

5.2 Mitt metodiske opplegg

Flere profesjonsteoretikere har påpekt at utviklingen av en profesjon ikke bør sees bare isolert eller stegvis men i sammenheng med utviklingstrekk i samfunnet for øvrig (Abbott 1988; Freidson 1982). I dag er det vanlig å ikke lenger skille mellom profesjon og andre yrker, men å se fagretningene som sosiale former som deler mange karakteristikker (Svensson og Evetts 2010). I denne avhandlingen er jeg ikke opptatt av å avgjøre om programvareutvikling er en profesjon eller fagretning.

Mitt prosjekt utforsker hvordan aktørene selv bidrar til å uttrykke en verdig profesjonell identitet. Samtidig forsøker jeg å ta hensyn til hvordan eksisterende strukturer i organisasjonene legger noen føringer for denne konstruksjonen, hvordan det tilgjengelige evalueringsrepertoaret legger føringer for hvilken vekt ulike eksisterende strukturer skal tillegges, og dermed hvilke mulighet aktører har til å endre gamle strukturer og skape nye. Grensetrekkinger formuleres alltid i tråd med det evalueringsrepertoaret som er tilgjengelig i et samfunn. Selv om en fagretning eller profesjonsutøvere deler interne idealer og har egne verdier, bruker utøverne i stor grad de samme kategoriene, forståelsene og språket som finnes i resten av samfunnet. Profesjonelle identiteter kan dermed analyseres i lys av Boltanski og Thévenots verdiverdener – og videre – forstås som mulig å reprodusere ved hjelp av rettferdighetstester og kritikker. I tråd med Evetts, som mener at profesjonsforskningen bør være mindre opptatt av å definere hva en profesjon er, forsøker jeg heller å forklare hvordan aktørene bidrar til å opprettholde, reprodusere og transformere den. Med dette rammeverket kan programvareutvikleryrket analyseres som en profesjon eller fagretning i utvikling bestående av flere, interne, konkurrerende, lokale logikker eller verdigheter – som hver for seg er muliggjørende og

begrensende i organisasjonssammenheng – men som også er del av et større samfunn.

Ved å se på arbeidsmoral og arbeidsidentitet på dette nivået tar jeg utgangspunkt i at den faglige eller profesjonelle identiteten ikke er noe som følger direkte av klasse, stillinger, rolle, subkulturer, utdanning-, profesjon eller tittel, men noe som heller er gjenstand for posisjonering, kritikk og tester av verdighet og stolthet over arbeidet blant pragmatiske aktører, og som over lengre tidsrom kan bidra til å endre fagkarakter- eller trekk. Det er imidlertid viktig å understreke at dette ikke er en grunn til å la være å utforske informantenes bakgrunn, tvert imot. Ved å inkludere biografisk informasjon i analysen kan jeg vise at de ulike idealene ikke er tatt fra løse luften, men at de er knyttet til erfaringer informantene har gjort seg, og til virkelige relasjoner og strukturer i bedriftene.

5.2.1 Personlig og kulturell identitetsformasjon

Det er verdt å merke seg at lønnsmottakerrelasjonen i liten grad påvirker profesjonelle eller faglige idealer i en organisasjonskultur. Det kan godt følge en stolthet med det å ha et fast lønnsarbeid, men lønnsrelasjonen utgjør i seg selv ingen distinksjon til alle de andre som også har lønnsarbeid. Lønn, eller aktørenes sosioøkonomiske status, er i dagens samfunn ikke noe som kan brukes for avgjøre faglige konflikter, disputter, og penger kan ikke brukes direkte for å gi komplimenter mellom kolleger som regner seg som ekvivalente. Man kan se for seg den håpløse situasjonen hvor en akademiker takker en kollega offentlig gjennom å si at vedkommende har betydd mye, og så gi – godt synlig for alle tilstedeværende – gi en bunke tusenlapper for å uttrykke dette. Situasjonen vil antakelig oppleves som pinlig for alle som observerer, men spesielt den intenderte mottakeren vil antakelig føle at den akademiske anseelsen og verdigheten bli truet.

Tradisjonelt sett har profesjonell identitet blitt forstått som personlig motivert, men utspilt i et kollegialt fellesskap. Jeg skiller mellom tre typer personlige, identitetsformende faktorer – indre dedikasjon, internalisert språk/verdi/norm-system og konkurranse. I tillegg ser jeg den profesjonelle identiteten som et kulturelt produkt. Begge nivåene eksisterer i samspill med hverandre, og legger muligheter og begrensninger for hverandre.

Bernstein beskrev *indre dedikasjon*, spesielt i spesialiserte, profesjonelle eller akademiske yrker, som en nøkkelrolle i menneskers og gruppers identitetsformasjon (Beck og Young 2005; Bernstein 2000). Man kan forstå denne som en lidenskapelig motivasjon, men samtidig var den noe mer. Ifølge Bernstein definerte de tidligste profesjonene (legeyrket, advokatyrket, og filosofene) selv sine faglige verdier, idealer og problemer. Bernstein hevdet at disse første verdiene ble dannet av personenes karakterer eller karismaer. De første profesjonsutøverne var dermed fri fra

profesjonell sosialisering, og de valgte selv å fokusere på det de gjorde, ut fra det Bernstein kaller *inwardness*. Han var opptatt av hvorvidt en *audit culture* (Power 1997; Strathern 2000) med ekstern målsetning, tilsyn og styring truet en profesjonell identitetsformasjon basert på indre dedikasjon. Bernstein lagde et eget teoretisk rammeverk for å analysere profesjonell identitetsformasjon i utdanningsverket, som jeg ikke går nærmere inn på her, men jeg vil beholde hans idé om inwardness som kilde til identitetsformasjon som kommer fra aktøren selv, og som noe som kan bli begrenset av ytre påvirkning.

En profesjonell identitet kan imidlertid også forstås som basert på internaliserte, tillærte koder, symboler og språk, formidlet gjennom sertifiseringer, kursing, universitet og kollegiale fellesskap, og dokumentert gjennom diplomer og karakterer. Dette er et tradisjonelt syn på hvordan en profesjonsmoral utvikles, formidles og deles. Begrepet *profesjonsmoral* brukes for å forklare hvordan internaliserte normer og verdier gjør at profesjonelle kan opptre på en annen måte enn aktører styrte av økonomiske insentiv eller hierarkiske kommandolinjer, og som også utgjør basisen i en kollegial organisasjonsform som er ulik både markedsorganisering og byråraktisk organisering (Grimen 2008).

En tredje identitetsformende faktor ligger i det kulturelle aspektet som er forbundet med gaveøkonomi, først beskrevet av Mauss i forhold til gavepraksisen i arkaiske stammesamfunn hvor det i mange sosiale sammenhenger var vanlig å gi mat eller ytelser (Mauss og Cunnison 1970). Den som kunne bidra med mest til fellesskapet hadde størst respekt og makt. Ideen i *Gaven* er at en giver ikke separeres fullstendig fra gaven som blir gitt. Offeret som en ytelse innebærer skaper en verdighet (kalt *hau*), som styrker giverens verdi. Gavemetaforen er brukt for å kontrastere produksjonen av fri programvare fra produsenteid programvare (Raymond 2001). Det å *gi* er bedre enn å skylde, så gjennom å yte blir man en bedre person, og man øker både sin egen verdighet og kollektivet man er en del av sin verdighet. I denne kulturen er det bare de virkelige store personer har råd til å gi alt, ofre alt, både økonomisk og moralsk (Strathern 1988; Zeitlyn 2003). En prestasjon er knyttet til verdighet. Når øybeboerne som beskrives av Mauss ikke klarte å leve opp til forventningene, medførte det skam og tap av ære. Når utviklere med forpliktelsebasert motivasjon ikke klarer det målet man har satt seg, eller at de ikke klare å leve opp til kollektivets idealer, kan de på samme måte oppleve skam og tap av motivasjon. Denne kulturelle forståelsen kan også brukes innad i organisasjoner for å forstå identitetsutvikling – ved å kunne prestere og yte får man en positiv verdighet og stolthet, og hvis man – uansett årsak – forhindres i å prestere, risikerer man også å tape denne.

Disse tre identitetsformende faktorene eksisterer i et samspill med det tilgjengelige evalueringsrepertoaret i en organisasjonskultur. Viktigheten som tillegges indre motivasjon er

ikke gitt, men et resultat av kritikker, forhandlinger og virkelighetstester i organisasjonene. Selv om disse tre motivasjonene utspiller seg i handlingsregimer hvor refleksivitet og ekvivalens spiller liten rolle, kan tapet av verdighet uttrykkes og i noen grad bli forsøkt kompensert gjennom legitimering. Og selv om verdigheten som ligger i indre dedikasjon ofte kan brukes i en faglig disputt, er det ikke sånn at profesjonell karisma kan brukes på kryss og tvers av fagområder – de profesjonelle grensene vil alltid kunne kritiseres og dermed kreve legitimering. Det er heller ikke gitt at høye prestasjoner gir høy verdighet innenfor en profesjon – høye prestasjoner kan også kritiseres, noe blant annet denne sommerens søkelys på legers overtidsbruk er et eksempel på (Arbeidstilsynet 2011). De non-refleksive, indre motivasjonene for profesjonelt arbeid, må dermed kontekstualiseres og forstås i tråd med legitimeringer, og er begrenset og muliggjort av gyldiggjøring.

5.2.2 *Meningen med arbeid* som reprodusert gjennom parallelle tester

Min idé er å analysere hvordan ulike meninger med arbeidet reproduseres gjennom *tester* og *kritikk*. Kritikken er formulert og gyldiggjort av aktører, men i tråd med oppfatninger av hva som er godt og hva som er virkelig. Endringer i organisasjonen, både langsomme og mer krisepregede, kan endre testenes validitet og bidra til at de – og dermed det vi kan forstå som *meningen* med arbeidet – er nødt til å hekte seg på nye forståelser for at de kan la seg reprodusere. Mening, som blir gyldiggjort av flere, kan gi grunnlag for verdighet og identitet. Her bruker jeg en todelt forståelse av faglig/profesjonell identitet, som skissert i kapitlet ovenfor. Den (1) interne forankringen til et fag er mindre refleksiv og non-ekvivalent og kommer fra (a) en personlig lyst til å gjøre noe godt eller meningsfylt, (b) tillærte koder, verdier og normer og (c) personlige prestasjoner (konkurranse). Den (2) eksterne forankringen krever refleksivitet og ekvivalens, og kommer fra reproduksjon av parallelle tester, som tåler eller transformeres, av kritikk. Denne kan analyses ved å studere verdiverdener og logikker som regnes som gyldige innad i et fag, men som også gjenspeiler det øvrige samfunnets evalueringsrepertoar i stor grad.

For å kunne eksistere og forandres over tid, må en profesjonell identitet være mulig å reprodusere gjennom praksis, men også gjennom språk og samhandling både med kolleger og *andre deltakere* i samfunnet, hvor også strukturer og føringer i organisasjon og samfunn inngår.

Den korte forklaringen

Den profesjonelle eller faglige identiteten kan i dette perspektivet analyseres som reprodusert av tester av ulike variabler representert gjennom et mangfold av idealer. Det er bare *aktører* som kan gyldiggjøre legitimeringer og kritikker med referanse til oppfatninger om god, rettferdig

faglig praksis, organisasjonen og samfunnet utenfor. Hvilke idealer og forståelser som kan brukes for å reprodusere faglig identitet, vil variere i ulike situasjoner og under ulike betingelser.

Den lange forklaringen

I et komplekst sosialt system, som en organisasjon eller et nettverk av organisasjoner, vil det aldri være én måte å reprodusere en faglig verdighet på. Kollegiale fellesskap spiller en viktig rolle for å gyldiggjøre hvilke idealer som skal regnes som viktig, men det samme gjør kunder, samarbeidspartnere og føringer i organisasjoner og samfunn. Innenfor faget vil ulike meninger og forståelser av hva som er rettferdig og verdig gjelde, og disse kan raffineres og i ulik grad brukes som utgangspunkt for å danne "lokale" faglige verdigheter, med sitt eget sett av gyldige idealer, rangeringer og oppfatninger av rettferdighet og investering. I stor grad eksisterer disse lokale variantene uavhengig av roller eller stillingsbeskrivelser. Den subjektive oppfatningen av verdighet *kan* være knyttet til en rolle eller stilling, men også til en gitt kompetanse eller kunnskap på et felt eller til spesifikke objekter (maskiner, programmer, metoder) eller til personlige egenskaper som at man for eksempel er initiativrik, oppfinnsom eller sosial. Tilknytningen er ikke gitt, gyldigheten må kunne legitimeres når noen setter spørsmålstegn ved den. Hvis et subjekt har høy verdighet i tilknytning til faget, gjennom indre dedikasjon eller andre styrker, kan dette rettferdiggjøre privilegier. Det er imidlertid ikke gitt at privilegier vil bli oppfattet som rettferdig over tid. Det går an å kritisere *gyldigheten* av alt det en verdighet inneholder: Privilegier, personlige egenskaper, ulike former for kunnskap og måter å investere i arbeidet på. Etter hvert som organisasjoner og verden forandrer seg, vil aktørene nødvendigvis lansere nye kritikker og virkelighetstester, som forskyver gamle idealer og oppfatninger i tråd med hva som oppfattes som virkelig og rettferdig. De tidligere måtene å reprodusere faglig identitet på blir ugyldiggjort, og nye måter blir gyldiggjorte.

5.2.3 Eksempler

Med denne framgangsmåten kan jeg si noe om hvilke idealer som faglige/profesjonelle identiteter er dannet av, og hvordan aktørene bruker interne logikker for å definere den større meningen i en profesjon eller et fag. Jeg vil nå vise reproduksjon av faglig identitet i tre eksempler. Det første er lånt fra Sennett, de to andre er hentet fra mitt eget innsamlede materiale. I det første eksemplet lar ikke den faglige stoltheten eller meningen med arbeidet seg lenger reprodusere. I det andre eksemplet fungerer grensedragninger mot "de andre" som en suksessfylt måte å reprodusere faglig verdighet eller mening på. I det tredje eksemplet antyder jeg hvordan en verdighet som en gang var sterke innad faget, nå fremstår som svekket gjennom

tester og kritikk. Eksempelet blir grundigere gjennomgått i analysen *Den gode utvikleren*.

Eksempel 1: Sennetts baker

I *The corrosion of character* (1998) har Sennett et eksempel fra et bakeri. Bakeren i dette eksemplet har en faglig stolthet eller verdighet som lar seg reprodusere gjennom at han har den nødvendige kompetansen, kjenner de nødvendige teknikkene og behersker kunsten å jobbe effektivt. Denne verdigheten, er dels faglig og dels personlig, og dessuten bare mulig å reprodusere gjennom at kundene faktisk kjøper – og helst sier at de liker brødene. Krisen oppstår da bakerovnen blir skiftet ut med en helautomatisk ovn. Alle testene som reproduserer den interne, personlige faglige identiteten blir ugyldiggjort. Hvis bakeren, eller nå: bakemaskin-teknikeren, skulle kunnet utvikle en ny, personlig stolthet i arbeidet, måtte denne i så fall skapes i tråd med de objektene (maskinene) som finnes i det nye, helautomatiserte bakeriet. Men dette er umulig, fordi alle kan lære seg å betjene den automatiske maskinen. Det finnes ingen *andre* som profesjonelle grenser kan trekkes mot. Dette er bakerens identitetskrise. Kundene aner imidlertid ikke forskjellen, fordi brødene smaker like godt.

Eksempel 2: Fra empirien – lønsjen

Så hvordan ser en sånn test basert på symbolsk grensedragning ut? Programvareutviklere lever i en virkelighet hvor utskiftninger av kodespråk, teknikker, biblioteker, og maskinvare foregår kontinuerlig. Deres oppfatning av arbeidets mening må tåle disse utskiftningene hvis de skal kunne være i jobben over lengre perioder. Dette betyr at det i utgangspunktet er utfordrende å ha sin profesjonelle stolthet knyttet til spesielle kodespråk, spesielle teknikker, dingser eller maskiner – alle disse har relativt kort levetid. Utviklernes profesjonelle stolthet må derfor finne næring fra annet hold, og her spiller legitimeringer en stor rolle.

Jeg skal helt kort gi en generell fortelling som inneholder kulturelle grensetrekkinger – uten å knytte denne til spesielle verdiverdener.[20] I Beta spiste jeg lønsj med tre andre utviklere, i dette tilfellet var alle relativt unge. Den ene hadde jeg intervjuet tidligere. Rundt bordet ble det snakk om hva en god utviklerkultur egentlig var, og hvordan Beta skilte seg fra mange andre arbeidsgivere de visste om. De trakk fram spesielt en arbeidsgiver som to hadde jobbet for, og som alle rundt bordet hadde hørt om. Denne arbeidsgiveren var en stor bank, som ifølge dem var beryktet for å ha en spesielt dårlig utviklerkultur. Ting som ble trukket fram som spesielt negative var at både lederne og programmererne "holdt kortene tett til brystet" og ikke var interesserte i samarbeid. Ledelsen var generelt lite opptatt av utvikling, og de ansatte folk "helt vilkårlig".

20 Følgende informasjon er basert på notater fra bedriftsbesøket, ikke opptak.

"Ingen der brydde seg om kvalitet". Alt dette er grensetrekkinger, som blir uttalt for å skille god kultur fra dårlig. I fortellingene beskrives de strukturelle føringene som negative for utviklingsmiljøet. Når utsagnene får stå uten å bli kritisert, kan de gjentas og forsterkes. I dette konkrete eksemplet ble testen stående uimotsagt – og den foreslår at mine bordpartnere definerte generøsitet, åpenhet, kompetente og interesserte ledere med kvalitetsfokus som elementer som skilte deres nåværende jobbsituasjon med situasjonen hos "de andre". Denne meningen, som viser noen konkrete idealers fordel kontrastert til noe dårlig, ga mine lunsjpartnere næring til en identitetsdannelse: Beta er en god og spesiell bedrift, som representerer noe eget. Eksempelet de brukte kunne alle forstå, også ikke-profesjonelle som bare var innom for en kort periode, slike som meg. Slike grensetrekkinger, som får stå uimotsagt, bidrar til å reprodusere en stabil verdighet over tid. Denne typen verdighet kan imidlertid også utfordres fra subjekter og objekter eksternt for fellesskapet, for eksempel hvis en organisasjon må slankes i en økonomisk krise, hvis kolleger plutselig blir urettferdig behandlet, eller hvis det kommer til en ny leder, som kanskje er "uinteressert", eller kanskje for interessert – i å endre måten arbeidet skal organiseres på.

Eksempel 3: Fra empirien – forskyvning

I denne avhandlingen finner jeg at forestillingen om at en god IKT-arbeider kan være drevet av indre motivasjoner som lek, lidenskap og prestasjon (Levy 1984; Himanen 2001; Castells og Himanen 2002; Graham 2004), blir sterkt kritisert i både organisasjonene jeg besøkte (analyse 1) og i metodelitteraturen (analyse 2). Jeg beskriver dette som en forskyvning fremmet gjennom virkelighetstester og kritikk. De personlige kvalitetene, som muliggjorde det å investere mye tid, personlig prestisje og ressurser i arbeidet, ble regnet som viktige i programvarefagets historie. I dagens IKT-bedrifter og metodelitteraturen settes det spørsmålstegn ved gyldigheten til disse idealene. Dagens virkelighetstester er knyttet til en virkelighet hvor marked, profitt, kunder og balansen mellom arbeid/fritid blir tatt med i formelen. Verdigheten som ble forbundet til det å ha et lidenskapelig forhold til faget blir kritisert fra så mange hold at de ikke lenger lar seg reprodusere i organisasjonssammenheng. Denne måten å investere (tid i arbeidet) på, "å gi alt", gir ikke lenger verdighet, men blir heller noe som blir ansett som lavverdig.

6. Metode og utvalg

Avhandlingen består av to analyser. *Analyse #1 – den gode programvareutvikleren* (side 87) er basert på 23 intervjuer samlet inn i norske programvareutvikler-organisasjoner i 2008-2009. *Analyse #2 – den ukomplekse verdiverdenen* (side 135) består av en analyse av 12 håndbøker i smidig utvikling og prosjektstyring. I analysene har jeg forsøkt å kombinere et synkront komparativt perspektiv (jeg sammenligner ulike fagverdigheter i norske organisasjoner) med et diakront komparativt perspektiv (jeg beskriver hvordan og hvorfor den ukomplekse fagverdigheten har vokst fram). I forhold til begge kriteriene har analysene begrensninger. Selv om informantene har mange ulike nasjonaliteter, består utvalget hovedsakelig av nordmenn, og bare 5 kvinner – alle norske. Samtlige organisasjoner er lokalisert i Norge, selv om noen hadde hovedkontor i andre land. Analyse 1 baserer seg dermed på en sammenligning av samtidige kulturer og profesjoner slik disse fremstår i norske organisasjoner i nåtida. Jeg sammenligner og beskriver fagverdighetene som jeg fant på tvers i disse organisasjonene, og i svært liten grad forsøker jeg å beskrive forskjeller på de ulike organisasjonene. I større grad trekker jeg fram noen biografiske forskjeller på enkelte av informantene mine. Samtidig inneholder analysene en begrenset diakron tilnærming ved at informantene sammenligner dagens situasjon med tidligere jobberfaringer de siste ti årene, og et tilfelle helt tilbake til 80-tallet.

11 av håndbøkene i utvalget for analyse #2 er på sin side USA-amerikanske, og en er fra Hong Kong. Tilsammen inneholder de eksempler og innhold som referer til nasjoner, byer og organisasjoner som ligger i både USA og Europa og i litt mindre grad Asia. Det er disse metodologiene, med sitt språk og sine normer og verdier, som arbeidere i utvalget mitt viser til når de snakker om smidig metodikk. Et diakront perspektiv er forsøkt ivaretatt gjennom at jeg analyserer håndbøkene som *kritikker* av (1) tidligere management-håndbøker, og (2) problemer med IKT- og kunnskapsarbeid, og gjennom at jeg (3) vurderer alle disse i lys av verdiverdener funnet i management-litteratur slik denne blir fremstilt i Boltanski og Thévenot og Boltanski og Chiapello (2005). Det diakrone perspektivet er imidlertid også her begrenset gjennom at hele håndbokutvalget jeg analyserer er publisert i 2000-2010, og gjennom at sammenligningen med management-litteratur på 60 og 90-tallet er annenhånds.

6.1 Symbolsk grenseanalyse – den gode utvikleren

6.1.1 Utvalg – strategi og forventninger

Datainnsamlingen er basert på et strategisk utvalg organisasjoner og personer. Organisasjonene ble valgt fordi de representerte et bredt utvalg av norske programvarebedrifter og fordi jeg vurderte dem som stabile. Kriteriene for organisasjonene var at de skulle tilhøre ulike bransjer, ha ulik størrelse (store/små) og ikke være plassert i samme region. Jeg valgte også bort bedrifter som utvikler såkalte mobilapplikasjoner, webdesign eller driver med konsulenttjenester. Næringer eller fagområder jeg har besøkt er offshore, mediebransjen, og applikasjonsutvikling (programvare for PC/Mac/etc.), "hobbybasert" fri programvare-utvikling og internasjonal fri programvareutvikling. 23 intervjuer ble utført iløpet av høsten 2008 til og med høsten 2009. Kriteriene for informantene var at de skulle ha ulik alder, være av begge kjønn, gjerne ha ulik utdanning men fylle omtrent den samme stillingskategorien – programvareutvikler. I tillegg ønsket jeg å utføre noen intervjuer med ledere og personer som jobbet i støttestillinger for programvareutviklerne. Bakgrunnen for dette utvalget var at jeg nettopp ville se nærmere på et bredt utvalg high skill-utviklere – personene som Kelley (1985) kalte gold-collar workers. Dette betyr imidlertid ikke at alle i utvalget mitt betrakter seg selv som suksessfulle, eller formidler at de er lykkelige. Noen problemer som kom opp under intervjuene er tidsklemma, at man føler at man ikke klarer å yte nok, og at man har mistet mye av inspirasjonen og friheten som man hadde tidligere i karrieren.

Analysens studieobjekt er dermed først og fremst *uttalelser* fra IKT-arbeidere som for det første beskriver profesjonell eller faglig arbeidspraksis,[21] og som for det andre beskriver verdier, rangeringer og tankesett som gis legitimitet gjennom at de vurderes i relasjon til nåtidig eller tidligere arbeidserfaringer. Kontekstuelle data om faget, arbeidspraksis og biografiske data om informantene ble også innsamlet. En informant, Fridtjof, arbeidet for tiden ikke som utvikler, men ble likevel intervjuet pga. hans tilknytning til et norsk programvaremiljø han tidligere var del av. Jeg intervjuet alle kvinnene jeg hadde tilgang til i bedriftene jeg besøkte, det vil si kvinner med mastergrad i informatikk eller som jobbet eller hadde jobbet med utvikling – og dette var altså fem stykker. Som i mange andre mannsdominerte yrker, er fulltid den dominerende tilknytningsformen. Jeg hadde ikke hatt større tilgang til å intervjue f.eks. kvinner om jeg hadde inkludert deltidsarbeidere. Arbeidet er for noen organisert i arbeidsgrupper eller par, andre jobbet alene. Noen hadde liten oppfølging fra ledere eller managere, og fikk tilbakemeldingene

21 Med arbeidspraksis forstår jeg her både *arbeidsutførelse*, *arbeidsetikk* og *relasjoner* til kolleger og maskiner (ting).

sine fra kunder og lignende, mens andre mente at de hadde mer oppfølging. I praksis ble intervjuene gjennomført semistrukturert. Jeg forsøkte å følge strukturen i intervjuguiden noenlunde, men oppfordret samtidig informantene til å avbryte hvis de kom på noe de mente var viktig, og ellers forsøke å knytte svarene til assosiasjoner og fortellinger med utgangspunkt i egne erfaringer og tanker.

6.1.2 Organisasjonene

Mange av de største, rene programvareutviklerselskapene i Norge er geografisk konsentrert rundt Oslo. Begge de største firmaene (Beta og Charlie) ligger i Oslo-regionen. Det er også i denne regionen de aller fleste konsulentvirksomhetene ligger. Siden mange bransjer har egne programvareutvikleravdelinger finnes det også arbeidsplasser mange andre steder, spesielt rundt kystbyene (olje og offshore). Små firma som driver med spill, grafikk og medierelatert virksomhet finnes også i distriktene, og i de siste årene har det dukket opp mange små firma som driver med applikasjonsutvikling eller småspill for mobilmarkedet. Bedriftene Alfa og Delta ligger utenfor Oslo.

Pseudonym	*Lokasjon HK*	*Type*	*Marked*	*Størrelse R&D-avd.*
Alfa	Distrikt	Produsenteid	Offshore	30+
Beta	Oslo	Produsenteid	IKT/bedrift-privat	200+
Charlie	Oslo	Fri programvare	IKT/bedrift-privat	80+
Delta	Distrikt	Produsenteid	Mediebransjen	30+
Echo	Utlandet	Fri programvare	IKT/bedrift-privat	10000+
Foxtrot	Distrikt	Fri programvare	IKT/bedrift-privat	1

Tabell 3: Bedriftsprofiler

Alfa

Alfa utvikler offshore-teknologi, og har opplevd en rask vekst siden oppstarten på tidlig 2000-tall. På intervjutidspunktet i 2008 var bedriften en internasjonal aktør med over 200 ansatte, men selve utviklingsavdelingen av kjerneteknologien var lokalisert i Norge, og besto av færre enn 30 personer. Alfa utvikler produsenteid programvare, og de benytter Scrum, en smidig utviklingsmetodologi, for å styre prosjektene sine. Jeg var på besøk i Alfa i tre dager, og jeg ble invitert med på møter i tillegg til at jeg gjorde fem intervjuer. Jeg fikk selv velge hvem jeg ønsket å intervjue.

Beta

Beta utvikler ulike applikasjoner for en rekke plattformer, både mobil og PC, og har blitt en stor aktør med over 200 utviklere siden oppstarten på 90-tallet. Jeg besøkte Beta ved to anledninger, høsten 2008 og høsten 2009, tilsammen fire dager. Første gang utførte jeg fem intervju, som jeg valgte ut blant ansattprofiler som personalsjefen mente var aktuelle ifølge mine kriterier. Den andre gangen intervjuet jeg to informanter som ble valgt ut eksplisitt fordi de hadde sluttet i Beta – prøvd noe annet – og så kommet tilbake igjen. Tanken var at disse ville ha interessante erfaringer og ha gjort seg opp en del meninger om nettopp hvorfor de ønsket å jobbe i organisasjonen. Beta hadde enkelte team som jobbet med smidige metodologier. Jeg fikk ikke tilgang til å delta på noen møter, men jeg spiste lunsj med ulike grupper de dagene jeg var på besøk, og pratet med folk i fellesarealene.

Charlie

Charlie utvikler produkter for ulike plattformer, og baserer hovedproduktet sitt på blant annet fri programvare. Siden oppstarten på 90-tallet har Charlie blitt en internasjonal aktør innenfor sin nisje. Bedriften hadde over 80 utviklere ansatt da jeg besøkte dem sent i 2008, og over 200 ansatte totalt. Jeg hadde en god dialog med personalsjefen i Charlie om å få komme på besøk. Underveis i intervjuprosessen innså jeg etter hvert at jeg ikke fikk tilgang til å intervjue de utviklerne jeg helst ville. Dette var delvis på grunn av at personalsjefen ikke lot meg slippe til i utvikler-miljøet. Da jeg forsøkte å rekruttere medlemmer til intervjuer ved møtepunktene (kaffemaskin og lønsjrom) ble jeg imidlertid som oftest møtt med at folk ikke hadde tid. Dermed ble utvalget i denne bedriften noe skjevt i forhold til opprinnelig plan, og jeg endte opp med å intervjue en utvikler, tre på support og en leder. Dette bekymret meg en del til å begynne med, men i ettertid har jeg sett at intervjuene ble svært verdifulle. Ved blant annet å intervjue to unge menn som jobbet på support fikk jeg innsyn i måten de beundret og beskrev det høyverdige med det å være en høystatus-programvareutvikler i Charlie, og jeg fikk beskrivelser av topputviklerne fra deres perspektiv.

Delta

Delta utvikler produkter i mediemarkedet, og ble startet opp rundt 2000. Selskapet var i sterk vekst da jeg gjennomførte fire intervjuer der våren 2009, og hadde nettopp passert 300 ansatte globalt. Utvikleravdelingen som jeg besøkte besto av rundt 30 ansatte. Jeg tok opprinnelig kontakt med Delta fordi jeg hørte de hadde en kvinnelig utvikler (Dina), og på dette tidspunktet

var jeg på utkikk etter flere kvinnelige programmere å intervjue. Etter at jeg intervjuet henne hadde jeg fortsatt dialog med avdelingslederen (Dagfinn). Jeg gjennomførte flere intervju i organisasjonen, blant annet med Dagfinn selv.

Foxtrot og Echo

Fridtjof og Even ble intervjuet utenfor organisasjonene og nettverkene de var en del av. Fridtjof hadde midlertidig lagt programvareutvikling på hylla da jeg intervjuet ham. Da han var aktiv jobbet han hovedsakelig alene, men var også del av et norsk utviklermiljø med bånd til utlandet. Even er ansatt hos en stor internasjonal fri programvare-aktør. Organisasjonen har kontor i Norge, men Even jobber hovedsakelig fra hjemmet sitt. Både Even og Fridtjof ble valgt ut fordi jeg tenkte at deres perspektiv ville bidra med nye perspektiver. Jeg har ikke besøkt Echo, og jeg har ingen observasjoner herfra.

6.1.3 Utviklerne

Stillinger og roller

En karriere er tradisjonelt forstått som en vertikal forflytning oppover eller nedover i et hierarki, enten innenfor en organisasjon eller innen en profesjon. I Charlie kunne folk med lavere grad av utdanning regne med å forflytte seg oppover etter hvert. Ellers var den øvre klassen med high skill-utviklere relativt låst, og folk kunne ikke i stor grad forflytte seg oppover og fortsette med utvikling. En leder (Dagfinn) beskrev hvordan det i hans avdeling egentlig ikke var noen mulighet for utviklere til å stige i gradene. Dette måtte i så fall være ved å bevege seg ut av avdelingen og mot en av de andre områdene i bedriften. Informantene er navngitt med samme forbokstav som pseudonymet til bedriften hvor de var tilsatt.

Stillingsbetegnelsene i bransjen er mange, og det er ingen enighet om hva de betyr. Informantene omtalte seg som senior-software engineer, software developer, utvikler, programvareutvikler, lead developer – men alle kunne ha lik utdanning og utføre oppgaver som i de fleste henseender er like. I tillegg intervjuet jeg et par systemutviklere. Jeg har gjennom hele avhandlingen kalt alle med informatikk-utdanning eller stillinger som i hovedsak inneholder programmering eller systemutvikling for "programvareutvikler". I tillegg intervjuet jeg supportpersoner, som arbeider i støttestillinger for utviklerne, hovedsakelig med Quality Assurance (QA), som bidrar med testing og debugging av programvare, Technical Writing (dokumentasjon) og support (mellomledd mellom kunder og utviklere). Alle disse er av anonymitetshensyn samlet i kategorien "support". Jeg intervjuet også fem ledere på ulike nivåer,

som av anonymitetshensyn er samlet under betegnelsene leder/sjef. Utvalget er fra tilsammen seks nasjonaliteter, og disse er heller ikke spesifisert av anonymitetshensyn. Utdanningene er også til en viss grad obfuskert. Flere av informantene hadde spesialutdanninger som er ganske unike. Jeg har delt mellom kategoriene ingeniør eller informatikk, sistnevne er ren programmeringsspesialisering. Ingeniør er samlekategori for alle ingeniørutdanninger.

Informantene brukte både tidligere og nåtidige erfaringer om objekter/subjekter i sine evalueringer. I en viss grad forventet jeg å få diakrone data når jeg spurte om erfaringer fra tidligere arbeidsplasser, eller om de kunne vurdere en tidligere situasjon mot en nåtidig situasjon.

Pseudonym	*Bedrift/foretak*	*Stilling*	*Alder*	*Utdanning*	*Nasjonalitet*
Anders	Alfa	Programvareutvikler	47	Ingeniør (3 år)	Norge
Anton	Alfa	Programvareutvikler	30	Ingeniør (5 år)	Norge
Anne	Alfa	Programvareutvikler	36	Ingeniør (ph.d.)	Norge
Arvid	Alfa	Programvareutvikler	32	Ingeniør (5 år)	Norge
Amund	Alfa	Programvareutvikler	28	Ingeniør (5 år)	Norge
Borghild	Beta	Leder	32	Informatikk (5 år)	Norge
Britt	Beta	Programvareutvikler	38	Informatikk (5 år)	Norge
Bjørnar	Beta	Support	31	Ingeniør (5 år)	Norge
Bill	Beta	Support	59	Ingeniør (3 år)	Utlandet
Birger	Beta	Leder	45	Ingeniør (3 år)	Norge
Bjarte	Beta	Programvareutvikler	34	Ingeniør (ph.d.)	Norge
Benny	Beta	Programvareutvikler	32	Informatikk (5 år)	Utlandet
Charles	Charlie	Support	21	Ufullført/Selvlært	Utlandet
Casper	Charlie	Support	24	Ingeniør (3 år) / selvlært	Norge
Christoffer	Charlie	Leder	30	Informatikk (5 år)	Norge
Craig	Charlie	Support	34	Ingeniør (ph.d.)	Utlandet
Cihan	Charlie	Programvareutvikler	30	Ingeniør (5 år)	Utlandet
Dina	Delta	Programvareutvikler	31	Ingeniør (5 år)	Norge
Dagny	Delta	Leder	34	Ingeniør (5 år)	Norge
Daniel	Delta	Programvareutvikler	30	Informatikk (5 år)	Norge
Dagfinn	Delta	Leder	34	Informatikk (ph.d.)	Norge
Even	Echo	Programvareutvikler	32	Informatikk (5 år)	Norge
Fridtjof	Foxtrot	Programvareutvikler	45	Ingeniør (ph.d.)	Norge

Tabell 4: Informantprofiler

17 av informantene er utviklere eller har ingeniør/programvareutviklerutdanning. Fem av informantene jobber nært tilknyttet utviklere, i support-posisjoner. Noen av personene med

utviklerutdanning jobbet som ledere på intervjutidspunktet. Alle lederne hadde enten tidligere programmeringserfaring eller informatikkutdanning.

Tidskulturer

Programvareutvikling er ofte assosiert med en grenseløs tidskultur, men dette synes ikke å gjelde mange i utvalget mitt. Westenholz (2006) gjorde en studie av tidsidentiteter blant IT-arbeidere i Danmark. Tidsidentitet ble definert som en kombinasjon av den faktiske arbeidstiden og den symbolske betydningen av et skille mellom arbeid og fritid. Fire grupper ble identifiserte. *Blurred timers* utgjorde den største gruppa, og hadde et uklart skille og stor variasjon i tilgjengelighet for jobben. *Invaded clock timers* jobbet tidvis utenfor normal arbeidstid, men hadde en tydelig grense mellom fritid/arbeid på det symbolske planet. *Clock timers* orienterte seg etter normal arbeidstid, og deres symbolske forståelse av skillet mellom fritid/arbeid reflekterte dette. Den minste gruppa ble kalt *task timers*, og lot antallet og størrelsen på arbeidsoppgavene definere både arbeidstid og symbolsk betydning av tidsbruken. I mitt utvalg spiller også tid en viktig rolle. I analyse 2 av smidig-håndbøkene er tidsenheter den viktigste entiteten. De fleste i analyse 1 vil ut fra Westerholz kalles clock timers, mens noen få også faller inn under de andre kategoriene. Mitt utvalg består hovedsakelig av stabile eller voksende bedrifter. Mange av informantene mine har likevel erfaringer fra mer kaotiske arbeidsplasser, og jeg har derfor noe data om denne typen arbeidsplass. De fleste i utvalget mitt har god utdanning, trygge arbeidskontrakter og gode lønnsavtaler. Majoriteten velger selv en strukturert arbeidsdag, og å jobbe max. 40-timers uke. Bare lederne, samt Even, oppgir at de vanligvis jobber over 40 timer i uka.

6.1.4 Framgangsmåte

Bedriftsbesøkene og intervjuene ble gjennomført i 2008 og 2009, i tilsammen fem perioder. I Alfa, Beta og Charlie var jeg på besøk i tre dager, og jeg fikk låne kontor eller sitte i et konferanserom. Jeg spiste delvis lønsj med ulike utviklere hver dag. Delta besøkte jeg flere ganger over en periode på 3 uker. Intervjuet med Fridtjof foregikk hjemme hos ham. Intervjuet med Even ble gjennomført i et konferanserom i et kontorfellesskap i et foretak hvor verken jeg eller han jobbet, men han hadde jobbet der før. Jeg kom tilbake til Beta ett år etter første besøk og gjennomførte to ekstra intervjuer. Alle intervjuene ble transkribert verbatim av meg, bortsett fra intervjuene med Bjarte og Even som ble satt ut til konsulent. Alle kjennetegn som kan identifisere informantene er anonymisert eller obfuskert.

Intervjuene ble gjennomlest og kategorisert i flere omganger. I steg en markerte jeg uttalelser med klare grensetrekkinger og/eller som definerte god fagpraksis. Ved å skille mellom god/dårlig

sender informantene også signaler om sin egen identitet og deres opplevelse av trygghet, verdighet og ære (Epstein 1992). Bortsett fra at jeg hadde en vag ide om at jeg var på utkikk etter en hackerkultur eller prestasjonsbaserte gaveøkonomier, stilte jeg som forsker uten forhåndskategorier til intervjuene. Intervjuguiden min hadde noen nøkkelspørsmål som spurte etter vurderinger av typen "hva er god utvikling", "kan du fortelle om en kollega som har gjort noe stort eller beundringsverdig", "kan du fortelle om en prestasjon eller hendelse du er stolt av" og "kan du fortelle om en periode i livet hvor du var veldig inspirert eller produktiv – hvordan skiller denne perioden seg fra situasjonen du befinner deg i nå?". I tillegg markerte jeg utviklernes egne grensetrekkinger. Jeg delte inn grensetrekkingene i moralske, kulturelle eller sosioøkonomiske grenser. Moralske grenser omhandler karakter, integritet eller arbeidsmoral, kulturelle grenser omhandler smak, formell utdanning og oppførsel, mens sosioøkonomiske grenser har med variabler som klasse, status og profesjonell suksess å gjøre (Lamont 1992:9).

- *Moralske grenser:* Disse skiller folk etter blant annet karakter, integritet eller arbeidsmoral. Når Dagfinn skiller mellom karriereutviklere og folk som brenner for det de gjør, så er det et uttrykk for både arbeidsmoral og karakter.
- *Kulturelle grenser:* Har med smak, formell utdanning og oppførsel å gjøre. Når Christoffer tar avstand fra utviklere som bare følger egne agendaer og ikke jobber i team, trekker han en kulturell grense.
- *Sosioøkonomiske grenser:* Har med sosioøkonomiske variabler som klasse, status og profesjonell suksess å gjøre. De fleste utviklerne i utvalget mitt tjener godt. Når de trekker distinksjoner på bakgrunn av hobbyer som golf og seiling er dette eksempler på sosioøkonomiske grenser.

Disse inndelingene viser tydelig at grenser trekkes på tvers av vanlige kategoriseringer som stillingstyper, posisjon i hierarkiet og kjønn/alder. Intervjuene besto også av andre typiske typiske kvalitative spørsmål, om informantenes biografi, arbeidspraksis m.m.

I steg to ble disse utsagnene kategorisert etter hvilket ideal eller øverste prinsipp de henviste til. Jeg fikk fire tydelige kategorier av *verdigheter* og fem mindre tydelige kategorier. De fire kategoriene med et rikt utvalg sitater kalte jeg (1) *team/hybrididealer*, (2) *lidenskap/hackeridealer*, (3) *strukturer, planer og perspektiv/industri/ingeniøridealer* og (4) *ukompleks/smidig*. Fem kategorier fremsto som mindre tydelig, og ble senere fjernet eller satt sammen med de fire første kategoriene. Disse var i utgangspunktet (1) briljant, intelligent (2) oppmerksom og detaljfokusert,

(3) kunnskapsrik, (4) med vilje til å eksperimentere, (5) åpen for kritikk.

I steg tre skrev jeg den symbolske grenseanalysen ut fra kategoriene hackeridealer, ingeniøridealer, hybrididealer og smidige idealer. Her viser jeg hvordan informantene posisjonerer faglige verdigheter opp mot hverandre når de snakker om arbeidspraksiser og meningen med arbeid. Tilsammen utgjør fagverdighetene et evalueringsrepertoar som informantene mine i stor grad deler og behersker, men som de ikke er enige om betydningen av.

6.1.5 Pragmatisk analyseopplegg

Forestillingen om at det eksisterer *faglige verdigheter* er sentral i analysen. Med faglig verdighet eller fagverdighet forstår jeg sett av idealer, logikker eller mikrokompromiss (med referanse til flere mer eller mindre lokale verdiverdener), som brukes av aktørene for å rangere kolleger, relasjoner, objekter og skille mellom godt og dårlig håndtverk. Fagverdighetene kommer til uttrykk når informantene skal gyldiggjøre hva som er god fagpraksis og hva de anser som viktige kriterier hos en kollega. Enkeltpersoner og grupper kan i større eller mindre grad bruke forestillingsverdenene i en verdighet for å forstå meningen med arbeidet, arbeidspraksiser og sin egen rolle i organisasjonen. Med dette mener jeg at fagverdighetene kan være konstituerende for identitet. Informantene er imidlertid ikke bundet til et ideal eller en fagverdighet, de kan ha flere forståelser av en god utvikler, og referere til ulike verdigheter i løpet av et intervju eller en setning. Dette er litt av styrken i analysen. De mulige logikkene i organisasjonene og faget består av et mangfold av idealer, med historisk og strukturell forankring, som kan benyttes på tvers av stilling, alder og kjønn, og som gir sine egne føringer for handlinger. Fagverdigheter representerer altså *ikke* typeskikkelser. Jeg fant imidlertid at ulike miljø, som for eksempel medlemmer eller beundrere av subgrupper i Charlie og Delta, i større grad forbandt hackerverdigheten til sin egen arbeidspraksis og identitetskonstruksjon. I alle organisasjonene fant jeg henvisninger til en hybrid verdighet, denne kunne være generell og knyttet til det å kunne kommunisere med kunder, kolleger og det å formidle.

I hvor stor grad kan da jeg si at de faglige verdighetene har markerte grenser mot hverandre? Jeg forstår fagverdighetene som eksklusive i den forstand at de er regulert av evalueringsprinsipp som ikke kan sammenlignes direkte. Hvorvidt man er kreativ, effektiv, sosial eller flink til å fokusere representerer ikke-kompatible målestokker. Betydningen av hvert enkelt prinsipp må derfor legitimeres, og vurderes opp mot betydningen av andre prinsipper – her er det relativt klare grenser mellom hvilke idealer og prinsipper aktørene i mitt utvalg gyldiggjør. Siden verdighetene i noen grad er baserte på kompromiss eller mikrokompromiss, er det ikke alltid det finnes et enkelt øverste prinsipp. Verdighetene har ulike forutsetninger for å kunne reproduseres,

og over tid vil de kunne endre seg.

Analysen følger opplegget jeg skisserer i kapittel 5.2. Jeg tar utgangspunkt i informantenes uttalelser, og kategoriserer disse i fire ulike fagverdigheter. I tillegg til å knytte verdightene til beskrivelser av god fagpraksis, viser jeg hvilke verdiverdener de faglige verdighetene referer til, og hvordan disse brukes for å *kvalifisere* eller *gyldiggjøre* ulike praksiser, og hvordan disse har både muliggjørende og begrensende kraft i ulike subkulturer og i ulike situasjoner. Til sist vurderer jeg hvorvidt disse fagverdighetene er stabile eller reproduserbare, og beskriver hvordan de utgjør et gyldiggjort repertoar som informantene bruker for å beskrive meningen med arbeidet eller uttrykke en profesjonell identitet i formasjon eller endring.

For å skille idealene fra aktørene og organisasjonene, skriver jeg analysen på tre nivå: *Aktørnivået*, *kulturnivået* som inneholder størrelser som teknologi, organisasjoner og roller, og *idealnivået*, som består av mer eller mindre stabile fagverdigheter, eller mikrokompromiss som forstås i lys av generelle verdiverdener. Analysen er organisert etter de fire fagverdighetene, og jeg presenterer sitater, betraktninger og observasjoner fra interjvuene underveis. I noen grad trekker jeg fram biografier, og forsøker å vise hvordan enkelte informanter i større eller mindre grad forsøker å knytte seg til en bestemt type verdighet. Fortellingene til Fridtjof, Anders, Casper, Dagny, Christoffer og Dagfinn er spesielt viktige.

6.1.6 Fordeler

Det som etter min mening gjør symbolsk grenseanalyse og Boltanski og Thévenots rammverk spesielt fruktbare i min analyse er at de tar utgangspunkt i et aktørperspektiv, samtidig som metoden er sensitiv for kulturelle og kollektive preferanser som aktørene bruker som en referanseramme. Moralske og ideelle valg blir muliggjort og begrenset eller styrt av eksterne faktorer. Informantenes vurderinger kontekstualiseres dermed i forhold til føringer og strukturer i organisasjonene og det omkringliggende samfunnet. Det blir tydelig hvordan disse er med på å forme verdier og motivasjoner over tid, og er med på å utvikle og endre faglig/ og profesjonell identitet gjennom at aktørene gylddigjør de idealene de mener er viktige i forhold til noen referanser. Siden programvareutviklingsfeltet har mange spenninger,[22] foregår det akkurat nå et omfattende kategoriseringsarbeid blant arbeiderne som befolker det. Denne metoden er derfor unik metode for å vise aktørenes egen rolle i å forme mening og praksis i faget.

Spørsmålet om den gode utvikler ledet til mye interessant i intervjuene. Som profesjonelle arbeidere har alle en mening om hva en god arbeider er – men ikke alle reflekterer aktivt rundt

22 Både siden feltet er nytt, under utvikling, og har i seg spenninger mellom fri og produsenteid programvare, uformell/formell kompetanse og mellom ulike nasjonale kulturer .

spørsmålet i hverdagen, eller har anledning til å uttrykke det de mener. Dermed ble spørsmålet en innfallsport til refleksjon og tanker rundt fag og arbeid som de fleste informantene så ut som de satte pris på. Metoden gir mulighet for å diskutere transformasjoner og endringer, uten å "tre kategoriseringer" nedover hodene på informantene, og uten å kategorisere intervjuene i ettertid på måter informantene ikke vil kunne kjenne seg igjen i. Hvis man i stedet tar utgangspunkt i idealer hos informantene, og snakker om størrelser som *effektivitet*, *fleksibilitet* og *inspirasjon*, bruker man et språk som informanter kjenner seg igjen i, og som referer til idealer som de kan være stolte av.

6.1.7 Svakheter

Det metodiske opplegget har også noen svakheter. Som forsker observerer jeg legitimering i intervju-uttalelser, ikke reelle samhandlingssituasjoner. Det er godt mulig at informantene ville brukt et annet repertoar i reelle disputter med kolleger. Opplegget utforsker i liten grad hvorvidt aktørene er strategiske, og hvordan de bruker makt for å vinne privilegier på arbeidsplassen. Kanskje er det også en svakhet at mennesker ser ut til å tillegge verdighet ulik betydning – noen ser ut til å ha større personlig behov for å teste visse legitimeringer og knytte seg til eksterne idealer og objekter. Man kan kanskje si at noen mennesker er mer pragmatiske enn andre. Dette bidrar til å skape dynamikk, uro og tegne opp noen skarpe grenser som rammeverket fanger opp, mens personer som trekker mindre tydelige grenser muligens kan bli oversett.

Preferanser for å benytte evalueringsrepertoar i ulike situasjoner vil antakelig påvirkes av ulike psykologiske faktorer som eksempelvis selvsikkerhet og trygghet. Introverte personer vil muligens søke mot logikker og verdisystemer som belønner indre motivasjon. Det ligger ikke innenfor denne avhandlingens rammer å vurdere i hvilken grad psykologiske faktorer påvirker legitimering, men problemet bør nevnes siden flere har hevdet at programvareutviklere er innadvendte, og at faglige prestasjoner brukes for å kompensere for manglende sosiale kunnskap (Capretz 2003; Faulkner 2000). Forestillingen om hackeren som asosial men mektig har også kommet til uttrykk i deler av norsk forskning på kjønn og teknologi, hvor hackere ble beskrevet som nærmest hegemoniske i visse miljøer (Gansmo, Lagesen, and Sørensen 2003). Mine funn tyder uansett på at sterk indre motivasjon i liten grad har gode vilkår for å skape utbredt mening og motivasjoner i IKT-organisasjoner.

6.2 Diskursanalyse – den ukomplekse verdiverdenen

Den andre analysen består av en programvarestøttet analyse av 12 håndbøker i smidig utvikling og prosjektmanagement. I analysen skildrer jeg trekk ved *det ukomplekse* som skiller seg

fra innholdet i de andre verdiverdenene i korpuset av bøker. Jeg beskriver også det ukomplekse verdisystemet gjennom å påpeke verdier, prinsipper, idealer, subjekter, teknologier, objekter, relasjoner og tester som skiller seg fra språket i 60- og 90-tallets management-litteratur som dokumentert av Boltanski og Thévenot (2006) og Boltanski og Chiapello (2005). Deretter kodifiserer jeg verdisystemet ved hjelp av den samme grammatikken som Boltanski og Thévenot brukte i *On Justification* (2006) og viser at den ukomplekse logikken har systematisk karakter.

Min problemstilling i denne analysen er: *Hvilke nye idealer, verdier og prinsipper brukes for å beskrive fortellingene og formidle innholdet i disse bøkene, og hvordan brukes disse for å legitimere at programvare- og prosjekt-management trenger endring?* Spørsmål jeg har hatt i bakhodet under lesningen av smidig-litteraturen er, i tråd med Boltanski og Chiapello (2005:64): Hvilke problemer trekkes fram av forfatterne? Hvilke svar har forfatterne på disse problemene? Hvilke aspekter ved situasjonen som er under lupen forkastes? I denne analysen er jeg altså ikke opptatt av å beskrive mangfoldet av verdiverdener som uttrykkes i bøkene, men den nye logikken i en rendyrket form. Analysen vil potensielt kunne kritiseres for å generalisere endringer i en yrkesretning ut fra noen få håndbøker, attpåtil skrevet hovedsakelig for en USA-amerikansk kontekst. Kritikkene som kommer til uttrykk i bøkene er imidlertid representative for utviklingen av yrket. Interessen for smidig utvikling hos moderne organisasjoner og forskningsinstitusjoner tyder på at disse kritikkene kan generaliseres.[23] Debatten rundt produktivitet i IKT-organisasjoner har vært stor de siste 15 årene, og de smidige løsningsforslagene ser ut til å være del av en global trend, ikke minst fordi smidige prosjekter oftere enn andre prosjekter når de planlagte målene sine (Moløkken-Østvold og Jørgensen 2005). Forrester Research hevdet i 2009 at smidig utvikling har blitt mainstream, med en utbredelse på ca. 35 % globalt (West and Grant 2010; Laanti, Salo & Abrahamsson 2011). Da jeg gjorde det kvalitative feltarbeidet mitt for to år siden, brukte tre av de seks organisasjonene jeg besøkte en eller annen form for smidig utvikling. Universitet i Oslo tilbød som første norske læringsinstitusjon undervisning i smidig metodikk fra og med vårsemesteret 2011.[24] Nokia bruker smidig, Apple, Microsoft og Google bruker smidig (se f.eks. Striebeck 2006) – og NTNUs formidlingsavdeling på web bruker smidig metodologi for å organisere arbeidet.

23 I en kvalitativ analyse av seks bedrifter bekrefter lokalisert i Norge bekrefter jeg at disse kritikkene også benyttes av utviklere i en norsk kontekst, også i organisasjoner som ikke brukt smidig utvikling på det tidspunktet jeg besøkte dem. Jeg kontrollerer imidlertid ikke for en eventuell norsk, kulturell varians i denne analysen.

24 Som en del av kurset systemutvikling, med ca. 300 studenter i året (se også Kirknes 2010).

6.2.1 Om korpuset

De tolv engelskspråklige tekstene som utgjør korpuset i denne analysen fremstiller kvaliteter og idealer i arbeidspraksisen, svakheter med tidligere metoder, og beskrivelser av ulike oppgaver og roller i IKT-organisasjoner. Bøkene er gitt ut i perioden 2000-2010, under emnene *project management* eller *agile software development*, med den hensikt å gjøre IKT-organisasjoner mer effektive, lønnsomme og stabile gjennom å øke bevissthet, drøfte teori, og gjennom å gi beskrivelser av og eksempler på faglig praksis for utviklere og ledere. Bøkene har derfor et delvis normativt og argumenterende uttrykk. Utvalget er gjort ut fra at dette er populære bøker som inneholder de metodologiene informantene mine refererte til, med forfattere som regnes som opphavsmenn til de smidige metodene, som det refereres ofte til på konferanser om smidig utvikling (se f.eks. xp2010.org og xp2011.org) og som hadde høye vurderinger i nettbutikken Amazon.com.

6.2.2 Kildetekster i korpuset

Beck, Kent, og Cynthia Andres. 2004. *Extreme Programming Explained: Embrace Change*. 2nd ed. Addison-Wesley Professional.

Beck, Kent, og Martin Fowler. 2000. *Planning Extreme Programming*. Addison-Wesley Professional.

Berkun, Scott. 2005. *The Art of Project Management (Theory in Practice*. 1st ed. O'Reilly Media.

Cockburn, Alistair. 2001. *Agile Software Development*. 2nd ed. Addison-Wesley Professional.

Cohn, Mike. 2005. *Agile Estimating and Planning*. Prentice Hall.

Cohn, Mike. 2009. *Succeeding with Agile: Software Development Using Scrum*. 1st ed. Addison-Wesley Professional.

Gyurky, Szabolcs de. 2006. *The Cognitive Dynamics of Computer Science: Cost-Effective Large Scale Software Development*. 1st ed. Wiley-IEEE Computer Society Pr.

Jeffries, Ron, Ann Anderson, og Chet Hendrickson. 2000. *Extreme Programming Installed*. 1st ed. Addison-Wesley Professional.

Lui, Kim Man, and Keith C. C. Chan. 2008. *Software Development Rhythms: Harmonizing Agile Practices for Synergy*. 1st ed. Wiley-Interscience.

Martin, Robert C. 2008. *Clean Code: A Handbook of Agile Software Craftsmanship*. 1st ed. Prentice Hall.

Pichler, Roman. 2010. *Agile Product Management with Scrum: Creating Products that Customers Love (Addison-Wesley Signature Series*. 1st ed. Addison-Wesley Professional.

Poppendieck, Mary, and Tom Poppendieck. 2009. *Leading Lean Software Development: Results Are not the Point*. 1st ed. Addison-Wesley Professional.

6.2.3 Framgangsmåte

Jeg utviklet analysen i tre steg. (1) Det første var lese- og drodlefasen, hvor jeg laget en manuell analyse av tre av bøkene fra korpuset (Beck and Andres 2004; Cohn 2009; Poppendieck and Poppendieck 2009). (2) Deretter inkluderte jeg ytterligere ni bøker i utvalget, og lette ved hjelp av dataprogrammet Tropes[25] etter de mest brukte ordene og uttrykkene som ikke allerede hørte hjemme i Boltanski og Thévenots verdiverdener, eller som ellers framsto som viktige for den ukomplekse logikken ut fra hvordan kontekst eller situasjon. (3) Til slutt skrev jeg den ukomplekse grammatikken, og sammenlignet resultater i korpuset med den industrielle og den prosjektorienterte verdiverden fra de engelske oversettelsene av Boltanski og Thévenot og Boltanski og Chiapello. Jeg har ikke sammenlignet resultater fra disse tre verdiverdenene med forekomsten av andre verdiverdener i tekstene. Selv om dette ville vært interessant, har jeg i denne analysen vært opptatt av å rendyrke den ukomplekse verdiverden, og ikke beskrive det brede mangfoldet av verdiverdener slik disse forekommer i bøkene.

For å kunne analysere bøkene (det andre trinnet) brukte jeg elektroniske og scannede versjoner av bøkene (i ett tilfelle) som jeg konverterte til ren tekst. Jeg fjernet innholdsfortegnelser, referanser og søkeordregister først og sist i bøkene for å begrense antallet falske treff, og satt da igjen med tolv tekst-dokumenter med en størrelse på 7 250 823 bytes og 1 188 650 ord.

Programvaren jeg brukte, *Tropes*, er laget for Windows-systemer og kan lastes ned gratis fra http://www.semantic-knowledge.com/tropes.htm. Tropes gjør det mulig å analysere innhold etter hvilke ord og uttrykk som blir brukt og hvordan disse blir satt i relasjon til hverandre. Denne analysemetoden kalles gjerne semantisk eller lingvistisk tekstanalyse. Framgangsmåten jeg brukte ligner måten Boltanski, Thévenot og Chiapello gikk fram i sine sine analyser. De utførte sine tekstanalyser med programvaren @Prospero, på franskspråklige tekster av ulike karakter (delvis oversatt fra engelsk). Analysen er basert på kvalitativ, statistisk bearbeiding av tekstene. Prospero sorterer teksten i kategoriene *entiteter* (vanlig substantiv, egennavn, sammensatte substantiver), *kvaliteter* (adjektiver eller tidligere eller nåværende participles som kvalifiserer entiteter), *tester* (hovedsakelig verb i infinitiv eller konjugert form), ord som markerer eierskap eller relasjoner (pronomen, konjunksjoner, osv.) og *markører* (adverb, men også former som modaliserer uttalelsen), tall og ord som er vanskelig å klassifisere. Tropes måte å behandle tekstene ligner, og kan dessuten programmeres til å behandle tekstene på samme måte som Prospero@. Tropes deler inn teksten i ordklasser og kontekster. Dette gjorde det andre trinnet av

25 http://www.semantic-knowledge.com/tropes.htm

analysearbeidet enklere. For eksempel er ordet *process* nevnt som substantiv 1628 ganger, men 282 som verb (*to process* er en relasjon i den industrielle verdiverden). Tropes regner med ulike former og bøyinger av ordet for de fleste ord. Noen viktige ord som f.eks. *decouple,* lå ikke inn i programmets database, og formene ble derfor skrevet inn av meg som en kategori.

6.2.4 Entiteter og kategorier

Brukeren kan lage egne sammenhenger, entiteter og kategorier, dette kalles å lage et *scenario*. I arbeidet med den ukomplekse grammatikken har jeg konstruert følgende fiktive kategorier:

accessible@: accessible, readable, understandable
agile@: agile, agility
adjust@: adjust, adjustable
clarity@: accessibility, clearness, readability, transparency
cultivate@: cultivate, grow, nurture
complexity@: chaos, chaotic, complex, complexity
cultivate@: cultivate, grow, nurture
decoupled@: decouple, decouples, decoupled, decoupling
discussions@: discussion, oral, talk
distraction@: distraction, distractions, interfere, interference
expose@: exposed, expose
iteration@: iteration, iterations, iterative
leader@: adviser, gardener (gardening), leader, product owner, product owners, scrummaster, scrummasters
orderly@: clean, orderly
overload@: overload, overloaded
professional@: expert, master, professional
reduced@: reduced, isolated, separated
removed@: removed, eliminated
refactor@: refactor, refactors
resist@: resist, sabotage
simplicity@: simplicity, simple, ease
stability@: stability, constancy, steadiness
sustain@: sustainability, sustainable

Inkludert disse ser listen over de oftest brukte entitetene i korpuset sånn ut:

team	7 928	programmer	1 927	prerequisite	1 190
project	5 696	chapter	1 884	programming	1 157
software	4 643	day	1 839	task	1 131
product	3 283	difference	1 763	sprint	1 081
people	3 192	procedure	1 740	idea	1 064
work	3 172	thing	1 740	practice	1 062
time	3 101	director	1 719	estimate	1 049
system	3 099	iteration@	1 665	member	1 048
story	2 791	scrum	1 648	person	1 038
development	2 662	iteration	1 545	case	1 034
way	2 265	user	1 487	owner	1 034
design	2 259	method	1 440	plan	1 025
code	2 249	organization	1 421	point	1 025
test	2 154	information	1 415	value	1 023
customer	2 099	part	1 337	discussions@	1 016
leader@	2 042	to-do	1 241	similarity	1 011
problem	1 945	feature	1 206	...	

Tabell 5: Entiteter med over 1000 treff i korpuset (både substantiv og fiktive entiter)

Project (assosiert med den prosjekt-orienterte logikken) ligger på andre plass med 5 696 forekomster. Jeg var først usikker på om lederroller skulle gjøres til en del av den ukomplekse logikken, men ifølge innholdet i bøkene og viktigheten som tillegges lederskap skal de helt klart det. Jeg har korrigert resultatene sånn at leder-metaforer som finnes i bøkene (som f.eks. gardener) også blir sortert og regnet inn leder-entitet. User er et annet viktig subjekt i tekstene, programmene er skrevet for *brukerne*. Etter at jeg hadde sett hvilke ord og uttrykk som var viktige i tekstene – og som jeg ikke fant igjen hos Boltanski og Thévenot eller Boltanski (2006) & Chiapello (2005), lagde jeg den ukomplekse grammatikken.

6.2.5 Sammenligning mellom den industrielle, prosjektorienterte og den ukomplekse

For å sammenligne representasjonen av ulike verdiverdener (det tredje trinnet) har jeg skrevet inn grammatikken fra den industrielle verdiverden fra den engelske oversettelsen av *On Justification* og den prosjektorienterte verden i *The Spirit of the New Capitalism* og sammenlignet

treff i mitt korpus. Delvis har jeg laget fiktive dummy-variabler som inneholder flere substantiv, men jeg har ikke forsøkt å gjenskape de eksakt samme kategoriene som Boltanski og Chiapello brukte i sine analyser. I stedet har jeg skrevet inn grammatikkene direkte ut fra de engelske oversettelsene, og sammenlignet treff og relasjoner.

Logikk	Industriell	Prosjekt/nettverk	Ukompleks
Totalt	*27 493*	*18 725*	*28 016*
Ekvivalensprinsipp	*586*	*6 239*	*1 291*

Tabell 6: Antall treff i korpuset

Totalt dominerer den ukomplekse og den industrielle logikken i korpuset. Det bør imidlertid ikke legges for mye vekt på akkurat dette poenget, siden jeg ikke har skrevet inn verdiverdenene til Boltanski og Thévenot og Boltanski og Chiapello i det samme programmet som de brukte, jeg har heller ikke programmert verdiverdenene på den samme måten som de viste i Boltanski og Chiapello, med fiktive entiteter. Det som er hovedpoenget mitt er at den ukomplekse verdiverdenen eksisterer, og at den gjør seg gjeldende i tekstene. Det at *industriell* og *ukompleks* logikk dominerer den *prosjektorienterte* logikken rent tallmessig stemmer forsåvidt overens med konklusjonen i "*analyse #2 – den ukomplekse verdiverdenen*". Industrielle og ukomplekse argumenter og objekter brukes i et kompromiss for å begrense den prosjektorienterte verdenen – som fremdeles er den mest utbredte hvis man bare ser på forekomsten av ekvivalensprinsipp.

Under presenterer jeg en tabelloversikt over ordene jeg har søkt etter i de tre største verdiverdenene i korpuset. Den høyre kolonnen markerer det totale antallet treff per overskrift.

Den industrielle verdiverden (27493 treff)	
01. higher common principle	586
efficiency	97
future	123
performance	366
02. state of worthiness	456
functional	231
operational	62
reliable	96
03. state of unworthiness	458
failure	432
inefficient	16
unproductive	8
unreliable	10
04. dignity	3356
work	3170
energy	186

necessary	417
need to	1152
optimize	52
organize	255
process (verb)	282
scenario	121
solve	317
stabilize	9
standardize	13
10. harmonious figure of natural order	4520
organization	1421
system	3099
11. model test	119
achievement	41
launching	61
setting up	17
12. judgement	490
correct	113
effective	365
functioning	12
13. evidence	485
measure	485

Tabell 7

Den prosjektorienterte verdiverden (18725 treff)	
01. higher common principle	6239
activity	511
extension	32
project	5696
02. natural relations	1639
adjust	159
communication	1137
connection	128
coordinate	68
trust	147
03. condition of great man	293
adaptable	5
autonomous	107
enthusiastic	13
flexible	44
in touch	3
involved	102
tolerant	16
04. repertoire of subjects	4543
coach	215
customer	2100
director	1719
expert	166
innovator	5
relationship	294
supplier	43
05. repertoire of objects and mechanisms	1884
alliance	49

company	1152
consent	35
informal	57
link	33
loop	53
network	100
partnership	5
subcontracting	11
technology	354
nervous system	35
06.condition of little person	189
attached	7
intolerant	3
local	107
rigid	12
security	60
07. decline of the city	16
immorality	16
08. status relation	837
integrating	22
provide	815
09. formula of investment	1575
adaptability	3
agile	1330
exclude	10
permissiveness	201
reject	31
10. model test	782
telecommunication	782
11. judgement	621
avoid	296
distance	65
ignore	127
insert	18
participate	115
12. dignity	72
connect	72
13. harmonious figure of natural order	35
Network@ (internet, intranet, lan, network)	35

Tabell 8

Den ukomplekse verdiverden (28 016 treff)	
01. higher common principle	1291
clarity@	123
perspective	299
simplicity@	869
02. state of worthiness	2325
accessible@	62
agile	1404
aware	151
clear	265
decoupled@	39
elegant	34

isolate	35
limit	133
refactor@	64
remove	168
separate	108
sustain	85
10. harmonious figure of natural order	5475
cycle	202
harmony	14
Incremental	180
iteration@	1665
time	3101
transition	252
11. model test	173
continuous	151
Responsiv	22
12. judgement	157
flow	157
13. evidence	719
duration	132
long term	21
quality	508
stability@	58
14. the fall	34
naive	7
plain	14
simplistic	13

Tabell 9

6.3 Svakheter

Den rent kvantitative analysen over ord som jeg referer til over har mange svakheter. Boltanski og Chiapello identifiserte sine verdiverdener i et fransk korpus. Verdiverdenene er i ettertid oversatt til engelsk, og programmert inn i Tropes av meg uten at jeg har hatt mulighet til å skrive inn verdiverdenene på samme måte som Boltanski og Chiapello. Den totale tellingen viser ikke hvorvidt ordene er brukt i positiv eller negativ forstand, og tar heller ikke hensyn til hvilken relasjon ordene står i. Tropes har funksjoner som viser relasjoner mellom ordforekomster, men formatet på disse finner jeg lite hensiktsmessig å gjengi som del av denne teksten. Meningen hvor dette verdisystemet kommer til uttrykk bør dermed leses ut fra tekstene. I analysen løser jeg dette ved å referere passasjer hvor logikken gjør seg gjeldende. Dette ligner for øvrig på måten Boltanski og Chiapello (2005:539-557) har gjort dette på.

7. Analyse #1 – den gode programvareutvikleren

Kategoriene jeg benytter i denne analysen er funnet gjennom informantenes egne symbolske grensetrekkinger, hvor det overordnete temaet er "den gode programvareutvikleren". Informantene bruker grensetrekkinger for å beskrive og kategorisere ulike typer programvareutviklere, verdier og faglige praksiser i intervjusituasjonen. Mens grenseanalyse ofte har vært forbundet med å markere avstand til "de andre" – ser jeg i denne analysen etter grenser som trekkes både for å ta avstand fra, for å markere idealer man strekker seg imot, samt beskrive gode kolleger og miljø man ser opp til.

I utvalget mitt gir grensetrekkingene fire forståelser av hvem en god utvikler er. Jeg har delt forståelsene inn i fire kategorier som inneholder *hackeridealer*, *ingeniøridealer*, *hybride idealer* og *smidige idealer*. Jeg analyserer disse fire forståelsene som "mikrokompromiss" – lokale verdiverdener som hver for seg utgjør en mer eller mindre stabil verdighet knyttet til fagutøvelsen. I analysen kaller jeg kategoriene *fagverdigheter*. Hva en fagverdighet inneholder er i noen grad sammenholdt med hva tidligere teoretikere har skrevet. Hackerverdigheten er for eksempel navngitt ut fra blant annet noen av idealene som nevnes i Levy (1984) og Castells (2001), men innholdet som strukturerer og fyller denne kategorien er tatt fra utvalget mitt.

7.1 Fire forståelser

Her vil jeg kort presentere de fire fagverdighetene. *Hackerverdigheten* er konstituert av stort sett moralske grensetrekkinger som fremhever verdien av lek, lidenskap og eksperimentering. Om man har evne til å *tilegne seg ny kunnskap selv*, om man klarer å benytte *autonomi* til noe konstruktivt, om man har *fantasi* og *vilje til å eksperimentere,* eller om man *presterer,* er symbolske grenser som noen av informantene bruker for å skille mellom gode og dårlige utviklere. Jeg har sortert grensetrekkingene under hackeridealer, siden de ligner på beskrivelsene av hackere i diverse litteratur, spesielt Levy (1984), Himanen (2001) og Graham (2004).[26] Hackerverdigheten er

26 Når jeg trekker inn Himanen (2001) her er det spesielt gjennom beskrivelser av hackere som lidenskapelige og indre motiverte i arbeidet. Når Himanen snakker om hackeretikken bruker han en sjupunktsmodell som er mer kompleks, denne kommer jeg tilbake til mot slutten av avhandlingen.

ganske sammensatt, og inneholder referanser til den artistiske, familiære og kollektive verdiverden, samt idealer forbundet med konkurranse og marked.

Ingeniørverdigheten består i store trekk av kulturelle grensetrekkinger som stort sett holder seg innenfor den industrielle verdiverdenen. Gode planer, effektive løsninger og målbare resultater regnes som verdifulle. Jeg forbinder disse kategoriene med en ingeniørverdighet i tråd med beskrivelser fra Kraft (1979), Turkle (1984), Turkle og Papert (1990) og Dijkstra (1993).

Den *hybride* verdigheten tillegger det å være sosial og flink til å kommunisere høy verdi, og er satt sammen av kulturelle og moralske grensedragninger som refererer til industrielle, kollektive og prosjektorienterte verdiverdener. I tillegg til å være faglig dyktig, må utvikleren ha gode sosiale egenskaper (Dahlbom 1997; Friedman 1989; Goles, Hawk, and Kaiser 2009) og evne til å danne og vedlikeholde nettverk.

Den *smidige* verdigheten er satt sammen av stort sett kulturelle grensetrekkinger som fremhever verdien av å forenkle, konsentrere, filtrere, stenge ute og fokusere. Dette er idealer som i liten grad er brukt i rammeverket til Boltanski og Thévenot, men i utvalget mitt blir de brukt for å danne et mikrokompromiss sammen med den industrielle verdiverden. Ekvivalensprinsippet er derimot ikke effektivitet, men forenkling. Den "smidige" fagverdigheten, slik jeg påviser den i mitt utvalg, er ikke forbeholdt utviklere som jobber med smidige metodologier – den utgjør et repertoar med gyldighet blant utviklere som ikke bruker metodologiene.

De fire fagverdighetene eksisterer i spenning, og har koherens på den måten at aktørene ikke uten videre gyldiggjør en god faglig praksis med referanse til flere idealer samtidig. Man lukker ikke døra (smidig verdighet) fordi det gir bedre rutiner og planer (ingeniørverdighet). Man sier ikke ja til å reise til utlandet for å lede prosjekter (hybrid verdighet) fordi det gir mer tid til å eksperimentere og leke (hackerverdighet). I noen grad kan imidlertid verdighetene spille sammen, f.eks. finner jeg at den smidige verdigheten i stor grad kan brukes for å legitimere fagpraksis på "lag" med de andre verdighetene. Tilsammen utgjør disse fagverdighetene et tverrsnitt av en profesjonell identitet i formasjon.

7.2 Hackerverdighet

Flere av informantene trakk fram at kunnskapen de benytter i arbeidet er tilegnet gjennom *personlige erfaringer* og *eksperimentering*. I Charlie møtte jeg to personer som hadde avbrutt utdanningsløp, og fått trainee-stillinger på grunnlag av kunnskap de hadde tilegnet seg på egen hånd. Casper er en av disse. Casper er 24 år og jobber som support. Charlie er en av få større

norske bedrifter som utvikler fri programvare. Casper bidrar med å teste, lese over og presentere kode, finne feil (såkalte bugs) og være en støttespiller for utviklerne. Jeg velger å åpne denne analysen med Casper av flere grunner. For det første er han *ung*, og for det andre *mangler* han *formell utdannelse* utover en bachelorgrad. Casper har imidlertid lært seg programmering på egen hånd. Det at han er selvlært ser ut til å utgjøre en viktig del av hans måte å presentere seg selv på i intervjuet. I tråd med litteraturen om hackere, mener Casper at en god utvikler trenger å ha en "passion".

> *Jeg tror at – man trenger ikke å ha det som en hobby, men det må være en sånn "passion". Så når man setter seg ned foran datamaskinen, så kan man tenkte at man har det kult på jobb. Og at man ikke tenker "argh, nå skal jeg jobbe" (Casper).*

Casper mener videre at gode utviklere bør oppleve arbeidet som lystbetont, og idealene skisseres ut fra den personlige karakteren eller *innstillingen* hos arbeideren. Argumentet impliserer en grense for hva Casper mener at en god utvikler kan tillate seg – i denne typen yrke *kan* man ikke tenke at arbeidet er kjedelig. Fridtjof, en annen fri programvareutvikler, knytter god utvikling til *vilje*:

> *Han må ha intelligens. Eh. Interesse. Viljen til å lage noen ting som er bra. Viljen til å eksperimentere veldig mye. Ofte når man har et produkt, så har det vært 100 versjoner av produktet før det endelige produktet er ferdig. Man må eksperimentere mye (Fridtjof).*

Vilje til å eksperimentere knyttes sammen med intelligens og personlig interesse for faget. En god utvikler må være villig til å prøve og feile. Det å være en god utvikler kan ifølge denne forståelsen ikke læres på et universitet eller en høyskole. *[...] mye av programmering er problemløsning. Du finner som regel ikke svaret i en bok (Even).* Beskrivelsene baserer seg på personlige karakteristikker, som støtter en forestilling om at "alle" utviklere kan bli gode hvis de bare *investerer*, hvis de legger *viljen* til, hvis de har den rette *innstillingen*.

7.2.1 Fantasi og evnen til å lære

Kunnskapen Casper benytter seg av på jobben har han tilegnet seg gjennom programmering på fritida. "*Universitetstiden – den er fin å ha på papiret, men jeg lærte egentlig ikke så veldig mye*", sier Casper. Mens Casper gikk på universitetet, tok han oppdrag på et nettsted hvor folk utlyste programmeringsjobber. På fritiden utvikler Casper fremdeles små applikasjoner og spill, og han

kobler dette til det å føle seg inspirert og tilegne seg ny kunnskap på en morsom måte. Underveis i intervjuet spurte han meg om han kunne få vise meg noe på den bærbare datamaskinen min. Han åpnet et spill som jeg ikke visste om, men som var installert på maskinen min fra før. "*Dette spillet har jeg lagd på fritida*", sa han. Innenfor enkelte utviklermiljø regnes det som en relativt stor ære å få applikasjoner eller spill inkludert i de store operativsystem-distribusjonene, og det var tydelig at Casper var stolt av å kunne understreke det han fortalte ved å vise fram programmet til meg. Selvlært kunnskap, eller det å kunne tilegne seg kunnskap på egen hånd, er et ideal som flere av informantene mine verdsetter, og spesielt når det kan knyttes til *kreasjon* (den artistiske verden), et ferdig produkt skapt uten industrielle former for investering i arbeidet. Samtidig trekkes det ofte fram at verktøy og metoder man bruker i dag avviker fra det man lærte om på universitetet.

I motsetning til Casper, som mangler utdanning utover bachelorgradsnivå, så har Fridtjof en ph.d.-grad – på et helt annet felt enn programmering. Fridtjof mener at det at han har akademisk erfaring fra et *annet* område enn informatikk er noe positivt.

> *For mange år siden snakket jeg med en annen programmerer. Han sa, "with software, you're only limited by your imagination". For de fleste er fantasien en stor begrensning (Fridtjof).*

Fordi han mangler informatikkutdanning, mangler Fridtjof også *begrensningene* som kommer med på lasset, når han skal realisere en idé i ett eller flere programmeringsspråk. Fridtjof drar en grense mellom seg selv og de utviklerne som mangler *fantasi*. Dette er en moralsk grense, i tråd med Lamont (2001), den sier noe om *karakteren* hos personer Fridtjof vil distansere seg fra. Fridtjof mener selv at fantasi og lek har vært viktig i hans egen karriere: Selv om Fridtjof manglet fagutdanning, manglet han aldri *idéer*. Det å tenke utenfor boksen, det å vedlikeholde en kreativ flyt, det å tenke på siden av etablerte rammer – det er verdifulle egenskaper hos en god utvikler, ifølge Fridtjof. En tung, faglig ballast, kan derimot gå utover fantasien. *[...] mye av programmering er problemløsning. Du finner som regel ikke svaret i en bok (Even)*.

7.2.2 Autonomi og indre drivkrefter

Å disponere sin egen tid er et ettertraktet privilegium blant noen utviklere. For å kunne få og utvikle de virkelige gode ideene kreves det frihet fra rutiner, systemer og alle andre begrensende faktorer. Fridtjofs tilnærming til problemet var å si opp jobben han hadde, fordi han ville forsøke å leve som uavhengig programmerer og disponere sin egen tid. Han jobbet som selvstendig fri

programvareutvikler i over fem år fra hjemmet sitt og etter hvert fikk han økonomisk støtte. Da jeg gjennomførte intervjuet med Fridtjof høsten 2008 hadde han imidlertid sluttet som programvareutvikler og tatt seg jobb i en assisterende stilling i en middels stor teknologisk bedrift. Han jobbet ikke lenger med programmering.

De fleste av intervjuene mine ble utført i møtelokalene ute på bedriftene jeg besøkte. Dette var også det jeg foreslo da jeg kontaktet Fridtjof, selv om jeg signaliserte at intervjuet ville handle mest om hans fortid som såkalt *fri programvare*-utvikler.[27] Fridtjof ønsket imidlertid at jeg skulle komme hjem til familiens villa på kveldstid, hvor han bodde sammen med kone og barn. Jeg antok på intervjutidspunktet at Fridtjof ønsket å ta imot meg fordi han ville være lojal mot sin nåværende arbeidsgiver. I ettertid har jeg tenkt at det kunne være at han ønsket å ta imot meg i de omgivelsene han jobbet i som fri programvareutvikler. Fridtjof er meget velformulert, med et sterkt engasjement og en positiv framtidstro. Omgitt av mørke møbler og bugnende bokhyller, fremstår Fridtjof som en mann med innsikt og autoritet. I disse omgivelsene jobbet han i over fem år med egne prosjekter, og det at han kunne disponere over sin egen tid fremstår som viktig i fortellingene hans. Fridtjofs historie som fri programvareutvikler begynte på slutten av 90-tallet, da han var lei av å flytte rundt fra by til by etter prosjektstillinger som forsker. Det var mange utlyste prosjekter på hans felt, men ingen faste stillinger. Dessuten, siden han nå hadde stiftet familie, ønsket han å være bofast både for sin egen og familiens del. Fridtjof hadde lagt av penger fra sin 15-årige akademikerkarriere, og hadde det økonomisk sett romslig nok til klare seg noen år uten inntekt. Han kjøpte seg sin første datamaskin, installerte Linux og brukte snart datamaskinen til å programmere. Ved å si opp jobben og begynne å utvikle fri programvare, realiserte Fridtjof et hackerideal som skriver seg rett inn i mytene om de store hackerne i historien (Levy 1984):

> *Det er kanskje litt for sterkt sagt at jeg var en "slave" [i mine tidligere jobber], men jeg hadde ofte en følelse av at jeg ikke hadde nok tid for å utvikle egne ideer. Tiden jeg holdt på med fri programvare, den tiden jeg var uten fast jobb, det var en slags tilbakebetaling til meg selv, jeg ville gjøre noe med livet som var interessant og som jeg selv rådet over (Fridtjof).*

Fridtjof var kompromissløs, og han fulgte hands-on-imperativet ved bygge maskinen

27 Fri programvare referer til åpen kildekode, alle kan se i koden, endre den og publisere egne programmer, i motsetning til produsenteid programvare, som er lukket og proprietær. Spillet som Casper lagde er basert på åpen kildekode.

komponentvis og ha full kontroll over programvaren (Levy 1984). Indre motivasjon, personlig utvikling og følelsen av å gjøre noe meningsfullt, er grunner som Fridtjof lister opp for sitt uortodokse karrierevalg.

Siden han ikke hadde noen arbeidsgiver, var Fridtjof faktisk helt fri. Han kunne selv velge når han ville jobbe, hva han ville jobbe med og hvor lenge han ønsket å jobbe med med et program, helt autonomt. Dette gjorde også at han kunne eksperimentere i forhold til jobbpraksiser. Fridtjof foretrakk å jobbe på dagtid, men mente at det av og til kunne kunne gi noen kreative fordeler å jobbe på natten:

> *Jeg merket at hvis jeg skulle regne ut noe, så var det best å gjøre det om morgenen. Men hvis jeg sto litt fast, og måtte prøve ut noe helt nytt – så var det OK å jobbe på natten. Jeg oppdaget at jeg hadde flere hemninger på dagtid, disse forsvant da jeg var trøtt. På dagen tenkte jeg "nei, jeg kan ikke gjøre sånn her – det funker aldri"... Og sent på natten, når jeg sto fast, så våget jeg mer. Det skjedde flere ganger, når jeg turte å tenke "gale ting" – så kom jeg videre (Fridtjof).*

Fridtjofs fortelling har så langt pekt mot en indre forankring av motivasjon og identitet. I den grad han referer til noe allment godt, er dette knyttet til den inspiratoriske eller artistiske verdiverden. Etter hvert ble imidlertid Fridtjof ferdig med flere prosjekter, fikk oppmerksomhet, sponsorer og et fagmiljø knyttet til fri programvare. Alle de personlige motivasjonene og investeringene fikk etter hvert en kobling til dette fellesskapet.

7.2.3 Fri programvare og det kollektive gode

Mange i utvalget mitt jobber med fri programvare-verktøy, og flere gir uttrykk for at fri programvare er noe de synes er viktig, noe de gjerne de skulle ha bidratt mer til selv. For en del synes akkurat fri programvare å være identititetsskapende. Daniel sier for eksempel at han gjerne kunne gått ned i lønn hvis han kunne begynne å jobbe i Trolltech.[28] Daniel og familien bor imidlertid i en helt annen by, og det er egentlig ikke aktuelt å flytte. Flere trekker fram at penger er underordnet, det viktige er rettferdighetsprinsippet. Casper flyttet til Oslo kun for å jobbe i Charlie. Charles knytter videre produktene de lager i Charlie til et allment gode, for hele samfunnet:

28 Trolltech var et Oslo-basert firma som lagde splitt/lisens-programvare, og var spesielt kjent for en del av programvaren sin. De ble kjøpt opp av finske Nokia i 2008, og delvis solgt videre i 2011.

> *Det er mange prosjekt som bruker [produktene] våre, mange skoler. [Produktet] brukes i fattige land som bruker Linux, fordi Windows er for dyrt, eller fordi maskinvaren ikke klarer å kjøre Windows. Så da kan man kjøre Linux med [produktet] – også har man en datamaskin som funker åpent og bra. Jeg tror nok veldig mange av oss har forstått det, at det er open source som gjelder, at det hjelper folk over hele verden – og ikke bare folk som vil tjene penger. Det hjelper også å vite at [produktet] brukes i kjernekraftverk og i flygeledersystem. Så da vet man dette er et produkt som gjør at verden går framover også – og ikke bare hjelper fattige mennesker. Så for meg tror jeg det er viktig at lisensen forblir åpen her, at produktet forblir åpent – og gratis for dem som vil ha det. Hvis det skulle bli et lukket produkt, så kommer jeg antakelig til å flytte herfra (Casper).*

Fri programvareproduktet til Charlie er til det beste for *alle* samfunnslag, ikke bare fattige. Casper sier direkte at han er stolt av å jobbe i Charlie. Even, som er 32 år og tobarnspappa, fant ut at han ville slutte å jobbe i konsulentselskap, og heller arbeide for en større, utenlandsk fri-programvare-aktør fra hjemmekontoret sitt. Arbeidet er godt lønnet og fast, men det var det faktum at denne organisasjonen er assosiert med fri programvare som var utslagsgivende for Even. For ham spiller det mindre rolle at jobbtidene til tider er litt uortodokse:

> *Akkurat nå jobber jeg sammen i et knyttet team sammen med to andre – en som bor i London og en som bor i Sao Paolo – så døgnrytmen bli litt varierende. Siden jeg har unger i barnehagen, må jeg liksom opp. Så da prøver jeg å ta det fra dem er i barnehagen til de kommer tilbake. Og så blir det litt kveldsjobbing, noen ganger blir det mye kveldsjobbing. Men det er veldig sjeldent det er noe tidspress, det er heller lystbetont (Even).*

Fri programvare er viktig, og ser ut til å gi et sterkt insentiv for å investere tid eller engasjere seg. Even mener at fri programvare kunne vært en mulig løsning på mer fornuftig ressursbruk i det offentlige i Norge.

> *Jeg har sett hvor mye konsulentfirmaer suger ut av statlige etater og prosjekter, siden jeg har vært en del av det selv. Og hvor konsulentfirmaet sitter igjen med kunnskapen om hvordan produktet ble [laget], og sitter igjen med kildekoden eller eierskapet nesten, da. Og så selger de den samme løsningen til en annen [etat] som trenger nesten det samme, men [de] må likevel [...] betale full pris, for de vet ikke at mesteparten av lagd fra før (Even).*

Fri programvare knyttes i utvalget mitt til verdighet på mange nivå, men spesielt gjennom at det er til det beste for samfunnet. Dina bruker en del fri programvareverktøy i sin yrkesutøvelse. Når jeg spør om hun føler seg forpliktet til å gi noe tilbake, svarer hun:

> *Nei ... men har forsåvidt tenkt på det. Men det kan jo ta litt tid, føler nok ikke at jeg har tid. Men i forhold til det å gi tilbake og sånn, så har jeg vært og sett på FN sine nettsider. De har program hvor de spør etter fri programmeringshjelp til ulike prosjekter, og jeg har hatt veldig lyst til å gjøre noe sånt. Er innom og ser på hvilke prosjekter de har utlyst med jevne mellomrom, men de fleste ønsker 11 timer i uka og denslags. Det blir litt mye når man jobber 40 timer i uka fra før (Dina).*

Det at fri programvare finnes og er såpass mye brukt ser ut til å skape et behov for å gi noe tilbake. Fri programvare er bygget opp gjennom en delekultur, gjennom at det *ikke* er noen krav til å betale for å bruke det. Produktet er åpent, fritt – og man kan som regel bruke det uten vederlag. Paradoksalt nok ser dette ut til å gi noen moralske føringer om at man nettopp da *bør* gi noe tilbake. Dette peker mot idealer i den *kollektive* verdiverden.

7.2.4 Gruppemedlemskap og prestasjonskultur

Etter hvert kom Fridtjof med i en arbeidsgruppe som besto av ulike fri programvareutviklere. Fridtjof benytter moralske grensedragninger når han skal beskrive egenskapene hos disse kollegene. Disse grensedragningene viser trekk Fridtjof vil assosieres med – ikke distansere seg fra:

> *Jeg synes at de som jobber med dette i arbeidsgruppa – det er på en måte sånne mennesker som var med på å bygge opp Norge etter krigen. Det er folk som er villig til å jobbe* sammen. *De er praktisk anlagt, og de har en egen evne til å samarbeide (Fridtjof).*

Fridtjof omtaler de andre i arbeidsgruppa som politisk engasjert, med høy integritet og vilje til å få til noe. Dette er egenskaper han ser opp til og som han selv ønsker å bli assosiert med. I sine beskrivelser av kollektivet snakker Fridtjof om tillit, ærlighet og lojalitet. Dette er størrelser som hører hjemme i den *hjemlige* verdiverden. Han fremstiller det som at han ble oppriktig glad i dette miljøet. Fridtjof forteller at han vokste veldig på å være del av dette miljøet, og han mener at alle deltakerne hadde gjensidig utbytte av hverandre. Fridtjof husker tiden hvor han deltok på disse møtene som veldig positiv, og han forteller hvordan gruppa hele tiden motiverte ham til å yte litt ekstra:

> *På et møte lovte jeg ut å lage en spesiell funksjon til programmet gruppa vår utviklet. Og – jeg hadde ikke noe godt grunnlag for å love ut det – for jeg hadde aldri lagd noe lignende før. Men jeg ønsket å gjøre det, og folk ble veldig glade for at jeg tok på meg oppgaven (Fridtjof).*

Med gruppa som en referanse og en utfordrer for egne ambisjoner, lovte Fridtjof å gjøre ting han strengt tatt var usikker på om han klarte. Ved hjelp fra gruppa og faglige tilbakemeldinger fra digitale fora, klarte han oppgaven han tok på seg. Dette ga selvfølgelig Fridtjof en følelse av suksess og mestring. Dette kaller jeg en prestasjonsbasert identitetsforankring, i tråd med Mauss (1970). Fridtjof hadde fått et fellesskap, og han ble motivert av å gi tilbake. Dette kan assosieres lokalt til den hjemlige verdiverden, men også til den *kollektive* verdiverden, gjennom det at Fridtjof ofret seg for egen vinning og fordi han kjempet for en sak. Fridtjof hadde nå kvalifisert seg for å fortsette i gruppa, han hadde fått seg et miljø. Fridtjof hadde på dette punktet i livet sitt ganske mange ulike motivasjoner for å fortsette med fri utvikling. Han syntes fremdeles det var morsomt og utfordrende, han hadde fremdeles autonomi, og han fikk i tillegg tilbakemeldinger, hjelp og anerkjennelse fra et faglig miljø som han var en del av.

Prestasjon er også koblet til konkurranse og rivalisering, som er ekvivalensprinsipp i *markedsverdenen*. Dette er ikke så tydelig i eksemplet over med Fridtjof, men det er desto klarere i organisasjonen Charlie. Charlie, som er en stor aktør involvert i fri programvare internasjonalt, fremmer en organisasjonskultur basert på prestasjon og konkurranse, gjennom ulike typer konkurranser. Casper forteller:

> *I blant har vi dedikerte konkurranse-dager, hvor vi dedikerer en hel dag til å følge opp feilmeldinger. Så da deler vi opp folk i lag, også kan man vinne prices hvis man fikser flest feil (Casper).*

Konkurransedager holdes ved jevne mellomrom, alt etter behovet i organisasjonene. Et annet interessant eksempel på en institusjonalisert praksis som er prestasjonsfremmende, er konseptet *Google-fredag*, som har fått sitt navn etter at Google innførte praksisen og gjorde den kjent. Konseptet har gjerne ulikt navn fra organisasjon til organisasjon, og Charlie heter konseptet noe helt annet. Tanken bak er imidlertid at organisasjonen kan nære fram inspirasjon, innovasjon og lek ved å sette av arbeidstid, for eksempel annenhver eller hver fredag, til personlige prosjekter. Google-fredag er dermed et slags privilegium for de fast ansatte – ikke bare utviklerne, men også

support-personer som Casper. For organisasjonen sin del er fordelen at denne friheten potensielt vil kunne trekke til seg de mest kreative utviklerne, og at mange av ideene uansett vil kunne relateres til programvarepakken organisasjonen selger. Dette er også en "bottom-up"-tilnærming, utviklerne kan selv velge hvordan de skal investere tiden sin, og dermed kan det oppstå arenaer for uformell læring og informasjonsutveksling gjennom samarbeidsprosjekter. Tiltaket ser ut til å fungere bra for å danne gruppetilhørighet i Charlie.

Tiltaket ser også ut til å kunne skape en motivasjon for å tilhøre de *verdige*. Flere av utviklerne i utvalget mitt bruker blogger og intranett for å presentere ideer og arbeid de har utviklet iløpet av Google-fredag og på fritiden. Bloggene fungerer som kanaler for å presentere arbeid som blir regnet som kreativt eller innovativt. Charles er 21 år, utvalgets yngste, og jobber som support engineer i samme avdeling som Casper. Han påbegynte en informatikkutdannelse i hjemlandet, men hoppet av da han ble tilbudt trainee-stilling i Charlie. Da jeg spør om hvilke kolleger som imponerer ham, og om han har noen fortellinger knyttet til dette, forteller han engasjert:

> *Here I mean, that happens all the time here, the developers are so talented, just before I came up here, one of the developers released a blog entry having [...] our widgets embedded in a 3D-game, and that's ridiculouos! And they are doing things like that, I mean, I am going crazy downstairs [...] And the support engineer in the office next to me, he ran over and said "You have to see this, you have to see this". We were both running over, and I was going crazy too. And then I ran to the engineer sitting besides my office. sThere is lots of this (Charles).*

Gjennom bloggene og intranettet kan alle de ansatte få ta del i gledene og prestasjonene til topp-utviklerne i Charlie. Charles blir mest imponert over innovative ideer satt ut i praksis. Det er imidlertid ikke delt *kunnskap* Charles trekker fram som høyverdig i bloggposten, det er selve *prestasjonen*. Utviklerne konkurrerer om *kreativ* verdighet. Charles uttrykker beundring og synes at de utviklerne som poster innovative ideer og som investerer mye er høyverdige. Charles' mål er å bli tilbudt fast jobb snart, og kanskje få litt programmeringserfaring før han etter noen år flytter tilbake til hjemlandet og til en forhåpentligvis god jobb.

Prestasjonskultur som forstått gjennom Mauss innebærer en egoistisk motivert forklaring på verdighet, den som yter mest får også høyest status. Dette er er det ikke mange i utvalget mitt som ser ut til å anerkjenne, men Bjarte sier:

> *Det er jo sånn det er med open source, det er jo det som er så genialt med det. Folk tror at det er gutter som jobber gratis. Det er jo ikke det. Det er mange egoister som hjelper seg selv og tjener bra på det (Bjarte).*

Det at prestasjoner presenteres på blogger og som ferdig implementerte ideer, gjør at utviklere i Charlie i mindre grad kan forhandle direkte om hva kreativ verdighet består i. Høy status i Charlie er knyttet til en kombinasjon av prestasjon, innovasjon og presentasjon. Spesielt i Charlie er dette tydelig, i mindre grad i Delta, og i liten grad i Alfa og Beta.

Noen utviklere ser ut til å trekke symbolske skiller mellom hvem som regnes som verdige til å benytte seg av Google-fredag og de som ikke gjør det. Flere sa at de valgte ikke å benytte seg av privilegiet. Spesielt Cihan kobler dette mot en forståelse av verdighet. Cihan er en 30 år gammel utvikler, og kommer opprinnelig fra India. Han har jobbet i Charlie i to år. Cihan er en av dem som kan benytte seg av fri fredag-konseptet gjennom stillingen sin, men som velger å ikke gjøre det. Jeg spurte ham i hvor stor grad han følte seg inspirert for tiden, sammenlignet med når han var på toppen i karrieren.

> *Uh, I am way below ideal. [...] I always think that I'm doing very less, or not doing, it's, uh ... up to my potential. That's always what I think about myself. [...] I have to do more (Cihan).*

Cihan trekker opp grenser mellom seg selv og de kreative kollegene når han skal sammenligne sin egen prestasjon, og han finner at han ikke kan måle seg. Dette kan tolkes som at Cihan mener at han ikke *fortjener* å bruke Google-fredag, så lenge han ikke jobber raskt nok ellers i uka. Cihan uttrykker imidlertid ikke dette som noe tap. Han er enig i at han for tiden mangler det som skal til, og derfor har han ingen ambisjoner om å forsøke å inkludere seg blant de mest kreative. Cihan aksepterer og uttrykker at han ikke er et verdig medlem av det kreative fellesskapet. Senere skal jeg vise at Cihan knytter sin faglige verdighet til hybride idealer, hvor det å være sosial og jobbe sammen er viktig.

7.2.5 Sosiale grenser

Selvlært kunnskap, fantasi og kreativitet er viktig for fagutøvelsen. Hvordan kan organisasjoner finne mennesker med disse egenskapene? Dagfinn jobber som leder på høyere nivå i Delta, og er beslutningstager i rekrutteringen til nye utviklerstillinger i firmaet. En av arbeidsoppgavene hans er å skille gode utviklere fra mindre gode. (Dette er for øvrig ikke en jobb

han gjør med særlig glede. Dagfinn bruker svært ofte legitimeringer som kan assosieres til hackeridealer, og dette repertoaret åpner i liten grad for å finne verdighet i leder- og administrative oppgaver.) Dagfinn har informatikkutdannelse på doktorgradsnivå. I motsetning til Casper og Fridtjof trekker ikke Dagfinn noen klare grenser mellom det å være kreativ og det å ha en fagutdanning. I hans verden har alle han møter en eller annen form for IT-faglig utdannelse. Likevel er Dagfinn overbevist om at titler, diplom eller vitnemål ikke kan gi ett tilstrekkelig bilde av den kunnskapen og de erfaringene som er nødvendige for å gjøre en god jobb. Dagfinn forbinder sin egen suksess til noen personlige moralske kvaliteter:

> *Jeg ser også at mye av det som gjør at jeg har gjort det bra i arbeidslivet i tillegg til den rent faglige utdannelsen, har vært den generelle interessen jeg har hatt for temaet som startet lenge før selve utdannelsen begynte. Koding har vært en form for hobby opp gjennom hele livet mitt. Og det at man lever og ånder for tingene har virkelig hjulpet i det å prestere ting (Dagfinn).*

Akkurat sånn som Fridtjof og Casper, tar Dagfinn utgangspunkt i at han har riktig type *innstilling* og interesse for programmering. Dette har hjulpet ham å tilegne seg ny kunnskap, og til å gjøre det bra i livet, ifølge han selv.

Da Dagfinn snakket om sin evalueringsrolle i forbindelse med nyansettelser, tegnet han først følgende symbolske grense mellom produktive og mindre produktive utviklere:

> *Man ser ofte om personen kun ser på utvikling som en karriere eller om dette er noe de virkelig brenner for. Det er ofte sånn at hvis man brenner for det, så er de langt mer nyttige og produktive som utviklere (Dagfinn).*

Med utgangspunkt i forståelsen han bruker for å definere sin egen suksess, skiller Dagfinn mellom utviklerne som brenner for arbeidet, og utviklerne som bare er innom utviklingsfaget for å få litt erfaring. Dette er også først og fremst en moralsk grensedragning, men Dagfinn knytter samtidig distinksjonen til noe konkret i bedriften: *produktivitet*. Dette er et ideal som ellers hører hjemme i industriverdenen (se lenger nede). Dagfinn kombinerer imidlertid idealer fra den artistiske og den industrielle verdenen. *Karriereutviklerne* blir nemlig ikke lenge, er hans erfaring; de blir bare fram til de innser at det ikke er noen muligheter for en videre intern karriere. Og mens de er ansatte har de jevnt over en lavere produktivitet. Det var ikke noen andre i utvalget mitt som knyttet bånd mellom hackeridealer og industrielle idealer på denne måten.

For å skille de kandidatene han mener er gode fra dårlige, spør Dagfinn etter fritidsinteresser. Hvis kandidatene har noen "sideprosjekter", eller hvis de har prosjekter som ikke er knyttet til en bestemt arbeidsoppgave, eller hvis de drifter noen websider, så anser han dette som gode indikatorer. Søkerne må også gjennom en praktisk programmeringstest. Her vurderer Dagfinn løsningen søkerne kommer opp med, men han ser samtidig etter noe mer:

> *Det er [...] viktig å prøve å se det lille hintet av inspirasjon som du ofte kan finne, det handler ikke bare om å finne løsningen på problemet, men også indikere en eller annen innsikt eller erfaring som du vanligvis ikke finner (Dagfinn).*

Det å yte litt ekstra og legge litt ekstra sjel i arbeidet utgjør dermed det magiske skillet Dagfinn er på jakt etter når han evaluerer jobbsøkere. Hvis man har dette lille ekstra, i tillegg til fagkunnskap, kan man få en jobb i Delta. Dagfinn innrømmer imidlertid at han ikke alltid klarer å finne de gode utviklerne. Dagfinn trekker en annen grense, som skiller de mer *produktive* utviklerne fra de andre:

> *Det er en tradisjon i dette firmaet å ha litt sånn artige sideprosjekter. Det slår meg det at det er en viss gruppe [programmerere] som alltid gjør denne typen ting. Og det er de samme typene personer som jeg har fått inntrykk av at virkelig gleder seg over denne typen arbeid. Folk som virkelig har en teknisk interesse for fagfeltet [...] utover kontortiden (Dagfinn).*

Den eksklusive gruppa utmerker seg ved at Dagfinn observerer at utviklerne gleder seg over arbeidet, og at de kommer med gode kreative ideer inn i bedriften. Dagfinn forsøker å skape en arbeidskultur som *daglig* har rom for eksperimentering og lek. Utviklerne i Delta kommuniserer vanligvis på et internt chat-system. Dagfinn forteller imidlertid om en nylig episode hvor en av utviklerne lagde seg en ny kommunikasjonskanal – et 64-punkters LED-display som hadde en sentral plass i kontorlandskapet. Dette gjorde at svært mange kunne se den.

> *Det startet med et enkelt program for å visualisere om byggstatusen på bygg-serveren var i orden, slik at det lyste opp rødt når det var noe galt. Men det begynte raskt å utvikle seg til å bli mer - den viste temperaturen ute, dato og tid, samt spilte pong-spill når den ikke var i bruk – og alt mulig rart. Det ville vært lett å si at det ikke er bra at de bruker arbeidstiden sin til dette. Men jeg vet at folkene som er involvert samtidig er de som yter mye og som er blant de mest produktive i firmaet (Dagfinn).*

Dagfinn forsøker bevisst å nære fram en god arbeidskultur på sin arbeidsplass, og mener derfor at denne gruppas moralske kvaliteter, og i noen grad kulturelle tilnærming til jobben, legitimerer det at de har noen privilegier som andre utviklere ikke nyter. I Dagfinns fortelling er både han selv og gruppa han tenker på som gode utviklere helt overlegen gruppa han omtaler som karriereutviklere. Førstnevnte gruppe innehar de moralske egenskapene en god utvikler etter hans mening bør ha: En positiv innstilling, tid til å programmere og fikle litt også på fritiden, og klare å omsette denne verdifulle kunnskapen til produkter.

7.2.6 Uoppnåelig lidenskap

De fleste grensetrekkingene som jeg har sortert i dette kapitlet brukes for å beskrive hvilke egenskaper som er viktige for en god utvikler, og er hovedsakelig basert på *moralske* grensedragninger, som rangerer personlig karakter, integritet eller arbeidsmoral. I tillegg til å legitimere privilegier, ulikhet og gruppedannelse på jobben, sier disse moralske grensedragningene noe om hvem informantene selv ønsker å være. Disse skillene tillegges stor betydning av noen få, og kan sees spesielt tydelig i fortellingene til Casper, Fridtjof og Dagfinn. Det å forsøke å leve opp til sine egne moralske standarder i en verden og en organisasjonskultur som fremdeles er profittbasert er imidlertid ikke uproblematisk. De aller fleste som knytter meningen med arbeidet til forestillinger om lidenskap, lek, eksperimentering og prestasjon fremstår generelt sett som frustrerte. Dette kommer spesielt fram hos lederne som har fagbakgrunn i utvalget mitt. Birger er leder for en utviklingsavdeling i Beta, men jobbet som programvareutvikler for en del år tilbake. Jeg spurte ham om han kunne tenke på et tidspunkt i livet hvor han var veldig kreativ, produktiv – eller følte seg veldig inspirert – og beskrive dagens situasjon ut fra dette. Birger svarte "*Jeg vil si at at jeg er et stykke fra det idealet faktisk*". Han forklarer dette med at lederjobben ikke gir han rom for å jobbe med det han egentlig er interessert i, de tekniske problemene. Birger løser dette problemet ved å forsøke å oppholde seg i utviklingsavdelingen, og klemme inn så mye programmering som han klarer – og det er ikke mye. Han jobber allerede over ti timer i døgnet på grunn av lederoppgavene. Dagfinn uttrykker akkurat det samme. Jeg lar Dagfinn beskrive arbeidsdagen sin selv:

> *D: Vi har alltid mye å gjøre, i form av ønsker fra kunder, og prosjekter som går og så videre. Min oppgave er ganske satt egentlig, budsjettet tar jeg hånd om, jeg tar hånd om selve den daglige "hva skal vi gjøre akkurat nå" og liksom fordele det utover, både snakke med hver enkelt utvikler, men også støtte meg ganske kraftig på ledergruppa, som jeg var tidligere var medlem av. Og, ja, være, ja rett og slett håndtere mye administrasjon, den litt*

mindre morsomme delen av arbeidet da. Det blir en del papirarbeid og den typen ting. Og jeg er en representant for utviklingsavdelingen i alle sammenhenger hvor det er nødvendig, f.eks. det å dra til kunder og snakke på vegne av utviklingsavdelingen da, og ellers prøver jeg å samarbeide mye med de andre utviklingsavdelingene vi har i firmaet. Jeg prøver så mye som jeg kan å fremdeles holde meg innenfor de tekniske aspektene ved disse tingene, men jeg ser det at etter jeg kom over i denne rollen her, så har det vært mye mindre tid til de tekniske tingene. Så jeg må si jeg savner, ja, jeg savner faktisk de tekniske utfordringene fremfor det jeg driver på med for øyeblikket (Dagfinn).

JF: Mm. Vil du da si at det er et mål for deg å komme tilbake til dette igjen?

D: Ja, det vil jeg si. Som sagt, den rollen som leder for utviklingsavdelingen det var noe som, når jeg ble spurt om det, egentlig sa jeg nei altså. Det var ikke noe jeg ønsket å gjøre. Selv om det var en form for hopp i karrierestigen. Så var det ikke noe jeg var interessert i, jeg var interessert i de tekniske tingene. Men det som overbeviste meg var at det på det tidspunktet ikke fantes noen gode alternative kandidater. På det tidspunktet var det mange som ikke hadde den tekniske biten som var nødvendig for å gjøre denne typen jobb. Så jeg gjorde det mer for det at det var nødvendig for firmaet sitt velvære. Så, men, selvfølgelig, jeg synes det har vært veldig lærerikt det jeg har gjort nå, og jeg har fått et annet perspektiv på ting. Men det er ikke, det er egentlig ikke dette aspektet ut av programvareutvikling som fascinerer meg. Jeg brenner ikke like sterkt for denne typen arbeid som jeg gjør nå til dags, som jeg gjorde tidligere (Dagfinn).

JF: Men vil det være aktuelt for deg å gå "ned" igjen til en programmerer-stilling da, eller?

D: Jo, det vil jeg si. Jeg har aldri vært fokusert på tittel eller lønnsnivå for den saks skyld heller, så. Å gå tilbake til å bli en utvikler, det hadde jeg sett på som positivt altså (Dagfinn).

Dagfinn er altså ikke spesielt fornøyd i lederjobben sin, og lengter tilbake. Flere jeg spurte om dette spørsmålet, og spesielt lederne, reagerte med en slags nostalgi, de hevdet at de hadde mer tid og rom for å være kreative før.

Mens Fridtjof jobbet sammen med de andre utviklerne i gruppa, mottok han for første gang godtgjøring (i form av offentlig støtte) for arbeidet. Omtrent samtidig gikk flere større aktører for alvor inn i fri programvareindustrien. Internasjonale fri programvare-aktører som IBM donerte

kode, og USA-baserte selskap som Red Hat dannet regionbaserte arbeidsgrupper og arbeidsceller for å fange opp de beste fri programvare-utviklerne i ulike deler av verden. Et stund etter at Fridtjof ble med i kollektivet, ble de beste utviklerne headhuntet til et større søster-foretak lokalisert til Sverige. Mange utviklere dro hit, men Fridtjof ble igjen. Fridtjof bruker den mest suksessfulle Linux-distribusjonen i massemarkedet, Ubuntu, for å konkretisere at han mener utviklingen med fri programvare ikke har vært udelt positiv:

> *Ubuntu-suksessen er både bra og dårlig. Det som er bra er at de har klart å heve Linux til et nivå hvor Linux blir tatt på alvor. Det er bra. Det som er mindre bra med Ubuntu, er at før så var det mange små utvikler-miljø, og disse er nå ødelagt (Fridtjof).*

Introduksjonen av store og profesjonelle fri programvareselskaper har ifølge Fridtjof gjort betingelsene for de små miljøene vanskeligere. Profesjonaliseringen av fri programvare-miljøene har spist av både brukermassene og engasjementet for de små prosjektene som han selv bidro til. Dette har bidratt til at de beste fri-programvareutviklerne har blitt ansatte hos store selskap. Fridtjof mener at de lokale fri programvare-nettverkene er splittet, kompetansen er sentralisert og i en del tilfeller flyttet ut av landet. Selv om det fremdeles finnes en del nettverk og lokale celler, blir strategier og planer for fri programvare-feltet nå tatt i utlandet. Dermed har den kreative og utfordrende prestasjonskulturen som preget utviklingsmiljøet blitt forsvunnet. Fri programvare har blitt big business. Fridtjof fremstiller dette som et tap.

Den eneste i utvalget mitt som fremstiller det som at han klarer å leve helt i tråd med hackeridealene, er Casper – og han er svært ung, og jobber ikke som utvikler – han jobber med support, som regnes for å være en assisterende stilling. I Delta ble det omtalt som en karrieremessig blindvei å begynne å jobbe som utvikler, men i Charlie er det mulig å jobbe seg oppover i utvikler-hierarkiet. Casper begynte som en trainee, men etter ett år ble han tilbudt fast jobb i bedriften. Han sier han fikk lov å velge hvilken stilling han ønsket å ha. Casper unngikk de tunge programmeringsavdelingene, og søkte seg heller mot support fordi han oppfattet posisjonen som *friere* og mer *variert*. Casper mener at det er dumt å bli sittende fast å jobbe med samme type kode, med samme type optimaliseringsproblematikk hele tiden, så det vil han helst unngå. Han posisjonerer seg selv i forhold til de andre utviklerne når han forteller følgende om sin egen stilling: "*For da blir det veldig variert, det finnes uendelige mengder jobb å gjøre, problemer å løse. Så da kan jeg gjøre hva jeg vil*." Casper ville svært gjerne jobbe for Charlie, som har høy status i fri programvaremiljøet, men han vil samtidig beholde sin autonomi og frihet i tillegg. Casper er ung, men på intervjutidspunktet hadde han i hvert fall så langt klart å realisere en slags lykkelig

hackeridentitet som ga ham trygg inntekt, og etter det han selv sier; meningsfullt arbeid, og tid til å utvikle egne programmer på si. Dette er han alene om i mitt utvalg.

7.2.7 Motivasjoner for å ta informatikk/ingeniørutdanning

Svært mange i utvalget mitt oppgir at grunnen til at de valgte å studere informatikk eller ingeniørfag, var at de holdt på med spill, programmering da de var barn/ungdom. Dette er ikke uvanlig blant programvareutviklere. Forestillinger om at programmering er koblet til lek, lidenskap og eksperimentering blir da en del av grunnen til at man velger en informatikk/ingeniør-utdanning.

> *D: Det var vel min far som introduserte meg litt for programmering når jeg var i tolvårsalderen. Ikke noe sånn avanserte greier, helt enkelt. Men før det hadde jeg jo brukt datamaskiner til spilling også da, vi hadde Commodore 64 når jeg var liten (Dina).*
>
> *JF: Og denne lærte du deg å bruke selv?*
>
> *D: Ja, hehe, måtte det hvis vi skulle spille de spillene vi ville. Pappa satte på et sånt mattespill til meg og søstra mi, men vi ville heller spille andre spill. Tvert han gikk så satte vi på det spillet vi helst ville spille (Dina).*

Jeg knytter denne typen spilling og utforsking til en indre motivasjon. Man lærer seg å bruke teknologien fordi det er morsomt og fordi man har lyst. Mange flere oppgir lignende erfaringer med datamaskiner i barne- og ungdomsårene.

7.2.8 Reproduksjon av hackerverdighet

Forestillingen om at at *inspirasjon* og glede over arbeidet er like viktig som fagkunnskap er tilstede, i en eller annen form, i alle bedriftene jeg besøkte. Men spesielt i Charlie og Delta ser det ut som om disse idealene blir viktige for noen av informantene for å reprodusere *meningen* med arbeidet. Idealene har blant annet en indre forankring. Lek og inspirasjon er koblet til konkurranse og prestasjon, gjennom mer eller mindre formelle konkurranser, synlige bidrag i blogger og på intranett, og privilegier som Google-fredag. Her er det *kollektive gode* også et sterkt element i meningsproduksjonen, spesielt synlig gjennom en uttalt stolthet eller beundring for fri programvare og open source. Hackeridealene åpner opp for forestillinger om at det ikke er fagkompetanse som skiller gode og dårlige utviklere, men hvilken *innstilling* man har til arbeidet, og sekundært hva man klarer å produsere. Disse vurderingene likestiller fagutdannelse med

amatørkunnskap og eget-initiativ. Grensetrekkingene henger i stor grad sammen med *moralske* vurderinger av utviklerens karakter, integritet eller arbeidsmoral, og i mindre grad sammen med *kulturelle* forventninger til atferd. Idealer som rangerer innstilling og engasjement høyere enn titler og formelle krav kan forstås gjennom den *artistiske* verdiverdenen. Ekvivalens måles ut fra inspirasjon og lidenskap for faget. De mest høyverdige er de som klarer å omdanne denne inspirasjonen i konkret produksjon, gjennom å programmere på fritida eller forplikte seg overfor fellesskapet. Dermed er det også elementer av prestasjonskultur gjennom det å yte og ofre seg for fellesskapet (den *familiære* og det *kollektive* verdiverden). Tekniske prestasjoner blir gjerne delt og vurdert på blogger eller på intranett, og de brukes for å konkurrere (*markedsverdenen*) om kreativ verdighet. Antallet publiserte innlegg eller kvalitet og oppfinnsomhet brukes for å måle hvem som er de mest kreative og dermed de mest *verdifulle*. Prestasjoner og å yte tillegges særlig stor verdi av de miljøene som er eller ønsker å være medlem av "hackerfellesskapet". Prestasjonene fungerer også som et referansepunkt for en legitim og i noen tilfeller selvvalgt *ekskludering* av utviklerne som regner seg som mindre kreative. I to av organisasjonene ble symbolske kategoriseringer assosiert med hackerverdighet brukt for å tydeliggjøre gruppetilhørighet og legitimere ulik praksis når det gjelder ressursfordeling og tidsbruk mellom ansatte med samme stillingbetegnelse. I Charlie og Delta gis utviklerne som regnes som spesielt *kreative* privilegier i form av at de får jobbe med egne prosjekter i arbeidstiden.

Det jeg kaller hackerverdighet består dermed av idealer fra den artistiske (inspirasjon, engasjement), den familiære (ære, prestasjon og ytelse) og den kollektive verdenen (for fellesskapet). I tillegg er markedsverdenen representert (å konkurrere). Tilsammen danner disse idealene et kompromiss som er relativt ustabilt, og vanskelig å reprodusere. Dette er tydelig gjennom eksempelet med Dagfinn: Han prøver bevisst å velge gode kandidater, men på tross av de raffinerte metodene han bruker klarer han ikke å forhindre at det blir ansatt karriereutviklere som bare er innom bedriften en kort stund. Selv er han frustrert over at han må bruke så mye tid på planlegging og administrasjon, og hvis han kunne ville han gjerne gått tilbake for å jobbe som ren utvikler. Men, i hvert fall våren 2009, da intervjuet ble gjort, er det bare han som kan utføre dette arbeidet. Etter hans mening er det fremdeles ingen andre som har hans kompetanse eller deler hans syn på god utvikling og visjoner for firmaet.

Testen som kan reprodusere hackeridealene er kompleks og sårbar. Både inspirasjon, engasjement, prestasjoner og fellesskap må sjongleres for at verdigheten skal oppfattes som levende og dynamisk. Som jeg vil vise utover i teksten, krever dette et omfattende legitimeringsarbeid. Problemet for hackeridealene er todelt. For det første vil evalueringsprinsippene bak de ulike verdiverdenene potensielt annullere hverandre. Verdigheten

man oppnår ved å vinne en konkurranse kan egentlig ikke sammenstilles med verdigheten av kreativ, strømmende inspirasjon. Disse er uforenlige. Lokal ære og tillit er ikke forenlig med distansen som kreves hvis man virkelig skal ofre seg for det kollektive beste, noe som understrekes av skjebnen til kollektivet Fridtjof var en del av. Den faglige hackeridentiteten må møtes i et kompromiss som tilslører dette faktumet, noe som best beskrives som at jobben – først og fremst – skal være *morsom*, *lidenskapelig* og *engasjerende*. Dette bidrar til at personene som er talspersoner for hackerverdier gjerne fremstår som svært flinke, men samtidig ufarlige og utydelige. Det andre problemet er at hackerverdiene realiseres først og fremst på et moralsk vurderingsplan, og de knyttes i liten grad til produksjon, stabilitet og profitt. De formulerer ingen autoritative argumenter for å legitimere endring i organisasjonssammenheng. Moralske argumenter kan aldri måle seg mot mer reelle, målte verdier, som profitt og resultat, og harmonerer dessuten dårlig med objekter som planer, regnskap og budsjett. I forhold til de andre tre fagverdighetene, framstilles hackerlogikken ofte i forbindelse med det amatørmessige snarere enn det profesjonelle.

Generelt i utvalget mitt har hackeridealene sterke begrensninger utenfor miljøene som ikke spesifikt ønsker å assosiere seg med dem. Som jeg vil vise under blir hackeridealene på et generalisert plan et referansepunkt som mange informanter distanserer seg fra, motargumenterer eller latterliggjør. Lidenskap og engasjement blir noe man posisjonerer seg imot.

7.3 Ingeniørenes fagverdighet

Borghild er 32 år, og har jobbet i Beta siden hun ble ferdig utdannet. Gjennom hele karrieren har hun jobbet som prosjektleder, og hun er del av Betas ledergruppe. Dette medfører et helt annet syn på både kunnskap og hvilke egenskaper som er viktige hos en utvikler. Borghilds hovedansvar i Beta er å koordinere leveranser (av programvare) til Betas bedriftskunder. Dette innebærer å være tilgjengelig på e-post og telefon, følge opp utviklernes oppgavestatus og progresjon og holde oversikt over prosjektene hun er involvert i fra dag til dag. Borghild har master i informatikk fra et anerkjent universitet i England, så hun kjenner godt både systemtenkning og utviklersjargong. Likevel er hennes hverdag på jobben svært forskjellig fra arbeidsdagen til informantene jeg har presentert hittil. Hun mener at den viktigste kompetansen hun har i sin stilling er at hun har "*Evnen til å holde i mange tråder samtidig, og ha overordnet oversikt over alt som skjer, å være strukturert [...] og holde det man lover og være pliktoppfyllende.*" Borghild opplever at hun har gjort en god jobb når teamet hennes "*leverer på de milepælene som vi har sagt vi*

skal levere på, og overholder fristene".

For å lykkes med å koordinere og gjennomføre prosjekter, trenger man en plan og en arbeidspraksis som fungerer godt selv om hverdagen er uoversiktlig. Dette er antakelig grunnen til at mange av informantene mine mener at en god utvikler først og fremst trenger *faglig ekspertise*, *gode planer* og *gode rutiner* i hverdagen. Turkle og Papert kontrasterer hackernes "myke", eksperimentelle tilnærming til faget med en "hard" tilnærming, preget av en detaljert, rasjonell planlegging (Turkle 1984; Turkle og Papert 1990). Jeg sorterte idealer som fremhever verdien av struktur, plan under overskriften "Ingeniørenes fagverdighet" (se også Kraft 1979; Dijkstra 1993). Disse holder seg stort sett innenfor den logikken som Boltanski og Thévenot identifiserte som den industrielle verdiverden (Boltanski og Thévenot 2006:203).

Mange av informantene refererer først og fremst til idealer som assosieres med produksjon, industri og formell fagutdanning når de snakker om jobben sin. Samtidig trekker de grenser som distanserer dem fra hackeridealene. Grensetrekkinger som dette baserer seg hovedsakelig på *kulturelle* vurderinger om smak, oppførsel eller utdanning. Ifølge ingeniøridealene er man en god utvikler hvis man har profesjonell *kunnskap* eller har gode *arbeidsmetoder*, hvis man klarer å lage *solide løsninger.* Det er viktig å være *punktlig* og ha gode *rutiner* og *arbeidsplaner*. Formell utdanning er et krav. Ingeniøridealene uttrykkes også gjennom å trekke fram viktigheten av struktur og orden. Anne, som har jobbet i Alfa i tre år, trekker for eksempel fram viktigheten av å ha struktur for "*å komme igang med det man ikke skulle ha gjort, sånn at man ikke begynner dagen med å jage seg selv rundt i ring*". Tidligere jobbet hun med en sjef som ikke satte så strenge krav.

> *[For ett år siden] hadde jeg en sjef som hadde stor forståelse for at ting tok tid, men jeg lurte på om han egentlig var skuffet eller om han bare var grei eller hva han egentlig forventet. Det er litt annerledes nå med tightere prosjektstyring (Anne).*

I dagens situasjon får hun tettere oppfølging, og hun knytter dette til at både jobber mer effektivt på jobben og at hun kan slappe av bedre når hun går hjem om ettermiddagen.

7.3.1 Devaluering og redefinering av hackerlogikken

Mange av utviklerne som refererer til den industrielle fagverdigheten posisjonerer denne i direkte opposisjon til hackerverdigheten. Dette gjøres typisk på tre måter, enten ved å *devaluere* hackerverdigheten ved å referere til stereotyper, ved å kritisere hackeridealene og *redefinere* deres mening eller ved å utfordre deres *gyldighet* i en organisasjonsmessig kontekst. Amund er 28 år, har fem års ingeniørutdanning og har jobbet 2,5 år som programvareutvikler i bedriften Alfa. Denne

bedriften utvikler programvare som brukes for å lokalisere og finne oljeressurser. Da Amund skal posisjonere seg selv og utviklerne som jobber i Alfa innad i bransjen, distanserer han seg fra "*... sånne bleike .. heh .. tenåringsgutter, forvokste tenåringer*". Beskrivelsene er en slags parodisk beskrivelse av den stereotype nerden, som grenser mot latterliggjøring. I sin jobbutøvelse ønsker ikke Amund å forholde seg til hackere eller deres konsepter, og han tar dem ikke seriøst. Amund vil ikke assosieres med "bleke, forvokste tenåringsgutter" – disse har ikke "*faglig tyngde*", sier han. Amund forbinder hackeridealene med noe umodent og uforløst.

Evnen til å raskt sette seg inn i ny kunnskap ble verdsatt blant utviklerne som ofte refererte til hackeridealene. Mange i utvalget mitt satte imidlertid større pris på kunnskapen de tilegnet seg på universitetet. Arvid er en av disse, selv om han ellers har flere fellestrekk med gruppa i utvalget mitt som ofte viste til hackeridealer. Arvid er 32 år, og utvikler helst på Linux. Han fremstiller seg som datanerd fra ung alder, og hadde sine første erfaringer med Commodore 64, Amiga og dataspill. Arvid studerte hovedsakelig kjemi og matematikk på universitetet, og mangler dermed ingeniørutdanningen som mange av kollegaene hans har. Han anerkjenner at den *praktiske* utdanningen med program-øvinger ikke har så mye å si for det han holder på med i dag. Verktøyene han bruker i sin nåværende stilling (Linux, C++, QT) har han lært seg å bruke gjennom praksis i arbeidslivet. Arvid mener imidlertid at han har stor nytte i hverdagen av *metoder* og *tenkemåter* som han lærte på universitetet. Han trekker spesielt fram statistikk, matematisk modellering og generell problemløsning som viktig for hans nåværende stilling. I dag er Arvid scrum-master[29] i Alfa. Arvid trekker en skarp kulturell grense mot gruppa som dominerte i Alfa før Arvids tid – han omtaler dem som *ad hoc-utviklere*:

> *Når jeg begynte her var det mye ad hoc-utvikling og lite software-utvikling fra en software-utviklers ståsted. [Det var ...] litt sånn hacker-preget, at man dundrer sammen noe på en time, mye patching og forbedring til det verre og den typen ting. Så den store utfordringen for meg var å løfte software-utviklingen til et høyere nivå (Arvid).*

Arvid bestrider betydningen av ordet hacker når han sammenstiller ordet med ad hoc, og skaper en grense mellom hacker og (profesjonelle) software-utviklere – som han selv. For han er "hacker-preget" ensbetydende med en dårlig, midlertidig løsning. Gode utviklere lager derimot løsninger som er *varige*, og som bidrar til å *rasjonalisere* arbeidet for hele organisasjonen:

29 Scrummaster er en av de to lederrollene i Scrum (den andre er produkteier) og har ansvar for å fjerne "hindringer" for andre ansatte, og opprettholde jevn produksjon.

> *[...] hvis du skal lage et eller annet program for geologi-gjengen da, at du kanskje ser det litt i perspektiv, at det kanskje er noe du heller bør lage som en modul, som kan gjenbrukes igjen siden (Arvid).*

Distinksjonene mellom godt og dårlig blir trukket gjennom vurderinger av konkrete *løsninger*. Arvid mener at hans arbeid er mer verdifullt for organisasjonen enn ad hoc-utviklernes arbeid, fordi han har profesjonell kompetanse og fordi han tenker systematisk. Gode utviklere har ifølge disse idealene tilstrekkelig kunnskap og kan angripe problem på en måte som passer formålet, uten å eksperimentere. Dette er en kulturell distinksjon. Kunnskapen har han tilegnet seg fra universitetet og fra profesjonell erfaring. Ad hoc-utviklerne mangler disse egenskapene.

7.3.2 Viktigheten av å være på jobb mellom 8-17

Jevnt over er det mange som direkte eller indirekte utfordrer gyldigheten av hackeridealenes tidsforståelse i en organisasjonskontekst. De fleste utviklerne og lederne jeg intervjuet mener at åttetimersdag er det ideelle, og at folk bør være på jobb mellom 8-17. Dette kommer typisk til uttrykk gjennom at folk påpeker nødvendigheten av kjernetid og en fysisk arbeidsplass for å møte kunder, eller at de forholder seg til levering og henting i skole- eller i barnehage. De færreste er imidlertid spesielt engasjerte i dette spørsmålet, og de tar åttetimers jobb som en selvfølgelighet, og trekker ingen større grenser på grunnlag av problemstillinger som reises i forhold til tid. Anders er imidlertid engasjert, fordi han har erfaring fra en bedrift hvor størstedelen av arbeidsstokken sjelden var på jobb på dagtid.

Anders er 47 år, og fungerer som senior software engineer i Alfa. Han har 26 års erfaring fra bransjen. Han jobbet størsteparten av sin karriere som utvikler i en forskningsinstitusjon, men ble oppsagt etter en rasjonalisering av arbeidsstokken. Den norske IKT-sektoren hadde i en periode lave barrierer for nyetableringer og god tilgang til risikovillig kapital. Dette førte til etableringen av mange små bedrifter med unge arbeidstakere. Mange av disse bedriftene gikk imidlertid konkurs under den såkalte dot.com-boomen. Det var i denne perioden Anders var på leting etter nytt arbeid. Dette var en ganske turbulent periode blant norske IKT-bedrifter, med mange nystartede og unge firmaer med ansatte som ikke fulgte etablerte normer i forhold til arbeidstid, beslutningshierarki m.m. (Rasmussen og Johansen 2002). Etter en stund fikk Anders konsulentjobb nettopp i en ung bedrift av denne typen. Anders trekker sterke grenser i mot tidligere kolleger som hadde en arbeidsstil han anser som uprofesjonell og useriøs:

> *Folk kom kl. 12 midt på dagen og var til kl. 12 på kvelden, og jobbet utover natta og de syntes at det var greit. Jeg kom jo kl. 8 og dro kl. 16 jeg, og det var jo noen eldre der som gjorde det samme. Men det er veldig irriterende synes jeg, det å planlegge møter og at folk ikke kommer, og når det blir veldig vanskelig å få tak i folk fordi de aldri er tilstede og ... (Anders).*

Dette er en kulturell grensetrekking. Anders distanserer seg fra denne typen oppførsel, som han mener har uheldige konsekvenser for driften av organisasjonen. De ulike arbeidstidene gjorde kundekontakt vanskelig, og det ble vanskelig å avtale møter. Den sosiale omgangen på arbeidsplassen var gjerne også lagt utenfor standard arbeidstid. Kollegaene hans hang gjerne sammen på kveldstid. Anders sammenligner denne arbeidskulturen med en studentkultur:

> *[...] man skal være så ung og impulsiv i dagens IT-jobber. Jeg har hørt om bedrifter hvor [...] de sparker litt fotball også jobber de litt og har pizzabar og burgerbar inne i lokalet, og du skal trives og jobbe litt når du har lyst, for da er det mest produktivt og ... Det er mulig at det fungerer for unge folk, men jeg forbinder det med mer sånn student-type-jobbing da. Sånn som da jeg gikk på skolen, da var det mye mer denne typen jobbing. Da man skrev oppgave og holdt på og rota og gjorde oppgaven på kveldene og midt på natta og ... det ... Når man har unger som skal gå på skolen og når man er i fast arbeid, så har man veldig vanskelig for å få til de greiene der ... (Anders).*

Anders ser på seg selv som en profesjonell, og i hans øyne ligner denne umodne bedriftskulturen en uprofesjonell og ukvalifisert studentkultur. Potensialet i organisasjonen blir ikke godt nok utnyttet, og hensynet til kunderelasjoner og arbeidernes arbeidsvilkår blir ikke tilstrekkelig ivaretatt. Anders er svært fornøyd med at hans nåværende arbeidsgiver har en helt annen arbeidskultur enn den bedriften han jobbet i tidligere.

7.3.3 Bugs

I intervjuguiden hadde jeg et spørsmål om i hvor stor grad informantenes arbeidsdag var rutinepreget. Dette spørsmålet ledet for det meste til benektelser og forsikringer om at informantenes arbeidsdag ikke inneholdt noen rutiner. Bjarte uttalte "*Altså, klart vi har jo rutiner for å gjøre ting, men jeg tenker ikke på det som rutiner.*" De fleste forbinder rutiner med noe negativt og repeterende. Når jeg derimot spurte mer konkret om de anså det å finne- og luke ut "bugs"[30] som repetitivt arbeid, så var flere i større grad enige i at dette var både rutinepreget – og kjedelig.

30 I organisasjonene omtales feil i koden eller programmene som bugs.

I mange organisasjoner er det større QA-team som står for testing og påvising av bugs, men i Charlie er det utviklerne selv som gjør det. Når det nærmer seg release,[31] blir noen av utviklerne delt inn i grupper, og jobber hovedsakelig med arbeidsoppgaver knyttet til å finne og påvise bugs.

Casper ble introdusert i kapitlet over. Han omtaler seg selv som en kreativ supportperson, og har et nyansert bilde på utvikling hvor hackeridealer og ingeniøridealer eksisterer i et samspill. Selv om han ønsker mest mulig varierte arbeidsoppgaver i sin egen stilling, består en del av jobben hans i å motivere andre utviklere til å lage seg gode, effektive arbeidsrutiner. Dette er noe av det han ser på som det mest utfordrende ved sin egen jobb. Casper sier "*...man vil jo ikke gå rundt og mase på folk. Da blir folk bare sure, man må være kreativ, prøve å skape motivasjon hos folk på annet vis*". Casper anerkjenner at den kreative fasen i programmeringsarbeidet er langt morsommere enn å fikse bugs:

> *Teknisk sett så er det viktigst å være defensiv når man programmerer, når man skriver tester for programvaren man utvikler, så er det viktig å se på testresultatene hver eneste dag. Man må være grundig. Det er viktig å fikse bugs. Jeg skal nå snart begynne med en propagandakampanje hvor jeg skal få fram budskapet at jo raskere man finner bugs, jo raskere er de å fikse. Så for hver endring man gjør, bør man se på testresultatet dagen etter, det bør være rutine. Da ligger endringen ferskt i minne, "i går var jeg på denne raden, og gjorde denne endringen". Senere blir det vanskeligere å finne feilen (Casper).*

De mest kreative utviklerne har en tendens til å fokusere på andre deler av programmeringen enn testing og bugfiksing. Med gode rutiner og metoder kan utviklere imidlertid eliminere bugs før de blir til større problem, mener Casper. Selv om det ofte er morsommere å løse problemer og skrive nye kode, må en god utvikler være disiplinert nok og ha en profesjonell innstilling sånn at han/hun fjerner bugs på jevn basis.

I utvalget synes de fleste at det å fikse bugs er kjedelig, mens noen få mener det er uproblematisk. Arvid sier for eksempel at han har automatisert testprosedyren sin, så den utgjøre ikke noe problem. Cihan har derimot en tøff uke når jeg besøker Charlie. Han er ansatt som utvikler, ikke support – men denne uka bruker han ifølge egne beregninger 90 % av tiden sin på å motta og sjekke rapporter, verifisere dem og sende dem til riktig person.

7.3.4 Byrden ved å lede

Kraft (1979) fant at det hadde oppstått et skille hvor managerne i IKT-organisasjoner var

31 Release er ferdigstillelse av et produkt.

ingeniørutdannede, mens selve utviklingen hadde blitt overlatt til ufaglærte (deskilled) og repeterende. Vallas fant at symbolske grenser var viktige for å konstruere skiller mellom ingeniører og produksjonsarbeiderne i organisasjonen. Førstnevnte ble for eksempel distingvert som fleksible og innovative, mens sistnevnte ble fremstilt som "vanedyr" som var imot alle former for endring (Vallas 2001:23). Jeg finner ingen direkte tegn til symbolske privilegier eller eksklusjoner på basis av industriell fagverdighet, men forskjeller er selvfølgelig etablert på rollenivå mellom de som har lederansvar og de som ikke har det. Lederne har gjerne eget kontor og mer lønn. De jobber også i snitt 6-8 timer lenger enn resten av informantene mine.

Man skulle forvente at mellomledere i større grad enn andre brukte ingeniøridealer, og snakket om fornuften ved målstyring og metodologier, men dette fant jeg ikke. Det å lede grupper eller avdelinger blir i det store og hele sjelden omtalt med entusiasme eller beundring. Jeg intervjuet tilsammen fem ledere. Ingen i utvalget mitt uttrykker at de er *stolte* av at de er medlemmer i ledergruppa, mens mange derimot er stolte over at de *ikke* er medlemmer. Bjarte sier:

> *Jeg vet ikke. Å bli manager og springe å mase på... En ting er å programmere selv, men er det noe gøy å springe rundt og mase på at andre skal programmere da? Da har jeg heller lyst til å gjøre det selv (Bjarte).*

Lederskap er noe mange i utvalget mitt posisjonerer seg imot. Dina fikk tilbud om lederstilling, men ønsket det ikke. Alle med lederoppgaver nevnte at det i større eller mindre grad medførte stress. Bjørnar jobbet som leder i tre år, før han spurte om å få tilbake til sin opprinnelige stilling som i support i Beta.

> *... Tidligere i en treårsperiode så var jeg gruppeleder for support, det sluttet jeg med for ett år sida. Og da var det mere stress da, jeg synes det gikk litt utover personlig liv. Det var derfor jeg valgte å avslutte det, så da ble det litt mye, på en måte. Men sånn situasjonen er nå, så er det bra (Bjørnar).*

Det å lede ble av nesten alle de mannlige lederne jeg intervjuet ansett som en byrde. Som i tilfellene med Dagfinn og Birger, var de fem lederne jeg intervjuet i større grad opptatt av at grunnen til at man var leder ikke først og fremst var ingeniøregenskapene, men at ingen andre personer var kvalifiserte, eller at man hadde spesielle sosiale egenskaper (se mer i kapitlet under om hybrid fagverdighet). Produktsjef Christoffer sier "*Ja, jeg [programmerer litt i tillegg til å lede] for*

jeg er såpass interessert, jeg VIL ha litt møkk under negla, for å si det sånn da. Jeg synes det blir feil å være produktsjef for et produkt som jeg ikke kjenner til litt". Borghild, som synes å være tilfreds med sin egen lederposisjon i Beta bortsett fra at det kan være litt hektisk av og til, mente hun var på sitt mest inspirerte da hun jobbet med masteren i informatikk. Hun har likevel ikke gjort noe forsøk på å inkludere seg i mer teknisk arbeid i Beta.

En mulig forklaring på at lederne ikke trekker noen ingeniørfaglige grenser for å skille sin kunnskap fra andre gruppers, kan være at ledernes autoritet ikke er utfordret fra noe hold. Privilegiene og statusen som følger med lederstillingen er rett og slett ikke veldig attraktive i organisasjonene.

7.3.5 Reproduksjonen av ingeniørenes idealer

De industrielle idealene er utbredt i alle organisasjonene i utvalget mitt. Idealene blir legitimert spesielt ut fra kulturelle grensetrekkinger, som utdanning, formell kompetanse eller organisasjonserfaring. De er videre koblet sammen til mer eller mindre konkrete størrelser som produksjon, resultat, reelle prosesser og roller i organisasjonen. Informantene som henviser til idealer som beskriver rutiner og struktur er opptatt av at en god utvikler må følge en metode eller ha *faglig ekspertise* for å kunne løse oppgavene på en god måte. Industri-idealene blir ofte lansert i direkte opposisjon til idealene som romantiserer hackeridealene. Fagkunnskap blir regnet som verdifull, og kunnskap og arbeidsmetodikk preget av eksperimentering og fantasi blir devaluert. Argumenter fra industrilogikken kobles gjerne sammen med elementer fra en av de andre verdenene, for å styrke legitimitet. For å bli regnet som verdig ifølge en industriell grensetrekking, er det viktig å levere resultater og være produktiv.

Når utviklerne skal snakke om egen motivasjon for å begynne med faget, eller hva som motiverer dem, ligger imidlertid denne andre steder enn i ingeniøridealene. Motivasjonen finnes enten i de andre fagverdighetene, eller i eksterne motivasjoner som lønn eller frynsegoder. Sånn som utviklerne snakker om det, gir det ikke umiddelbart *status* å være flink til å følge rutiner eller god til å planlegge. En annen utfordring er at utviklere som oftest må benytte hackeridealer for å kunne snakke om innovasjon, nye ideer og det lystbetonte i arbeidet. Å referere ensidig til en industri, eller ingeniørlogikk truer dermed å gjøre IKT-arbeidere *ordinære*. Da forsvinner også kilden til profesjonell eksklusivitet og stolthet. De rene industrielle idealene ser dermed ikke ut til å gi noe sterkt grunnlag for å finne mening med arbeid *i seg selv*. I mine observasjoner ble de industrielle idealene i liten grad brukt som profesjonell identitetsmarkør for å understreke høy status, men en del utviklere som trekker tydelige grenser mot hackeridealene, nevner at de har seiling, golf og sykling som hobby. Dette ble ofte omtalt som høyverdige, sosiokulturelle markører,

og gjerne posisjonert som at man var annerledes enn den "typiske" programvareutvikleren.

De personene som realiserer legitimeringer jeg sorterer under dette kapitlet, er ikke nødvendigvis ingeniører av utdanning, men ofte er de det. Det de alle har til felles er at de legitimerer sitt eget arbeids verdi gjennom å snakke om viktigheten av formell utdanning, planer, disiplin, rutiner og struktur og ting som gjør ting mer effektivt.

Testen som reproduserer industriell verdighet tar først og fremst hensyn til målbare, konkrete størrelser – og holder seg stort sett innenfor det Boltanski og Thévenot kaller den *industrielle* verdiverdenen. Her er rutiner og planlegging trukket fram som sentrale virkemidler for å opprettholde produksjon. Så lenge firmaet går bra så vil utviklere kunne reprodusere ingeniørbaserte idealer ved å vise til resultater, at man har fått tilgang til en karrierestige m.m, at man lager effektive og systematiske løsninger som *virker* – selv om arbeidet ikke alltid er så morsomt.

Helhetlig sett bidrar industrielle grensetrekkinger til å danne en relativt robust identitet, som lett kan kobles opp mot industrielle måleredskaper som statistikk, regnskap og budsjett. Denne verdigheten kan reproduseres uten utbredt legitimering, og omtales som regel uten referanse til glede eller stolthet. Noen få informanter ser imidlertid ut til å finne en profesjonell stolthet i ingeniøridealer. Arvid er en av disse. Han finner stor glede og mening i å *rasjonalisere* løsninger som bedriften hadde før, og er glad fordi han har en arbeidsavtale som gir han autonomi til å selv finne problemer og rette opp "ad hoc"-løsninger innimellom andre arbeidsoppgaver. Det å kunne posisjonere seg som *bedre* og mer verdig enn "hackere" – eller "vanlige" utviklere ser ut til å kunne være et sentralt element for å kunne reprodusere en ingeniørverdighet.

7.4 Den hybride fagverdigheten

Tidligere forskning hevder at programvarearbeiderne, og spesielt gruppen som har blitt kalt hackere, i større grad enn andre personer er introverte, og at de ikke har lett for å kommunisere med kolleger eller brukere (Turkle 1984). Capretz, som har gjort en studie av IKT-arbeideres psykologiske profiler, hevder at mange utviklere mangler evne til å uttrykke hvordan en kodeendring vil kunne endre forutsetningene for brukerne, og at programmerere i større grad enn resten av samfunnspopulasjonen følger sin egen mageføelse snarere enn å lytte til andres meninger eller erfaringer (Capretz 2003). I relasjon til stereotypiske beskrivelser av programmerere som usosiale har mange foreslått at moderne utviklere trenger *hybride* egenskaper for å gjøre en god jobb. I tillegg til å kunne programmere, trenger IKT-arbeidere å kunne administrere egne prosjekter, de må være gode selgere og de må være gode

kommunikatører. Mer spesifikt innebærer dette at de gode utviklerne trenger nettverks- og kommunikasjonskompetanse (Dahlbom 1997; Friedman 1989; Goles, Hawk, og Kaiser 2008). Lagesen og Sørensen (2009) skriver at hybriditet innebærer at den sosiale kompetansen står i et gjensidighetsforhold til den tekniske kompetansen. Dette betyr i så fall er det er viktig å være god på begge områdene for å regnes som en god utvikler.

I intervjuene mine finner jeg at mange fremhever *kommunikasjonsegenskaper* og *team-spirit* som spesielt viktige for å være en god utvikler. Grensetrekkingene er ofte både kulturelle og moralske – de vurderer både oppførsel og personlige egenskaper. Det er viktig å ha *sosiale antenner,* og det er viktig å ha evne til å danne *nettverk*. Det regnes som høyverdig å *unngå konflikter*, å gi gode *tilbakemeldinger* og være *generøs*. Siden alle er en del av det store bildet, kollegaene, kundene, underleverandører, de er gjensidig avhengige av hverandre i den samme økonomien, vil det ifølge den hybride verdigheten gi avkastning å ha en god tone og dele. Det trekkes grenser mot medarbeidere som *monopoliserer* ressurser, *isolerer* seg eller *distanserer* seg. Det å ha høyverdige hybride idealer forbindes med suksess, og flere i utvalget mener at de har klart å gjøre seg karriere fordi de har innehatt både tekniske og sosiale (hybride) egenskaper. Hybride idealer forbindes også med å ha gruppelederkompetanse, og dermed privilegier i form av mer ansvar og høyere lønn. Kritikk med grunnlag i hybride idealer brukes for å kritisere uheldige arbeidskulturer eller distansere seg fra utviklere som regnes som for isolerte.

7.4.1 Sosiale antenner og evne til å danne nettverk

Christoffer er 30 år og har informatikkutdanning. Da han ble ansatt jobbet han med å utvikle en kommunikasjonsløsning til intern bruk i bedriften, men har nå en lederstilling i Charlie. Christoffer har også jobbet litt på supportavdelingen. Han beskriver arbeidet sitt på supportavdelingen som en fin blanding mellom programmering og kundekontakt:

> *[Jeg måtte] forstå problemet deres, komme med forslag om nye ting og sånn da. Så det har vært både kundefokusert eller ute hos kunder og ha presentasjoner og sånn, men også veldig hands-on-teknisk da (Christoffer).*

Christoffer reflekterer over at denne sosiale kompetansen også var veldig viktig i den første tida da han jobbet som utvikler, fordi han "*lett kunne prate med alle folka som faktisk brukte det systemet*". Christoffer forbinder denne kompetansen med å ha *sosialt anlegg*. Christoffer mener at det å prate med brukerne er noe som skiller gode fra mindre gode utviklere. På spørsmålet om hvordan han definerer en god utvikler, svarte han:

> *[Man må] ha en evne som det ikke er så mange programvareutviklere som har, liksom å se utafor sin boble da, og kunne tenke "hvordan er det kunden ser det her?". Og hvordan ser open source-miljøet på dette her? Og også kunne ha et litt objektivt syn på det vi gjør da, og også det å ha sosiale antenner da (Christoffer).*

Christoffer forbinder den sosiale kompetansen med det å ha oversikt i tillegg til å kunne kommunisere. Christoffer kaller det å "se utafor sin boble" og ha "sosiale antenner". Han definerer dette som å ha mer enn bare tekniske ferdigheter. Den sosiale kompetansen defineres dermed som noe som man ikke kan *lese* seg eller *eksperimentere* seg fram til. Christoffer snakker om introverte typer og ekstroverte typer, som også referer til scoren som det henvises til i Myers-Briggs-tester.[32] Han mener at mange utviklere er mer introverte, og at konsulentene gjerne er mer ekstroverte siden de er ute og besøker mange kunder.

Spørsmålet om sosial kompetanse er interessant i forhold til utdanning. Hackeridealene åpner for å likestille evnen til å lære og eksperimentere med formell kunnskap. Ingeniørens idealer blir koblet til metodologi og teknikker. Flere informanter trekker imidlertid fram at noe av det viktigste de fikk fra universitetsutdanningen var et *sosialt nettverk*. Da jeg spør Dina, som er en 31 år gammel systemutvikler om tilhørighet til organisasjonen, svarer hun ja, etter seks år føler hun tilknytning til Delta.

> *Men jeg har også holdt kontakt med studiemiljøet mitt, og jeg føler veldig stor tilhørighet med de gamle vennene fra faget også (Dina).*

Utdanningen har den effekten at den skaper kollegiale nettverk. Mange av informantene mine henvender seg fremdeles til venner fra studietida når det er noe de vil diskutere eller dele. Nettverkene brukes for å diskutere problemer og stille spørsmål, og for å skape en faglig tilhørighet eller profesjonell identitet. Gjennom informatikkutdanningen kan man få seg mer enn bare en utdanning – et godt nettverk og mulighet til å utvikle sin sosiale kompetanse. De hybride idealene kobler utdanning til det sosiale nettverket man får på kjøpet. Dette er i tråd med den *prosjektorienterte* verdenen, hvor de høyverdige subjekter klarer å knytte kontakter og skape seg nettverk.

Sosiale egenskaper omtales som nødvendig for å formidle og utfylle tekniske egenskaper. Christoffer, som legger mye verdighet i sin egen sosiale kompetanse, sier at han må jobbe mer

32 Myers-Briggs er en personlighetstest som i noen grad benyttes for å sette sammen team i IKT-miljøer. Se psykologiske tilnærminger til feltet (se 140).

med å oppdatere seg på det tekniske. Det at han fremdeles programmerer litt gjør det enklere å kommunisere med sine kolleger, mener han. Even, som jobber for en stor utenlandsk fri-programvareaktør, beskriver en dårlig utvikler på denne måten:

> *Du kan være en dårlig utvikler men allikevel ha mye teoretisk kunnskap. Men det jeg ofte opplever er at når noen kommer borti nye teknologier eller problemstillinger har de ikke kapasiteten selv til å lære seg det. De må bli fortalt hvordan ting fungerer. Det syns jeg er en dårlig egenskap. Samtidig, hvis man ikke klarer å dele den kunnskapen man har med andre, så er man også en dårlig programmerer. Man kan være en god programmerer, sitte for seg selv i stua og det kan funke helt fint, men jeg syns også som en god programmerer så innebærer det å kunne dele og vise andre hva du kan (Even).*

Den tekniske kompetansen er viktig, men det som skiller de gode utviklere fra de mindre gode er ikke bare om *evne* til å sette seg inn i teknologier, de må også kunne *kommunisere*. Å kommunisere og tilpasse seg er en naturlig relasjon i den *prosjektorienterte* verdenen.

Et godt, sosialt arbeidsmiljø er imidlertid ikke bare opp til de ansatte, Christoffer mener at bedriften har et ansvar for å skape et sosialt inkluderende arbeidsmiljø:

> *Jeg vet ikke hvor mange nasjoner vi bare har her i Oslo, men jeg tror det er 20 eller noe sånn. Og når du da får en ny person som kommer inn, f.eks. fra Finland, India eller Tyskland, så har ikke de noe sosialt nettverk. Og alle de som kjenner seg igjen i den situasjonen, de er flinke til å invitere folk og finne på ting, gå på ski, spille fotball, gå ut og ta en pils, eller gå på kino. Også blir det bare den ... syklusen der gjentar seg og gjentar seg og gjentar seg, også blir det bare et vanvittig godt sosialt miljø, både på jobb, men også utenfor jobb. Og [...] det på en måte nærer opp under det at folk kanskje utvikler seg litt sosialt (Christoffer).*

Charlie har på intervjutidspunktet i stor grad klart å skape en god og inkluderende kultur, ifølge Christoffer. Lederne bør også ha sosiale antenner, mener han videre. Han tror at hans egen hybride kompetanse var viktig for at han kunne få en lederstilling.

7.4.2 Territorialitet og egoisme

Det som regnes som lite verdig i tråd med de hybride idealene er det å jobbe alene, å isolere seg med arbeidet og være egoistisk. Cihan, introdusert over, bruker betegnelsen *team player* for å definere en god utvikler.

> *[A good developer] should be able to work in a team, he should be a team player. The problem with open source developers is that they are mainly individualistic. They work alone. Even though they work for open source software, they really can"t work as a team. Not everybody. There are people who have done open source work right from the start of computer history, and for hobbies, and that kind of thing, right. And for those people to work as a team – it's difficult. But if you come from a service organization, there is no individuals as such, there is only the team. If somebody mess up, they won't be pointing to one guy, it's always the team (Cihan).*

Cihan trekker en grense mellom de gode utviklerne og de som ikke kan å spille på lag. Alle er ikke like gode lagspillere. Det han regner som særdeles lite verdig oppførsel er utviklere som *holder igjen ressurser* som andre trenger, eller som til stadighet forsøker å fremheve egne idéer som bedriften eller teamet egentlig har nedprioritert. Cihan kaller fenomenet å drive med "dirty politics". Flere snakker om egoisme og det å monopolisere ressurser. Christoffer kaller fenomenet *territorialitet.* Sånn var det ofte i Charlie før, mente han:

> *Tilbake når jeg begynte, så var det veldig sånn at noen folk hadde sine egne prosjekter, de de jobbet veldig i siloer, de hadde sine egne ideer, og sine agendaer, som ofte ikke var så godt kjent. Og det var veldig mye krangling og munnbruk (Christoffer).*

Det å bare forsvare egne prosjekter fører til situasjoner som verken gavnet bedriftene eller den enkelte på sikt, mener Christoffer. Han syntes selv denne typen oppførsel er utrivelig og mener at det skaper negativitet og "dårlig karma" på arbeidsplassen. Cihan trekker grenser mot utviklere som skjermer seg i sin "egen sone og aldri vil bli forstyrret." Selv forsøker han å svare på alle henvendelser med en gang de kommer, han regner en *egoistisk innstilling,* som han kaller det, en naturlig konsekvens av konkurranse og rivalisering, som uverdig. Den hybride fagverdigheten inneholder dermed en *kollektiv* verdsetting, i betydningen at hensynet til resten av organisasjonen (og kunder) bør settes foran egne planer, strategier og ønsker.

Cihan forteller at generelt er kollegaene hans gode på å gi positiv feedback når han har gjort noe bra, og han forbinder det å være støttende i både medgang og motgang med et godt arbeidsmiljø. Det å "ri kjepphester" regner han som negativt. Anne sier at det er verdifullt å "*[...] klare å plassere seg selv, særlig i et sånt miljø som dette her, hvor det er mange som sitter. Å ikke bli sånn* "*dette er måten man skal gjøre det på, all annen arkitektur er tull*" (Anne). Det å kunne ha en

pragmatisk innstilling er verdifullt. Britt, som har master i informatikk, snakker om det samme.

> *"Han må være [...] åpen for endringsforslag. Uten at det går utover stolthet, at han er åpen for at han har gjort et feil valg, at et problem kan løses på flere måter. Alle er inneforstått med at dette er et team, og at vi skal trekke i samme retning (Britt)."*

Borghild distanserer seg fra utviklere som isolerer seg:

> *Selv om man som utvikler må samarbeide med andre, så er det jo ofte sånn at hvis man har et spesifikt problem å løse så er det kanskje litt sånn at man kan få litt tunnelsyn, og jobbe veldig intenst – alene med et problem – "jeg skal bare få dette til å fungere", også har det plutselig gått 8 timer hvis man ikke har noen å gå hjem til (Borghild).*

I stedet for å bli sittende med et problem selv, er det jo bare å spørre en kollega, mener Borghild. Hvis man ikke spør kolleger, risikerer man å bli stående fast og ikke komme seg videre. Idealene som brukes for å verdsette gode kolleger i betydningen at de er lagspillere trekkes på grunnlag av kulturelle kjennetegn. Folks oppførsel og manerer, spesielt i forhold til andre kolleger utgjør grunnlaget for vurdering. Her kobles også det å være sosial, åpen (for endringer) og ha et nettverk opp mot effektivitet, hvis man er sta, hvis man ønsker å jobbe alene, så risikerer man å sette seg fast. Å jobbe isolert eller alene kan ifølge den hybride logikken bli kontraproduktivt.

7.4.3 Oversiktskompetanse og karrieremuligheter

Spesielt i de små organisasjonene, Alfa og Delta, har noen av informantene mine fått noen muligheter gjennom at de har vært pragmatiske, vært gode til å kommunisere, takket ja til tilbud når de har fått sjansen – gjennom at de var tidlige ansatte. Dette gjelder i hvert fall Anne, Dagfinn, Dagny og Christoffer. Siden de ble ansatt såpass tidlig, har de fått fordeler gjennom at de vet hvem de fleste som ble ansatt etter dem er, og hvilke roller de er ment å fylle. Anne mener at hun har en oversiktskompetanse i Alfa på grunn av at hun ble ansatt mens bedriften var liten.

> *[...] det handler om at jeg sitter på en oversiktskompetanse [...]. Etter hvert [...] ser jeg at det er flere som synes det er artig å sitte å trekke i trådene (Anne).*

Anne har notert seg at flere gamle kolleger nå er interessert i å påta seg lederoppgaver. Selv er hun ikke så sikker på om hun vil det enda, men hun har for ikke lenge siden påtatt seg å

koordinere en del møtevirksomhet mellom ulike avdelinger i Alfa. Oversiktskompetansen hennes blir neppe dårligere av dette. Hybrid verdighet settes dermed i sammenheng med privilegier som å sitte i ledergrupper, høyere lønn, og i noen tilfeller enmannskontor. I mitt materiale er det spesielt denne oversiktskompetansen, ikke ingeniørkompetanse, som trekkes fram som viktig for å bli tilbudt lederposisjon.

> *[...] de anså meg som den eneste aktuelle kandidaten til den stillingen jeg har nå, fra ledelsens side ser jeg på det som en tillitserklæring (Dagfinn).*

Dette gjaldt også i Dagfinns tilfelle, han sa ja til lederjobben fordi det ikke var noen andre aktuelle kandidater som hadde den kunnskapen og den organisasjonserfaringen som han hadde. De fleste lederne i utvalget mitt nevner sine kommunikasjonsegenskaper som viktige for at de innehar stillingen de gjør i dag. Det å kommunisere gjør det enklere å bli kjent med nye folk. Christoffer tror som nevnt at det at han hadde både sosiale og tekniske ferdigheter ga ham mulighet til å bli leder.

Dagnys fortelling viser hvordan hun gjennom å investere tid og bruke kommunikasjonskompetanse ble tilbudt en sentral posisjon i Delta. Dagny er 34 år, og holder på å skifte stilling fra systemutvikler til markedssjef da jeg møter henne. Hun har master i informasjonsvitenskap,[33] og har jobbet i bedriften Delta siden 2003. På tross av at selskapet må regnes som lite, har Delta hele tiden vært en global aktør. Da Dagny var yngre reiste hun sammen med en kollega til USA i to år for å starte opp og drive et regionkontor og jobbe direkte hos kundene. Mens Dagny var i USA, ble arbeidsoppgavene hennes forandret. Hun forteller:

> *Der ble jobben mer prosjektrelatert, i forhold til de store kundene, det var derfor vi ble lokalisert akkurat i [denne byen]. Også [gjorde jeg] mindre utvikling, bare litt sånn tilpassing av produkt i forhold til kundene da. Også ... mere support, og mere sånn demo og ... salg og ... og [vi tok hånd om] alt innenfor Delta sitt felt [...] (Dagny).*

Dagny var også fungerende daglig leder på regionkontoret. Hun hadde bare jobbet i 1,5 år i Delta da hun reiste til USA, og store deler av den perioden hadde hun brukt til å lære seg å bruke et programmeringsspråk som hun ikke hadde brukt før, og som hun heller ikke fikk særlig bruk for i ettertid.

33 Informasjonsvitenskap er noe helt annet enn informatikk, og går mer på kontorautomasjon, webutvikling etc. o.l.

> *Det var jo en utfordring det å få en bunke med Delfi-bøker klasket på pulten, og "Dette må du lære deg nå, for det er dette programmeringsspråket vi bruker." Men, det gikk seg nå til. Må bare hive seg på, hehe (Dagny).*

Dagny så på det å lære seg et nytt kodespråk som en utfordring, og da hun ble spurt tok hun også i mot utfordringen om å reise til USA og lede et regionkontor der. Dagny trivdes kjempegodt med de varierte arbeidsoppgavene. Da hun kom tilbake til Norge var hun noen erfaringer rikere, og det virket ikke lenger så fristende å gå tilbake til utvikleravdelingen. Dagny beskriver at hun følte seg "*stuck*" som utvikler, og at hun ikke ville jobbe alene. "*Altså jeg har funnet ut av erfaring at jeg fungerer mye bedre til å jobbe i kontakt med kunder og prosjektstyring enn å sitte for meg selv og løse programmeringsoppgaver ... det var ikke noe for meg*." Dagny mener imidlertid at en god programvareutvikler bør *forstå* og *vite* hva de andre avdelingene og kundene driver med. Det var sånn det var, da Delta var ungt og lite. Etter hvert som Delta ble større, har utviklerne fått mer skjermede stillinger.

> *Vi har blitt et mye større firma enn vi var i 2003. Nå kan man møte utviklere her som aldri har vært på tjenestereise, men som har jobbet her i flere år. Det var helt utenkelig når jeg begynte. Det gikk bare et par måneder, så ble du sendt avgårde til våre største kunder rundt omkring i hele verden. Nå har de blitt mye mer organisert i forhold til arbeidstid og arbeidsoppgaver og de har mer spesialiserte kontrakter (Dagny).*

Dette har gjort at det er mindre mobilitet blant utviklerne, og oppgavene er mer ensidige. Dagny tror dette har gjort jobben vanskeligere, og mener at det er uheldig fordi det har gjort at R&D-ansatte har blitt mer "innesperret":

> *[...] vi begynner å bli såpass stor bedrift, at det er naturlig at R&D blir mer konsentrert, og en lukket del av firmaet, at de IKKE gjør sånn som jeg gjorde. At de er med på support og salg og demoer og alt det der. Når man mister den delen så mister de også den nødvendige forståelsen for "hvorfor" en programmeringsoppgave er viktig. (Dagny).*

Det tar lenger tid å bli kjent med arbeidsoppgavene, arbeidet blir mindre variert, man blir lettere isolert, og man mister både oversikten og erfaringene som var viktige for at Dagny kunne avansere til leder. Dagny synes utviklingen er uheldig, men ser veldig fram til å jobbe i sin nye rolle som sjef. Hun er den eneste i mitt utvalg som sier at hun er oppriktig glad for at hun ikke skal drive mer med utvikling.

7.4.4 Hybriditet og kjønn

Flere hevder at symbolske grensetrekkinger om evne til kommunikasjon og oversiktskompetanse i noen grad resultert i institusjonaliserte, reelle, sosiale forskjeller i organisasjoner. Det er opprettet stillinger som spesifikt etterspør personer med hybride egenskaper, med support og kundekontakt. Noen steder blir denne typen stillinger dominert av kvinner (Guerrier mfl. 2009). Innenfor kjønnsforskingen blir den store kvinneandelen gjerne forklart med forestillingen om at menn har tekniske evner, mens kvinner har sosiale evner (Faulkner 2000). I mitt utvalg finner jeg ingen symbolske forklaringer som tilsier at kvinner har en annen type kompetanse eller andre typer evner enn menn. Det er imidlertid sånn at fire av fem kvinner i utvalget mitt innen få år forventer å jobbe i en stilling som inneholder administrasjon, og som krever sosial kompetanse. To av dem, Dagny og Borghild, jobber i administrative posisjoner eller lederposisjoner allerede – Borghild på tross av at hun jobbet med en inspirerende masteroppgave i informatikk ved større utenlandsk universitet.

7.4.5 Reproduksjon av den hybride verdigheten

Den hybride fagverdigheten henter sin styrke gjennom å vise til verdien av teknisk innsikt, sosial kompetanse, nettverksbygging og verdigheten som ligger i å unngå konflikter. Alle er viktige forutsetninger for at arbeidet i store organisasjoner skal kunne koordineres og utføres effektivt og fredelig. Den hybride-verdigheten forbindes videre til argumenter om trivsel på arbeidsplassen, og mer effektiv produksjon. Denne fagverdigheten legger helt andre føringer enn hacker- og ingeniøridealene for hvordan gode utviklere bør investere ressursene sine i utdanningen og på arbeidsplassen. Det lønner seg å legge like mye i nettverksbygging og det sosiale som i å programmere eller produsere. Indirekte trekkes det noen grenser mot det som er høyverdig ifølge både hacker- og ingeniøridealene. Den hybride verdigheten består av et kompromiss hvor effektivitet (*den industrielle verdiverdenen*), konfliktskyhet, kommunikasjonsegenskaper (vs. egoisme og territorialitet), vilje til samarbeid (*den kollektive verdiverden*), og nettverkskompetanse (*den prosjektorienterte verdenen*) er representert. Utviklere som isolerer seg, skaper ineffektivitet, danner konflikter eller ikke har et stort nettverk har ifølge denne logikken lav verdighet.

Mange stabile sosiale kompromiss har sitt grunnlag i den kollektive og den industrielle verdenen. Det å *kjempe sammen* (for felles interesser) i en økonomi som er avhengig av *jevn produksjon*, er et klassisk kompromiss mellom det kollektive og det industrielle, spesielt synlig i for eksempel den europeiske fagbevegelsen. Sosiale velferdstjenester i større organisasjoner er også

et resultat av hensynet til kompromisset mellom disse verdiverdenene (Boltanski og Thévenot 2006:325–332). Den samtidige koblingen til å holde et høyt teknisk nivå, eller å ha nettverk- og kommunikasjonskompetanse, bidrar likevel til å skape et ustabilt kompromiss når man skal reprodusere meningen med selve programmeringen.

Utfordringen er, som med hackeridealene, at den profesjonelle identiteten som reproduseres blir et lite stabilt byggverk, som låner legitimitet litt her og litt der alt etter hvilke situasjoner og tester som oppstår. Dette gjør de hybride idealene flyktige i møte med andre fagidentiteter og andre typer retorikk. En hybrid logikk kan nok reprodusere seg i et noenlunde homogent utviklerteam som jobber målrettet med utvikling og som deler på andre oppgaver. Flere i utvalget mitt sier imidlertid at når de fungerer som en representant for bedriften eller jobber med administrative oppgaver, er det samtidig vanskelig å holde et teknisk høyt nivå. Fra et profesjonelt identitetsperspektiv er det et problem at de menneskene som regnes som gode og verdige raskt ender opp i posisjoner som ikke har noe med utvikling å gjøre, som for eksempel *prosjektleder* eller *gruppeleder*. Det å legge for mye vekt på en hybrid stolthet, at man er flink til å balansere de ulike kompetansene arbeidet består av, kan være et karrieremessig steg *vekk fra* utvikling. Dina sier at hun opplevde at mange kolleger forventet at hun skulle ta et gruppelederverv selv om hun ikke hadde lyst. Hun beskriver det som om hun følte at det var en plikt, flere i gruppa hadde lyst på stillingen.

> *Jeg har lenge følt at jeg måtte ta på meg et sånt administrasjonsansvar snart. De andre i gruppa trodde at jeg var interessert, og de var også interessert siden det medførte mer lønn. Jeg føler meg ikke helt klar for lederverv, jeg ønsker å kode fra meg først (Dina).*

Dinas skepsis er antakelig berettiget. Fortellingene til informantene mine antyder at det er mulig å gå fra å være utvikler til andre posisjoner, men vanskelig å gå den andre veien. Det var den sosiale kompetansen som fikk Christoffer og Dagny bort fra utvikling til å begynne med.

7.5 Den smidige fagverdigheten

I organisasjonene fant jeg en en fjerde definisjon av god, verdig utvikler, rangert etter hvor *enkle* løsninger han/hun lagde. Jeg har valgt å kalle idealene *smidige*, men de kunne kanskje like gjerne hett idealer for *forenkling* eller *smarte* løsninger.

> *En må kunne programmere ryddig. Må kunne forstå hvordan andre ser på koden. Det finnes noen som er dyktig å programmere; du får et bra produkt. Men det er umulig å*

> *forstå hvordan de har tenkt – det er ikke bra. Hos den beste programmeren går man inn i koden og det er [...] oversiktlig (Bjarte).*

En viktig funksjon for smidige idealer er å legitimere at ulike former for kompleksitet og støy i en organisasjon eller i arbeidet kan fjernes – sånn at utviklerne kan konsentrere seg og fokusere. Dette skaper rom for lengre tankerekker og det viktige tankearbeidet som ligger til grunn for å skape gode programmer. Ifølge de smidige idealene er man en god utvikler hvis man raskt klarer å *bryte ned* et komplisert problem, og hvis man klarer å *skjerme* seg lenge nok til at man kan gjøre seg ferdig.

Smidige idealer åpner dermed opp for en særegen måte å investere i arbeidet på, man skal investere "smart", bryte ned arbeidsprosessen i mindre biter, gjøre egne prioriteringer og fokusere fra oppgave til oppgave. I dette ligger det også en disiplinering, det lønner seg ikke å investere for mye tid i en enkeltoppgave. Det straffer seg å brenne for detaljene i det man jobber for, eller å henge seg opp. Det trekkes grenser mot utviklere som lager for komplekse løsninger, som fordyper seg og bruker for lang tid på å løse en oppgave, som "sprer seg for tynt utover", eller som har for mange jern i ilden i samtidig. De smidige idealene blir følgelig ofte posisjonert mot både hackeridealer, ingeniøridealer og hybrididealer – i en hektisk arbeidsdag har man ikke tid til eksperimentering eller leking, man har ikke tid til å legge detaljerte planer som kanskje ikke lar seg realisere, og man har ikke tid til å kommunisere med resten av organisasjonen. Noen av de industrielle idealene er likevel gyldige innunder smidig fagverdighet. Idealer som fremhever gyldigheten av produksjon, timeplaner og målinger danner et "smidig" mikrokompromiss, hvor det øverste ekvivalensprinsippet er forenkling, transparens eller oversiktlighet. De industrielle idealene som fremhever verdien av å realisere et hvert (menneskelig) potensial og følge enhver ambisjon, effektivitet som ekvivalensprinsipp, er derimot nedtonet (Boltanski og Thévenot 2006:206).

Kunnskap blir her noe som har verdi hvis den *anvendes* på riktig sted til riktig tidspunkt, i en *passende* dose. Praksiser som forbindes med smidige idealer er korte møter som gir mulighet til å løse problemer og eventuelt justere kursen. Rutiner har verdi, så lenge de bidrar til å gjøre arbeidet mindre komplisert. Privilegier som blir gitt de beste utviklerne i tråd med disse idealene er muligheten til å jobbe i fred, lukke døra og ikke bli forstyrret.

7.5.1 Begrensning

De beste løsningene er enkle, og det er viktig å vite hvordan man skal begrense seg. Craig, som jobber support i Charlie, har bestemte meninger om når en kodebit er *bra nok*.

> *Things I appreciate with developers are for example if they maintain a bit of code, that they are aware of making changes to it that changes the way [the program] behaves. And respecting the fact people might already be using the software that they want to change, and maybe it's not ideal, or they don't think their design isn"t clean enough, or that it's something they got wrong. And they would like to fix – the really good software engineers knows about this, and knows when to accept that they made a mistake, and that it will not really be fixed in the way they like, and the less good ones don't know when to leave it alone (Craig).*

Craig mener en god utvikler er kjennetegnet av at han/hun klarer å legge arbeidet fra seg, og leve med at det ikke perfekt. Ifølge Craigs definisjon må en god utvikler være *oppmerksom*, vite når han/hun skal *begrense seg* og ta hensyn til *brukeren*. Det er flere grunner til at det kan være viktig å begrense seg. Flere informanter som var nyutdannede like etter årtusenskiftet hadde gått arbeidsløse en stund før de fikk jobb, siden det var nedgangstider. Jeg samlet inn en del av disse dataene mens finanskrisen i 2008 utviklet seg, og noen informanter sa at tryggheten bekymret dem. Dagny sier for eksempel:

> *Nå har vi jo måttet tenke litt annerledes enn vi har gjort før da, for nå er det jo på en måte viktig at vi har en trygg jobb. Det har vi på en måte opplevd litt nå da, det har blitt sagt opp en del folk for kort tid siden. Så det har blitt en ny problemstilling, som i hvert fall ikke jeg har forholdt meg til før (Dagny).*

Jevn, stabil produksjon og mindre risiko er bedre enn høyest mulig produksjon, dette er en av idéene i smidig-litteraturen. Jevn produksjon er her gjerne bedre enn høy produksjon, fordi det er tryggere og mere statbilt.

7.5.2 Begripelig og forståelig

Flere i utvalget mitt er opptatt av at denne begrensningen er en aktiv holdning man kan ha til programmeringen, at man ikke trenger å legge så mye i det. Anders, introdusert på side 108 kaller det å "levere kvalitet". Med kvalitet forstår han:

> *[...] det å kunne løse oppgavene ... godt nok ... at man ikke graver seg for dypt ned i ting, men at man klarer å skjønne hva brukeren vil ha, og at man klarer å omsette det enklest mulig til noe som er brukbart (Anders).*

Dette referer til idealer som står som rake motsetninger til *hackeridealene*, som nettopp åpner opp for å *eksperimentere*, *utforske* og *leke*. Hvorfor holder det å lage løsninger som bare er *brukbare*, hvorfor kan ikke løsningene være perfekte eller i det minste så bra som mulig? Daniel jobber i Delta, og hans definisjon av en god utvikler gir et mulig svar på dette spørsmålet. Mange ser opp til ham på jobben på grunn av løsningene han kommer med, og lederen hans, Dagfinn, trekker fram Daniel som en særlig viktig ressurs på avdelingen. Daniel er også trebarnsfar, og har ikke tid til å utvikle noe utenfor jobbtiden. Daniels definisjon på å være en god utvikler er som følger:

> *Det er hvis jeg kan ta arbeidet hans videre, å jobbe videre med det uten å få mark – da er det godt arbeid. Jeg vil si at det er det viktigste, hvis han klarer å skrive begripelig kode som er forståelig (Daniel).*

Så lenge løsningen er *forståelig*, så kan andre hoppe inn og føre arbeidet videre. Da holder det også å utvikle løsninger som bare er *brukbare*, andre kan fortsette der den første slapp. Enkel kode blir da det samme som god kode. Av andre i utvalget mitt hadde jeg hørt at det var viktig å kommentere i koden, sånn at andre kunne ta arbeidet videre. Daniel var uenig i dette:

> *Hvis strukturen er god, så trenger man ikke kommentarer. Det har blitt sammenlignet med at.. kodekommentarer er litt som deodorant: Du slenger det på hvis koden stinker... Jo færre kommentarer som trengs for at koden skal bli forståelig, jo bedre er det (Daniel).*

Å skrive enkel og forståelig kode er alltid bedre enn å skrive komplisert kode med kommentarer, mener Daniel. Birger er gruppeleder, og han deler disse betraktningene om god utvikling.

> *Vi har folk som løser slike problem helt greit i vårt team, og folk som flyter ut og lager en veldig komplisert løsning. Som forsåvidt virker, men som kan være vanskelig å forstå for andre. Jeg mener at man bør lage et enklest mulig løsning på et problem, slik at det også er enkelt for andre å gå inn der senere. Og vi har jo selvfølgelig kodestandarder vi skal holde oss til. Så ... utviklere som holder seg til kodestandarder, og som klarer å lage en enkel løsning på et komplisert problem – det vil jeg si er en god utvikler (Birger).*

Birger tegner altså et skille mellom de gode utviklerne som lager enkle løsninger, og de som lager en komplisert løsning. Han knytter også de enkle løsningene til *standarder* (som hører hjemme i en industriell verdiverden) og at andre skal kunne gå inn og redigere koden senere.

Det å gjøre det enkelt kommer også til uttrykk som en del av arbeidskulturen. Arbeidsdagen for utviklere kan noen ganger være uforutsigbar, og det kan oppstå problemer som trenger raske endringer. Noen ganger er det problemer med en implementering som ikke går som planlagt, andre ganger kan det være problemer hos leverandører. Da blir det verdifullt å kunne jobbe både fort og bra nok for å kunne løse problemer som oppstår. Det er ikke alltid det er anledning til å bruke tid for å sette seg inn i et problem. Å kunne gjøre kjappe, enkle løsninger henger her sammen med både faglig dyktighet og teknisk ekspertise. Krisene som informantene mine forteller om skyldes ofte kunder eller kolleger som ikke klarer å levere til riktig tid. Borghild sliter med kommunikasjonen med salgsteamet: "*det er ofte at prosjektet blir satt igang altfor sent, og så skal man levere altfor tidlig, slik at vi som produserer får alt for lite tid på oss*". Daniel forteller at selgerne i Delta også kan være litt ivrige av og til:

> *Det hender at en eller annen idiot har solgt et produkt som som vi ikke har, så får vi beskjed om å lage det på to uker, og så får vi en eller annen hysterisk tlf. fra utlandet, som ringer masse rundt og sier "dere må fikse det, vi skal på luften om en halvtime" (Daniel).*

Å kunne reagere raskt på denne typen kriser, krever først og fremst vilje til smidighet og spontanitet. Moderne organisasjoner er ikke strømlinjeformede og velorganiserte og situasjoner og kriser kan oppstå. Man trenger muligheten for å kunne justere oppmerksomhet og fokus raskt. Da er enkle og gode løsninger, og smidige arbeidere som kan reagere raskt uvurderlige. Smidig verdighet skaper en forståelse av at forenkling er basert på både intuisjon og faglig kunnskap. Ifølge dette verdisynet kan *alle* bli gode, smidige utviklere, hvis de klarer å lage en så god løsning som mulig, gitt de begrensningene som til en hver tid finnes i en organisasjon.

7.5.3 Verdien av det enkle

Den smidige verdigheten fører til at alt som bidrar til forenkling tillegges høy verdi, og personer, relasjoner eller ting som bidrar til det motsatte rangeres lavere. Å måtte gå for en uoversiktlig eller innfløkt løsning regnes som svært ugunstig. Jeg spurte Daniel om han kunne fortelle om en gang han ble veldig imponert over en kollega, og han fortalte om en nylig hendelse hvor flere fra hans team, inkludert noen gruppeledere, måtte lage en ny protokoll på et av de gamle systemene sine. Daniel forteller:

> *Vi kom fram til en ganske innfløkt men åpenbart nødvendig optimalisering for å slippe å få masse data inn, eh, men da klarte en kollega å se et punkt med utviklingen som gjorde*

> *at vi kunne ignorere hele den komplekse tingen, og kun gjøre noen enkle modifikasjoner, uten at det skapte problemer for oss. Det synes jeg var godt gjort (Daniel).*

Dette sparte gruppa for mye arbeid, og fremfor alt ble løsningen enklere å forstå og endre på siden. Intuitive løsninger og geniale idéer er altså velkomne, det viktigste er at problemene blir løst, at de ikke bidrar til å øke kompleksiteten. Teamet kunne nå legge bort problemet, bli *ferdig*, og konsentrere seg om noe annet. Når Daniel forteller er det alltid det enkle og ryddige som utgjør distinksjonen, prinsippet på hva som er godt. Jeg spør ham om han kan fortelle om en episode han kan være stolt av:

> *En gang var vel, jeg brukte et programmeringsspråk som heter Erlang, [...] som var et veldig fascinerede språk. Jeg og en kollega her fikk veldig sansen for det, og gikk sammen om å lage en presentasjon for firmaet, eller for utviklerne i firmaet om det språket og hva man kan gjøre med det. Og etter det så skrev jeg om et av produktene våre i Erlang, og ... ja, det var det mange som syntes var interessant. [...] De syntes at det var veldig imponerende at [jeg] hadde klart å få det til på et par dager (Daniel).*

Den smidige verdiverden gir altså rom for å være kreativ, så lenge det er rom for det, og så lenge man gjør det ryddig og skikkelig. Eksemplet over er koblet til verdien av formidling i tillegg til at Daniel trekker fram at Erlang kunne brukes for å skrive om et av produktene svært raskt. Det enkle eller forenklende kan institusjonaliseres, og dette skaper en forbindelse til den industrielle verdiverden. Det å automatisere rutiner, bugfikser og testprosedyrer regnes som viktig arbeid i seg selv. Jeg skrev på side 108 at Arvid er opptatt av å lage solide løsninger. Når han har ledige stunder bruker han disse til å gå gjennom funksjoner og kodesnutter som brukes i de ulike programmene bedriften tilbyr, for å rydde og lage gjenbrukbare moduler. Det å forenkle og rydde i "rotet" støtter følgelig både en smidig verdighet og en ingeniørverdighet, hvor rasjonalisering er av det gode.

7.5.4 Skjerming

Mange i utvalget mitt mener at for å kunne fokusere sånn at man får til å skrive god, enkel kode, trenger man ro og konsentrasjon. Det finnes derfor ulike metoder for å skjerme de gode utviklerne fra andre relasjoner eller andre oppgaver. I Charlie er det for eksempel en uskreven regel som sier at hvis døren til et kontor er lukket, så skal man heller ikke banke på for å forstyrre. I Delta gir det å kvalifisere som en god utvikler det privilegiet at man kan skjerme seg.

Det finnes former for skjerming i alle organisasjonene jeg besøkte. I Alfa var det flere av utviklerne i utvalget mitt som lyttet på musikk i øreklokker for å å skjerme seg fra resten av miljøet når de programmerer. I skjermingen ligger det en eksklusivitet, utviklere får "lov" til å isolere seg, de får plass og tid til å fokusere. Kjerneutviklerne har rett til å skjerme seg og jobbe bak lukket dør, mens supportpersoner er forpliktet til å svare.

Skjermingen omtales som nødvendig for å klare å gjøre et godt arbeid, og er samtidig med på å skape en sosial grense mellom utviklerne og de andre som mangler denne typen privilegier. Flere dro også symbolske grenser mellom de som har "skjønt det" eller ikke - at de ikke skal forstyrre unødig hvis døra er lukket og hodetelefonene på. Denne kulturelle grensetrekkingen sier at de som forstyrrer bidrar til støy og kompleksitet. Daniel sier "*hvis jeg skriver mer enn jeg tenker, så er det egentlig et tegn på at jeg gjør dårlig arbeid*". Dette er grunnen til at Daniel av og til trenger å skjerme seg:

> *Vi har jo sånn småting med folk som kommer inn – det er jo sånn det er. Men hvis det blir veldig mye av det så er det jo... av og til, når jeg sitter inne i noe så er det litt som å prøve å holde et korthus med begge hender – så hvis jeg skal svare noe på noe, så *poff* så må jeg bygge det opp igjen etterpå (Daniel).*

Gode kolleger vet altså å ta hensyn til behovet for konsentrasjon sånn at man ikke avbryter, mener Daniel. Man kan dessuten bli avbrutt selv om man sitter med lukket kontordør. Sosiale medier blir av flere omtalt som en tidstyv, ikke et verktøy for formidling. Mens gruppen som refererte til hackeridealer brukte blogger til å presentere prestasjoner og få anerkjennelse, blir sosiale medier og for eksempel Facebook noe som skaper støy. Store mengder e-post kan også være et problem. Christoffer forteller at mange i Charlie mangler e-postkultur:

> *Et eksempel kan være en mail som blir sendt ut til alle, som starter en eller annen slags diskusjon – og folk har ikke e-post-kultur, sånn at de svarer til alt, også kommer [...] hele firmaet på svaret (Christoffer).*

kolleger kan bli satt på som mottakere uten at innholdet egentlig angår dem, eller i hvert fall uten at de fleste faktisk *trenger* å involvere seg. Dette stjeler verdifull tid, og bidrar til å sinke produksjon i tråd med industrielle idealer. Christoffer har lagt merke til at noen alltid har en tendens til å legge til flere mottakere for hver gang de svarer:

> *[...] hvis du har en e-posttråd som vokser og vokser og vokser, også tenker enkelte at "jeg må legge til han. Og han. Og han" - men de fjerner aldri noen. Eller de legger til noen bare for å eskalere, ikke sant, og det er [...] en uting (Christoffer).*

Christoffer trekker en grense mellom de utviklerne som skaper unødvendig støy og involverer flere enn nødvendig. Dette er en kulturell distansering, de uverdige har ikke lært seg eller tatt seg bryet med å lære seg korrekt oppførsel.

7.5.5 Smidig utvikling

De smidige idealene henter legitimitet fra en etter hvert omfattende metodikk-litteratur, som viser ulike måter for å skjerme, avgrense eller fokusere på å gjennomføre arbeid i arbeidsmiljø preget av endringer. Ulike typer smidig utvikling nevnes av informanter som en type ting som bidrar til å forenkle organisasjonslivet. De ulike variantene smidig utvikling bidrar til å organisere arbeidsdagen, porsjonere og dele opp arbeidsdagen i håndterlige størrelser. Alfa har for eksempel brukt den smidige utviklingsretningen Scrum de siste tre årene, og metodikken læres bort av to innleide konsulenter. Smidige metoder ble også delvis brukt i Beta og Charlie.

Anders delte sine erfaringer med organisasjonen han jobbet i før han jobbet i Alfa i kapitlet om ingeniørverdigheten. Alfa er på intervjutidspunktet en relativt stor, profesjonell bedrift – uten grenseløse tidskulturer blant utviklerne, ifølge Anders. Arbeidsgruppen som Anders er en del av jobber etter Scrum-prinsippene. I smidig utvikling er arbeidsflyten delt opp i sykluser, f.eks. fire uker, med tre uker avsatt til problemløsning og produksjon, og en uke for sluttføring og leveranse. Korte, daglige eller ukentlige møter sikrer rom for arbeidsflyt og progresjon, og litt av poenget er at de ikke oppleves som påtrengende eller forstyrrende.

> *[...] og sånne langtekkelige presentasjoner eller ansattmøter som tar en tre timer – det blir man bare sliten av, selv om man sitter helt i ro (Daniel).*

Disse korte møtene har kraft til å strukturere arbeidet *først og fremst* fordi de er korte og poengterte. Oppdragsgiver eller kunde tar del i mange av disse møtene, enten personlig eller i form av en stedfortreder. På spørsmål om hva som gir Anders lyst til å stå opp om morgenen og dra på jobb svarer han:

> *Nei, det er jo det at du føler at du er til nytte da. At man kan bidra litt med ... ting. Det er en veldig stor fordel her i forhold til stillingen min på forskningsinstitusjonen, her står*

> *man veldig mye nærmere kunden. Man ser nesten brukerne her. Så da får du mye mere lyst ... du får en helt annen feedback. Det er vanskelig å gjemme seg, så dette er på godt og vondt da, men på arbeidslysten er det veldig positivt (Anders).*

Anders trekker spesielt fram den nære kontakten til oppdragsgiveren og synligheten i forhold til resten av gruppa som en motivasjon. Hyppige møter gjør det hele sosialt mener han, og man får et kollektivt ansvar. En egen stilling, *scrummaster*, er satt av til å koordinere og fjerne eventuelle hindringer for å nå målene arbeidsgruppa har satt seg. I Alfa fylles denne stillingen av Arvid, men Anders er vikar når Arvid er borte.

> *Vi er et team som diskuterer oss fram til hvor mye ressurser som kreves for å løse en oppgave. Vi deler opp oppgaven i tasks. For eksempel, det å lage skjermbilde, det er en oppgave, og å hente dataen fra en database, det er en annen oppgave. Først så gir vi oppgavene poeng, også gir vi hver oppgave timer. Hvis vi da innser at oppgavene vil ta 50 timer, så må vi tilbake til kunden og si at nei, denne oppgaven er så stor at denne trenger vi mere timer på (Anders).*

Alle utviklerne har dermed ansvar for å estimere hvor lang utviklingstid produktet vil ha, i samråd med kunden. Det skal ikke brukes mer timer på en oppgave enn det som er planlagt, da må oppgaven diskuteres og planlegges på ny. Den stadige møtevirksomheten og forhåndsplanleggingen i Scrum gir etter Anders mening en god kontroll på hvor lang tid faktiske arbeidsprosesser tar, og bidrar ifølge ham til å minske risikoen for leveranseavvik.

> *Så det er en sånn gjensidig trygghet i det i forhold til det jeg har vært vant med fra før. I et gammeldags prosjekt kom man til en deadline, og måtte si til kunden at man ikke var ferdig. Når vi deler opp arbeidet i småbiter sånn som nå, så har vi mye mer oppfølging og kommunikasjon rundt framdriften (Anders).*

Smidig utvikling bidrar altså til å institusjonalisere en disiplinert arbeidsdag, som regulerer hvor mye tid og oppmerksomhet som skal vies hver enkel oppgave. Kontrastert med den umodne bedriften, er Anders svært fornøyd med denne organiseringen av arbeidet.

> *JF: Mm. Ja også kommer det et personlig spørsmål da ... hva gjør du i ferier og fritida di?*
>
> *I: Nei, da sykler jeg stort sett, hehe (Anders).*

JF: Du sykler? Trener?

I: Trener og sykler ja. Også har jeg jo en familie da, tre unger (Anders).

JF: Ja.

I: Så vi er mye på hytta og... jeg jobber i hvert fall i det hele tatt ikke med noe som har med data å gjøre. Både jeg og kjerringa er sånne ... også hun er sånn dataperson da, [...] du skal lete lenge etter steder hvor det er så lite dataprat, for det er vi i grunnen ferdig med [når vi kommer hjem]. [...] Når klokka er 16, når jeg drar herfra, så tenker jeg ikke et sekund på [jobben] før jeg kommer tilbake hit neste dag, så jeg er ikke en sånn fagidiot altså. Ikke i det hele tatt. Nærmest det motsatte, tror jeg. Jeg får liksom utløsning for det der gjennom jobben og da har jeg ikke noe behov for å drive å rote med det der på kveldene (Anders).

Scrum gir kanskje ikke Anders mer rom på arbeidsplassen, men det skaper rom sånn at han kan legge fra seg arbeidet på arbeidsplassen. Han kan gå hjem med god samvittighet hver ettermiddag. For Anders er ikke *innholdet* i arbeidet så viktig for hans eget selvbilde. Dette passer godt med smidig-litteraturen, og for øvrig også den hybride verdigheten slik denne fremstilles av Cihan og Christoffer på side 116 når de snakker om territorialitet. I både Scrum og XP blir verdien av individualitet og unik kompetanse tonet ned, det er teamet som er viktig. Det blir sett på som en styrke å kunne begrense individuelle kvaliteter og personlig ytelse – og dele arbeid mellom seg i gruppa. Alle bør ha omtrent den samme kompetansen, sånn at det ikke blir som i eksemplet som Borghild forteller om:

[...] hvis det er et område i koden hvor en utvikler har spesiell kompetanse eller ekspertise på, så kan det lett bli slik at han eller hun må fikse veldig mange like ting, og det kan nok oppleves som [en] stressfaktor (Borghild).

Eierskap til koden og spesialisering er en uting. Hvis en enkeltutvikler ikke klarer det planlagte arbeidet, skal det rapporteres på morgenmøtet dagen etter, sånn at tiltak kan gjøres eller flere tidsenheter kan allokeres. Stoltheten som ligger til grunn for en prestasjonskultur blir dermed tøylet. Utviklere belønnes for å estimere riktig antall timer eller poeng, og ikke være for ambisiøs eller påta seg for mye arbeid. Dette kan også kontrasteres i lys av hackerverdigheten og viktigheten av å *prestere* på jobben – Anders bruker den smidige verdigheten for å legitimere at han realiserer seg på andre felt i livet. Anders trener mye på sykkel, deltar i ritt, og bruker ellers

fritida si sammen med familien.

Even mener at det som motiverte han til å begynne å utvikle var utfordring, i tråd med hackeridealene. Når jeg spør ham om framtidsplaner og hva han ønsker seg av en eventuell framtidig arbeidsgiver så sier han:

> *Hvis jeg skulle bli fristet av et nytt tilbud [...] – eller beskrive hvordan en ideell arbeidsplass vil se ut, så må det være en teknisk interessant krevende oppgave da. Som man kan utvikle seg på. Som kan være utfordrende. Kanskje å jobbe sammen med folk som gjerne er dyktigere. [...] Og at det ikke er en for tung prosjektstyring. At du ikke føler at du må – det er forskjellige prosjektstyringer, så om det er Scrum eller... [...] jeg føler det hele tiden dukker opp nye golden hammers, hvordan man skal jobbe i prosjekter og hvordan man skal få ting best til å fungere. Og jeg hater sånn nye ting som pålegger en programmerer mer jobb enn det er verdt, da (Even).*

Even er motivert av utfordringen (hackerverdighet), og kritiserer smidig utvikling (ut fra et prinsipp om at det enkle er det beste) for å være for umotiverende fordi det i seg selv er noe som skaper kompleksitet, noe som innebærer mer jobb. Hvorfor lære seg en ny kompleks metodologi bare for å gjøre en jobb? For Even virker dette kontraproduktivt, selv om han har forståelse for at "*...man må ha en viss oversikt i store team*".

7.5.6 Den smidige fagverdighetens reproduksjon

Det er i Alfa jeg finner flest referanser til smidig metodologi siden de bruker Scrum i hele utviklingsavdelingen. Spesielt Anders er opptatt av å posisjonere Scrum imot hans tidligere erfaringer i en "umoden" og uprofesjonell organisasjon (side 108). Den umodne organisasjonen manglet struktur og orden. Anders trakk grenser med røtter i den industrielle verdiverdenen mot denne useriøse måten å arbeide på, da han snakket om viktigheten av å være på jobben mellom 8-17. Anders sin forståelse av hva som gir arbeidet mening i Alfa er derimot i større grad knyttet til smidige idealer. Anders foretrekker den smidige måten å strukturere arbeidet på, og liker det at han kan gjøre seg ferdig og dra hjem hver ettermiddag. De smidige idealene er imidlertid ikke forbeholdt arbeidere som jobber med smidig utvikling. Idealer som fremhever verdien av å forenkle, skjerme, fokusere eller gjøre forståelig finnes i alle organisasjonene. Daniel jobber ikke med smidig utvikling, men han fremmer verdien av å programmere ryddig og klart. De smidige idealene har en identitetsskapende kraft gjennom at de er så anvendelige og fleksible, faglig stolthet reproduseres hver gang man blir ferdig med en oppgave.

Den smidige legitimeringsformen henviser gjerne til objekter og subjekter fra den industrielle verdiverden, men det overordnende prinsippet er alltid å forenkle: Rutiner tillegges høy verdi, men bare så lenge de virkelig gjør arbeidet mer smidigere. Anders sier for eksempel at rutiner er nødvendige for å "*bryte ned arbeidsoppgavene, sånn at man hele tiden har en veldig kort horisont*". Dermed får oppgaven man holder på med der og da større plass. På samme måte regnes planer som helt nødvendige, men de må være så enkle og skisseaktige som mulig, sånn at de enkelt kan *justeres* underveis i et utviklingsforløp. Møter regnes som positivt og sosialt, bare de er korte nok og det ikke blir for mange av dem. Den smidige verdighetens styrke er at den kan brukes for å støtte opp under de andre fagverdighetene. Smidige idealer kan knyttes både til ingeniørverdighetens mål som produksjon og profitt, rasjonalisering og ferdigstillelse, og spille en komplementerende rolle for å koordinere handlinger og praksis i tråd med den hybride verdigheten, hvor kommunikasjon, nettverksbygging og det å kunne sjonglere flere baller gir høy verdi. Hackeridealene kan også i noen grad låne legitimitet fra den smidige verdigheten, gjennom at det å isolere seg og skjerme seg blir gjort til noe forløsende, de skaper *rommet* som trengs for å være kreativ. Den smidige verdigheten er disiplinerende på den måten at den ikke belønner personer som investerer mye. Hvis man fordyper seg, hvis man tar med arbeidet hjem, hvis man *gjør* arbeidet for viktig,[34] så innebærer det at man taper smidig verdighet. Den smidige verdigheten sikrer at det lønner seg å begrense seg, å holde det enkelt.

Dette tegner opp noen profesjonelle grenser. Kunnskapen som ligger til grunn for den smidige verdigheten kan være instinktiv, men faglig ekspertise, erfaring og metodikk er viktigere. De fagutdannede utviklerne kan jobbe raskere og mer metodisk enn ufaglærte, og de kan i større grad programmere "riktig" med hensyn til gjenbruksverdi og kodebiblioteker. De profesjonelle klarer oftere å begrense seg enn amatør-utviklere, og de klarer å gjennomføre prosjekter på en bedre måte, samtidig som de bruker mindre tid.

Den smidige verdigheten er først og fremst en test av smidighet, evne til å begrense seg og fokusere. Den smidige testens store styrke er at den utfordres svært ofte i komplekse organisasjoner. Testen realiseres hver gang man møter på et problem, eller hver gang man skal fullføre en oppgave (opp til flere ganger daglig i detaljstyrte prosjekt). Hvis man klarer å håndtere et problem uten å stoppe opp, eventuelt gjøre nødvendige prioriteringer, kontekstualiseringer eller delegeringer fortløpende, så vil testen reprodusere seg selv. Hvis det skulle oppstå et produksjonsproblem, som en utvikler ikke klarer å håndtere selv, så finnes det en egen rolle, scrummaster, som har som arbeidsoppgave å fjerne hindringer og legge til rette for at utvikleren

34 Å *gjøre* noe man brenner for mindre viktig: Denne formuleringen er lånt av Ingar Mehus, ISS. Den er god fordi den viser at mennesker i en viss grad kan velge hvilket trykk de skal legge på faglig stolthet, profesjonell identitet etc.

skal kunne klare å løse problemet. Institusjonaliserte tester i Alfa, som bidrar til å reprodusere denne smidige typen verdighet er det daglige møtet hvor man evaluerer gårsdagens arbeid, eller sprint-perioden som gjentas med noen ukers mellomrom, når arbeidet innenfor en syklus skal ferdigstilles og evalueres.

Smidig verdighet reproduseres når arbeidet flyter, når man får unna oppgavene innenfor de rammene man har til rådighet, og når man klarer å håndtere kriser og andre situasjoner som oppstår spontant. Privilegier som arbeidsro, det at man kan lukke kontordøra og slå av telefon og e-postprogram gjør det lettere å konsentrere seg om arbeidet og bli ferdig – som igjen bidrar til at smidig fagverdighet kan reproduseres. På sitt vis kan smidige idealer bidra til å skape rom for inspirasjon. Det smidige kan ha en indre, personlig forankring, koblet til internaliseringen av metoder eller en intuitiv sans for å forenkle. Denne indre forankringen er imidlertid truet av at arbeidet kan bli så enkelt at det ikke lenger gir utfordring eller følelsen av å ha oppnådd en prestasjon, eller at man støter på arbeidsoppgaver som ikke kan forenkles.

Hvis man gjør arbeidet for viktig, hvis man ikke klarer å begrense seg, da blir det vanskelig å gyldiggjøre smidighet. Den smidige testen er imidlertid ikke avhengig av indre motivasjon eller glede for å kunne reproduseres. Det viktigste er at arbeideren får unna arbeidet, uansett om det er behagelig eller ubehagelig. Testen av smidighet kan godt reprodusere seg og regnes som verdig ved hjelp av ytre insentiver som lønn, profitt eller produksjonsmål, hvis det skulle være nødvendig. Av og til må organisasjoner og utviklere gå for en *innfløkt* og *uoversiktlig* løsning. I organisasjoner som domineres av en smidig logikk vil ikke langsiktige planer utvikles, og de gode idéene vil kanskje ikke få tid til å modnes. Anders mener at Scrum, med sine stadige tilpasninger, kanskje har gjort at utviklingen i Alfa har blitt litt "vinglete".

8. Analyse #2 – den ukomplekse verdiverdenen

Hva var dette smidige kompromisset satt sammen av? Hvorfor var idealer om forenkling blitt så viktige i fagutøvelsen for programvareutviklere? I dette kapitlet ser jeg nærmere på bakgrunnen for at ukomplekse idealer har kunnet bli gyldiggjort i fagutøvelsen, og viser at idealene er så tydelige at de kan organiseres i en egen ukompleks verdiverden.

8.1 Management-litteratur 1960-1990

Kapitlet åpner med en kort gjennomgang av noen av trekkene ved Boltanski og Chiapellos skildringer av Management-litteratur 1960-1990. En interessant hypotese *I The New Spirit* (2005:69, 135) er at legitimeringer med rot i den hjemlige (domestic) sfæren ikke i mindre grad enn tidligere regnes som gyldige i organisasjoner. I managementlitteraturen som ble utgitt på 60-tallet kom dette til uttrykk gjennom grensetrekkinger mot "personlige vurderinger" av ansatte, og gjennom at resultatbaserte vurderinger ble løftet fram som mer rettferdige i en organisasjonskontekst. Denne management-litteraturen var skrevet hovedsakelig for managere, og rettet seg derfor mot å "frigjøre" managere gjennom å gjøre dem bevisste på hvilke typer vurderinger som ville regnes som gyldige i moderne organisasjoner. De foreslåtte vurderingssystemene kritiserte vurderinger og forfremmelser som ble gjort utelukkende på grunnlag av ansiennitet som belønner lojalitet (fra den hjemlige sfæren). Effektivitet ble løftet fram som et bedre og mer passende rangeringsprinsipp, ikke minst siden organisasjonene faktisk var nødt til å være effektive for å kunne konkurrere. 1960-korpuset kritiserte byråkratiet og hierarkiets rigiditet, men foreslo aldri å fjerne dette helt. Kritikken fremhevet heller forestillingen at hierarkiet måtte "mykes opp" ved å bli basert på meritter og ansvar. Vurderinger i organisasjoner kunne ifølge dette bare være legitime hvis de kunne knyttes til eksterne kriterier, målinger og kvalifikasjoner (Boltanski og Chiapello 2005:67-69).

90-tallets management-litteratur forsterket og utdypet kritikken mot hierarkiet og byråkratiet. Som koordineringsform ble hierarkiet nå regnet som utdatert, siden det baserte seg på dominans. Her skulle alle lønnstakere frigjøres, ikke bare managerne. Grunnene som ble oppgitt hadde ofte moralsk karakter. Skillet mellom de som måtte lede og de som ble ledet ble

fremstilt som kunstig og uverdig, og ikke minst lite effektivt i organisasjoner siden skillet ville skape forakt mellom de to gruppene ansatte (Aktouf 1996). En annen grunn var knyttet til sosial evolusjon, det moderne mennesket blir ikke motivert av å måtte følge ordre. I komplekse organisasjoner vil ikke de ansatte akseptere å motta ordre, og de overordnede verken vil eller kan i særlig grad gi ordre (Boltanski og Chiapello 2005:70-71). Drücker forklarte hierarkiets som utdatert på grunn av det høye utdanningsnivået i vesten - "alle" hadde nå blitt spesialister på hver sitt område. Derfor måtte organisasjoner bestå av likeverdige kolleger. Ingen kunnskap kunne bli rangert høyere enn den andre. I det moderne arbeidet blir mennesker vurdert etter i hvilken grad de bidro til å løse organisasjonens oppgaver (Drucker 1992).

I 90-talls-korpuset ble det overordnede prinsippet for å løse de "nye" utfordringene basert på *fleksibilitet*. Dette var det ultimate verktøyet for å tilpasse seg verden som et marked, preget av konkurranse og raske endringer. Fleksible, innovative organisasjoner kunne tilpasse seg alle endringer, være attraktive for arbeiderne med den beste kunnskapen og sikre seg den beste teknologien. En ny "trussel" ble også påkalt i 90-talls-litteraturen. Asia, med Japan i spissen, klarte plutselig å konkurrere mot amerikanerne på deres eget marked. Etter observasjoner og forskning på Japanske firma, spesielt Toyota og deres styringsmodeller, ble ulike metoder trukket fram som viktige for å møte dette nye behovet for billigere produksjon. Just-in-time, Total quality, Total Production Maintenance (TPM), og Kanban var noen av disse verktøyene, som ble omtalt som *slank* produksjon (lean production). Boltanski og Chiapello skriver ut fra forestillingene om lean production om det slanke firma (Boltanski and Chiapello 2005:70-75). Det moderne firmaet blir i management-litteraturen kjennetegnet av at hierarkinivå er fjernet, og at prosesser og oppgaver blir strømlinjeformete. Firmaet er omgitt av et konglomerat av leverandører, underleverandører, tjenesteleverandører og midlertidig personell, som gjør det mulig å regulere arbeidsstokken i henhold til produksjonen. Firmaet tar dermed form som et fleksibelt *nettverk* bestående av gjensidig avhengige knutepunkt. Arbeiderne skal organiseres i små team, siden dette gir større fleksibilitet, handlingsrom og autonomi. Den virkelige sjefen er kunden, noe som ikke minst skyldes at kunden ofte er en annen bedriftsaktør. Lederens rolle er endret til å være en koordinator med visjoner, i Frankrike representert gjennom ordet *manager*, satt opp imot 60-tallets *cadre*. I 90-tallskorpuset blir manager beskrevet som visjonær, kreativ, intuitiv, mens cadre er blir forbundet med kald, kalkulerende, administrativ rasjonalitet (Boltanski and Chiapello 2005:77). Manageren er fremstilt som "nettverksmennesket", og han/huns prinsipielle kvalitet er mobilitet, evne til å bevege seg rundt uten å bli hindret av grenser (boundaries), uansett om disse er geografiske, profesjonelle eller kulturelle, hierarkiske. Derfor er karisma og personlige egenskaper de viktigste verktøyene, sånn at manageren kan komme i kontakt med andre aktører,

som i seg selv kan være langt fra hverandre sosialt eller geografisk (Boltanski og Chiapello 2005:78–79). Andre termer blir også introduserte i 90-tallskorpuset, for eksempel coach og team-koordinator.

8.2 Utdaterte ingeniører og hackere – IKT-faget i endring

Endringene i IKT-faget er drevet av kritikker som har blitt rettet imot konsepter forbundet med epoken som kalles informasjonsalderen/kunnskapsarbeidet og den prosjektorienterte verdiverden. Før jeg analyserer håndbøkene, vil jeg vise litt av bakgrunnen for at denne kritikken har kunnet vokse fram.

Konseptene *ingeniør* og *hacker* er viktige historiske roller i datafagets historie. Det har lenge eksistert et dualistisk syn på programvareutvikling som enten drevet av *personlig lidenskap* eller av *rasjonell ingeniørvitenskap*. Dette dikotome skillet går langt tilbake i tid. Kraft beskrev for eksempel overgangen fra programvareutvikling som en fri og artistisk, men krevende og kompleks kompetanse på 50-tallet, til en *deskilled* og repeterende prosess som krevde verken kunstnerisk sans eller kompetanse på 70-tallet (Kraft 1979). For å gå fra kreativ, kompleks programvareutvikling til organisert samlebåndsarbeid, var det ifølge Kraft nødvendig å skille "hode" fra "hender": High-skill managere, ingeniørutdannet ved MiT, Harvard og lignende, designet systemene, sørget samtidig for at arbeidet ble strukturert og utført i tråd med organisasjonens behov. De hadde også ansvaret for å fordele arbeidet mellom de dårligere betalte utviklerne, som skule programmere systemene. Kraft beskrev utviklingen som en uheldig taylorisering og hierarkisering av et kunstnerisk og kreativt felt, drevet fram av behovet for å kutte lønnsutgifter og skape formaliserte modeller for utvikling. Dette skapte det Kraft forsto som en uheldig arbeidsdeling, og en degradering av selve programmeringsfaget til ikke-profesjonell status. Sherry Turkle (1984; Turkle og Papert 1990) viderefører Krafts konseptuelle skille gjennom å beskrive programvareutvikling som motivert av hard eller myk mestring – den harde fremgangsmåten til en ingeniør, eller den myke, eksperimentelle tilnærmingen hos en kunstner eller *bricoleur* (artist).

Informatikk-fagene omfavnet gjerne den harde framgangsmåten, som var en måte for å møte programvareutviklingens kompleksitet på. Begrepet *Software Engineering* ble først brukt i 1968 av F. L. Bauer ved Technological University of Munich, blant annet med det formål å skille den ingeniørmessige, plan- og metodebaserte komponenten ved programmering fra andre typer oppgaver vedrørende samtidens datasystemer og organisasjoner (Dijkstra 1993). Frederick Brooks Jr skrev i *No silver bullet* (1987):

> *The complexity of software is an essential property, not an accidental one. [...] The classic problems of developing software products derive from this essential complexity and its nonlinear increase with size.*

Det vil aldri finnes en enkel oppskrift som kan løse alle utfordringene som skyldes kompleksiteten i programvaresystemer, hevdet Brooks, fordi *kompleksitet* er en iboende egenskap i programvare. Dette gjør at store team bare klarer å produsere en relativt liten mengde kode sammenlignet med små team på 1-2 personer. Stadig flere maskiner og applikasjoner skulle virke sammen, og stadig flere samfunnsfunksjoner ble avhengig av programvare som interagerer med mennesker og andre programmer. For organisasjonene som utvikler programvare, ble det vanskeligere å håndtere denne kompleksiteten jo større systemet og utviklingsteamet ble. En måte å tilnærme seg programvareutvikling på var gjennom ingeniørmessig *struktur* og *planlegging*. Dette utgjorde utgangspunktet for de metodiske undervisningsoppleggene for systemutvikling i årene som fulgte. Plandrevne utviklingsmetoder er ulike, men de kjennetegnes gjerne av mye dokumentasjon. På mange måter lignet de på tradisjonelle ingeniørmetoder, som brukes når man skal sette opp en skyskraper eller en bro: Arbeidet planlegges på forhånd, og først når tegninger og materialvalg er ferdige og godkjent begynner byggefasen. Siden tegningene tar hensyn til eksisterende konstruksjonsteknikker, er det relativt enkelt å estimere et budsjett. Ofte omtales plandrevne metodologier i en idealisert form kalt *fossefallsmetoden*. Denne modelleres typisk i fem sekvensielle faser som følger hverandre som et fossefall:

1. Requirements – finne ut hva behovet er og hva som er de nødvendige ressursene
2. Design – designe løsningen
3. Implementation – utvikle og implementere løsningen
4. Verification – verifisere at løsningen virker tilfredsstillende
5. Maintenance – vedlikeholde systemet

Begrensningen i fossefallsmetoden er at man måtte gjøre ferdig en fase, før man kan gå videre. Planleggingsfasen (trinn 1 og 2) er helt skilt fra implementeringsfasen. Dette fungerer bra for tradisjonelle ingeniørprosjekter, hvis man skal bygge en bro eller et hus. Programvare er imidlertid en mindre håndfast konstruksjon enn et fysisk reisverk, med mange ukjente faktorer og kontekstuell endring gjennom rask, teknologisk utvikling. Undervisningsopplegget i informatikkutdanningene utover 90-tallet ble tilpasset nye teknologier og nye kodespråk, men

den strukturerte måten å tenke om problemanalyse, krav og planlegging forandret seg lite.

Hackerne, som fremstilte seg selv som en selvdrevet motkultur, organiserte derimot arbeidet på en *kaotisk* og *eksperimentell* måte: "*Bricoleurs construct theories by arranging and rearranging, by negotiating and renegotiating with a set of well-known materials*" (Turkle og Papert 1990:10). Stephen Levy (1984) var kanskje den som først prøvde å definere hacker og the hacker ethic i boken *Hackers: Heroes of the Computer Underground*. Levy definerte og tok utgangspunkt i prinsipper, som skisserte noen kriterier for *the hacker ethic*. Blant disse var prinsippene om informasjonsfrihet, og at man ikke skulle stole på autoriteter. Når man vurderte god eller dårlig hacking – var det bare kvaliteten som skulle telle – kjønn, alder, rase eller stilling/status hadde ingen betydning. De første hackerne som levde opp til disse prinsippene, var ifølge Levy pionerer med en oppslukende interesse for datateknologi og dennes muligheter. Mange av disse er profilerte innen IT-verdenen i dag, Bill Gates, Steve Jobs og Richard Stallmann ble dratt fram i *Hackers* som eksempler på gode hackere. Til felles har de at pionerene var unge, hvite menn fra middelklassen med en tilgang til tid på stormaskin fra den gang maskinvaren kostet mer enn huset den var plassert i. Graham vektlegger det kreative i sin framstilling av hackere, og sammenligner dem med malere, arkitekter og forfattere. Graham trekker en grense mellom hackere og programvareingeniører (Graham 2004:19):

> *Sometimes what the hackers do is called "software engineering" but this term is just misleading. Good software designers are no more engineers than architects are. The border between architecture and engineering is not sharply defined, but it's there. It falls between what and how, architects decide what to do, and engineers figure out how to do it.*

Hackeren er her forbundet med estetikk og kreativ, artistisk verdighet. Hacker-tilnærmingen er ofte også assosiert med såkalt fri programvare og åpen kildekode (Raymond 2001, 2003; Stallman 1999; Zeitlyn 2003; Himanen 2001). Lakhani & Wolf gjorde en studie av fri programvareutvikleres motivasjon, og tok utgangspunkt i at denne var betinget av utfordring og lek (2005:16):

> *Programming has been regarded as a pure production activity that is typified as requiring payments and career incentives to induce effort. We believe that this is a limited view. At least as applied to hackers on F/OSS projects, we should regard their*

> *activity as a form of joint production -consumption that provides a positive psychological outlet for the participants as well as useful output.*

Man kan spørre seg hvorvidt indre motivasjon og personlig engasjement motiverer programvareutviklere spesielt eller kunnskapsarbeidere generelt. Ulike psykologiske studier foreslår at programvareutviklere i større grad enn andre har introverte personligheter.[35] Det er utført diverse psykologiske studier av personlighetstyper innen programvareutviklerfaget (Bush og Schkade 1985; Smith 1989; Capretz 2003; Martinez et al. 2010). Med utgangspunkt i Myers-Briggs-typeindikatoren finner Capretz at det har foregått en endring av personlighetstyper etter

35 I det moderne arbeidslivet er det i stadig større grad vanlig å bruke ulike typer personlighetstester for å kunne vurdere ansatte utover kvalifikasjoner. Myers-Briggs, Keirsey og Big Five er tre vanlige såkalt projektive tester som brukes for å vurdere personlige egenskaper som ikke fremkommer på vitnemålet (Rothstein 2006; Hough 2000). Tester som dette tar utgangspunkt i aktørenes egne preferanser, og kan følgelig ikke si noe om verken kompetanse eller evne. Testene tar imidlertid mål av seg å si noe om arbeidernes komfort og handlingsmønster i ulike stillinger og organisasjonsroller. Myers-Briggs-typeindikatoren er en personlighetstest basert på spørreundersøkelser (Myers 1995). Metoden har som utgangspunkt at menneskers personlighet kan kategoriseres etter 16 personlighetstyper (Jung 1976), som henger sammen med måten man behandler inntrykk og tar avgjørelser. Typeindikatoren har også vært en inspirasjon for andre som har utviklet lignende og mer historisk-filosofisk anlagte tilnærminger til personlighetstyper, som for eksempel David Keirsey (1995, 1998). Fire dikotomier spiller en rolle i kategoriseringen (Myers 1998): *Ekstrovert/introvert:* Om man er orientert mot å ta kjappe avgjørelser på grunnlag av breddekunnskap, eller om man er orientert mot dybdekunnskap og refleksjon. *Sansing/Intuisjon:* Om man er praktisk anlagt og forstår hendelser pragmatisk, eller om man foretrekker mer komplekse forklaringsmodeller. *Tenkning/følelse:* Om man tar avgjørelser på et objektivt og analytisk grunnlag, eller om man i større grad vektlegger menneskelige eller personlige forhold. *Bedømming eller persepsjon:* Om man holder seg etter planen, eller om man er åpen for å endre mening hvis man får ny informasjon. Myers-Briggs-typeindikatoren har mange metodiske problemer. Jungs kategorier, som ligger til grunn for personlighetene, er for eksempel funnet ved hjelp av metoder som ikke lenger er i bruk i moderne psykologi. Det er også problemer med validiteten. Spesielt dikotomiene om sansing/intuisjon og tekning/følelse har lav validitet i forhold til andre målemetoder (Nowack 1996). Mens kultursosiologien er er opptatt av hvordan aktørenes handlingsrom formes i forhold til kontekst og kulturelle repertoar, overser personlighetstester av disse typene jeg har nevnt her alle kulturelle aspekt ved arbeid og organisasjon. Personlighet blir implisitt noe konsistent og uforanderlig. Målemetoden er heller ikke reliabel, i og med at den baserer seg på å splitte normalfordelinger i to motsatte kategorier – om man den ene gangen får 1 % overvekt mot ekstrovert, og den andre 1 % overvekt mot introvert, blir man kategorisert vidt forskjellig. Psykologiske kognitive kategoriseringer som Myers-Briggs og Big Five er likevel interessante, siden de har som utgangspunkt at mennesker i organisasjoner er forskjellige. Siden det moderne arbeidet utspiller seg i organisasjoner, er det å få en forståelse av denne ulikheten verdifull både for de ansatte og organisasjonens egen del. Myers-Briggs er fremdeles i praktisk bruk i dag på jobbsøkerkurs, lederkurs og til opplæring av ansatte, mens Big Five-tester brukes i økende grad for å sette sammen arbeidsgrupper blant programvareutviklere (Martinez et al. 2010).

at informatikkfaget har blitt vanligere, men at faget fremdeles domineres av såkalte *introverte* som ikke har lett for å kommunisere med brukerne. Capretz hevder at mange av utviklerne mangler evne til å uttrykke hvordan en endring i kode vil kunne endre forutsetningene for brukerne, og at de i større grad følger magefølelsen heller enn å høre på hva andre har gjort.

Utover 90-tallet ble dynamikken mellom ingeniøren og artisten beskrevet gjennom ulike teoriseringer av framveksten av det moderne informasjonssamfunnet. Kunnskap ble ansett som den nye kapitalen (Drucker 1992). Castells hevder at diffusjonen av IKT i organisasjoner har bidratt til å gi *makt* til arbeiderne, samtidig som repetitive oppgaver kan kodes, programmeres engang for alle, og fjernes (Castells 2009:257-8). Dermed gis også arbeiderne mer autonomi (Castells 2001:92). Implisitt blir arbeidet mer tilfredsstillende enn før, ved at det inneholder mer ansvar og mindre rutiner. Gjennom forestillingen om at kunnskapsarbeidere i større grad er motivert av en indre lidenskap, hevder Himanen og Castells at det realiseres en cyber-kommunitarisme; et fellesskap basert på anerkjennelse, samarbeid, inspirasjon og spontan nettverksdannelse i prosjekter (Himanen 2001; Castells and Himanen 2002). Himanens (2001:6-8) forståelse av hackere begrenser seg dermed ikke til programmerere, men definerer hacker som en hvilken som helst person som først og fremst motiveres av lidenskap i yrket. Etter denne definisjonen vil hacker kunne beskrive akademikere, kunstnere, og en lang rekke andre yrker fra grafisk design til diverse lederjobber. Himanen og Castells "The hacker ethic" beskriver en generell arbeidsetikk – på linje med Webers idealtype "the protestant ethic". Hos Weber ble arbeiderne i ytterste grad drevet av askese og pliktfølelse – hackeren motiveres først og fremst av at arbeidet er interessant, inspirerende og gøy. Den prosjektorienterte verdiverden alluderer til noen punkter til hackeretikken, spesielt gjennom idealene som henviser til indre lidenskap, aktivitet, nettverksdannelse og prosjektorientering (Himanen 2001:139–141). Himanen posisjonerer hackeretikken mot den protestantiske etikken som han mener fremdeles dominerer i Finland, som etter hans mening inneholder mål- og resultat-orientering. Himanen forklarte hvordan prosjektarbeid kunne forklares ut fra de gamle industri-idealene (Himanen 2001:125–126):

> *In the fast-paced competition of the information economy, modes of operation have to be dynamic. This leads to the organization of operations into projects, and these for their part require ever greater goal orientation and result accountability. This goes for both the main projects the whole enterprise has embarked on and for individual workers' engagement in their partial projects. Projects must have clear-cut goals and schedules, and their progress must be followed systematically. This becomes increasingly important*

> *when information professinals have more freedom to choose the times and locales of their work: goals and deadlines become essential determinants of the working relationship. These modes are also, gradually, becoming more prevalent in the ways states operate.*

Prosjektorientert arbeid kunne dermed ligne på de gamle industrielle modellene, som gjorde at arbeidet likevel ble regulert av klart definerte mål og tidsplaner.

8.2.1 Kritikker rettet mot IKT-faget

Kritikker med røtter i ulike verdiverdener har blitt rettet mot IKT-faget. I dette kapitlet tar jeg for meg kritikker fra den industrielle, den kollektive og den markedsorienterte verden. Den mest åpenbare kritikken er med referanse til en *industriell* verdiverden. Selv om programvareutviklingsbransjen var fleksibelt organisert, møtte programvareorganisasjoner mange utfordringer rundt årtusenskiftet. Den kritikken som oftest fikk gjenlyd i det generelle moderne samfunnet, var at IKT-prosjekter hadde en vedvarende tendens til å bli forsinket, dyrere enn planlagt eller kansellert før de ble gjort ferdige. Allerede på starten av 90-tallet ble to tredjedeler av alle IKT-prosjekter signifikant dyrere enn estimatene tilsa (Lederer og Prasad 1992). Ifølge en rapport, som er ofte sitert i 2000-2010-korpuset, ble bare 16 % av påbegynte IKT-prosjekter fullførte etter planen 1994 (Standish 1994). En annen studie viste at 40 prosent av IS (information services)-prosjekter ble avbrutt før de ble ferdige (Field 1997). Definisjonen på et suksessfylt IKT-prosjekt i disse rapportene brukte den *sekvensielle* forståelsen av IKT-prosjekt som har røtter i ingeniørfaget (den samme som Himanen refererte til i forrige sitat, prosjekter med klart definerte mål og planer), og som kan spores helt tilbake til 70-tallet: For at et prosjekt skulle regnes som vellykket, måtte det bli ferdig til *planlagt* tid, uten å overskride budsjettet, og med alle de *planlagte* egenskapene intakte.

Den splittede forståelsen av programmering enten som en artistisk kunstform motivert av personlig lidenskap, eller som en rasjonell, plandrevet ingeniørvitenskap, viste nå sine klare begrensninger. Sett fra *IKT-industriens* side kunne den (1) lidenskapelige, artistiske måten å arbeide med programvareutvikling på ikke bare kritiseres i form av at de gode ideene ikke kom jevnlig, men også ut fra at ideene som faktisk kom ofte ikke kunne omsettes til gode, salgbare produkter. Hackeren var ineffektiv og ustabil som arbeidstaker, og dessuten var det ikke nok av dem. Utover 90-tallet vokste IKT-industrien raskt, og organisasjonene som produserte programvare rekrutterte i stor grad folk som hadde fagutdanning, i form av informatikk- eller dataingeniørfag ved høyskole eller universitetet.[36]

36 Dette bidro blant annet til at lønna var gjennomgående høyere i IKT-industrien enn i sammenlignbare deler av

Den (2) planlagte og rasjonelle tilnærmingen hos ingeniørutdannede utviklere kunne imidlertid også kritiseres fra de samme industrielle idealene. Svært mange godt forberedte, nøye planlagte og påkostede IKT-prosjekter ble forsinket eller aldri fullført. IKT-feltet var på nittitallet i svært rask utvikling, og hvis man brukte for lang tid i planleggingsfasen, kunne man risikere at hele prosjektet ble lagt ned før det kunne bli ferdig. Forutsetningene som gjaldt da planleggingsfasen ble innledet, var ikke lenger gyldige da designet skulle implementeres. Nittitallets IKT-organisasjoner hadde dermed ikke tid til å følge fossefallsmetodikken hvis de ville hevde seg i konkurranse med andre teknologiselskap. Som de fleste forfatterne av 2000-2010-korpuset påpeker, så er programmering svært komplekst og derfor uforutsigbart av natur – det er ikke *mulig* å se for seg resultatet før produktet var nesten ferdig. Prosjektplanlegging blir fremstilt som vanskelig og uoversiktlig. Detaljerte planer fører ofte til store reisverk som senere måtte forkastes da de teknologiske og sosiale premissene som ble lagt ved prosjektstart ikke lenger var gyldige. En annen industriell utfordring for plandrevet utvikling oppsto gjennom at planleggerne i organisasjonene fikk et legitimitetsproblem. Utviklerne som skulle settes til å implementere et design var ofte yngre og hadde rikere kunnskap om emnet enn de som hadde utformet designet. Størrelsen på prosjektene og antall ansatte varierer gjerne fra prosjekt til prosjekt. Når arbeidet ble organisert i prosjekter, førte det til større gjennomtrekk av mennesker i organisasjonen. Det ble mer vanlig å leie inn konsulentressurser. Det ble større behov for rekruttering- og opplæringsarbeid.

Denne kritikken fra et industrielt ståsted – at verken hackeren eller ingeniøren leverte løsninger som var gode nok, og at organisasjonene ikke var stabile nok – ble understøttet av en annen type *kollektiv* kritikk. Sosiologer og fagorganisasjoner hevdet at de fleksible organisasjonene verken ga langsiktige muligheter eller jobbsikkerhet. Det moderne kunnskapsarbeidet, det å stadig måtte tilpasse seg og gå fra prosjekt til prosjekt, gjorde at fleksible arbeidere i større grad enn tidligere manglet materiell sikkerhet og psykologisk trygghet (Sennett 1998). Selv om fleksibilitet og autonomi preget det nye språket i organisasjoner, førte det til at programvareutviklere i praksis jobbet mer enn før, og ble eksponert for nye former for kontroll (Thompson og McHugh 2001). De ansatte slapp kanskje å forholde seg til en streng manager, men de måtte ta hensyn til en annen type kulturell styring i tråd med organisasjonenes offisielle identitetsuttrykk. De ansatte fikk ansvar for å overholde budsjett og tidsfrister for levering. Ofte betød dette at produktiviteten skulle øke fordi arbeiderne brukte en større andel av privatlivet til å jobbe (Rasmussen and Johansen 2005; Baldry et al. 2007). Dette førte til at kvaliteten på fritiden ble dårligere, og gikk utover tiden med barn og familie (Hochschild 2001;

arbeidslivet.

Perlow 1998). Dette fikk negative konsekvenser for karrierene til de som ikke ville eller kunne være permanent tilgjengelige (Watts 2009). Kunnskapsarbeidet skapte tidspress og problemer med å balansere tiden mellom arbeid og familieliv (Halrynjo 2007; van der Lippe 2007; Perrons mfl. 2007; Watts 2009). Gjennom nittitallet økte ulikheten mellom IKT-arbeidere i Silicon Valley signifikant (Benner 2002). Lignende trekk er dokumentert i en undersøkelse i Skottland (Hyman, Scholarios, og Baldry 2005). På tross av at fagorganisering blant programvareutviklere var lav, satte fagorganisasjoner fokus på problemer med individuelle lønnsforhandlinger, manglende HMS, lønnsavtaler uten pensjonsrettigheter, utbrenthet og stress.[37] Samlet sett var den *kollektive* kritikken uttrykt av arbeidere, fagorganisasjoner og samfunnsvitenskapen sterk, og den kunne oppsummeres som at kunnskapsarbeidere generelt ikke har blitt noe friere eller fått mer makt, snarere har den individuelle utvikleren i større grad tilpasset seg behovene til kapitalinteressene og fått mindre trygghet og dårligere livskvalitet.

En tredje kritikk kom i form av en styrketest knyttet til *markedsverdenen*: økonomisk kollaps. Dot.com-boomen, konkursene etterpå og det påfølgende fallet på NASDAQ viste for alvor at verdien i kunnskapsøkonomien ikke kunne tas for gitt. Prisen på produktene, på servicen og på organisasjonene var ikke lenger verdt å betale. Kritikken som ble rettet mot .com-modellen i ettertid påviste alvorlige feil med business-planer og ambisjoner på de tilgjengelige markedene (Lowenstein 2004; Goldfarb, Kirsch, og Miller 2006). I etterpåklokskapens navn så mange at de økonomiske vurderingene av IKT-industrien fremsto som mindre rasjonelle, og man trengte å knytte IKT-organisasjoner til mer håndfaste resultat. *Kvaliteten* på produktene, og på arbeidet, på måten å planlegge og organisere på, måtte heves betraktelig hvis IKT-bransjen igjen skulle bli levedyktig.

Kritikkene mot prosjektarbeid og prosjektbaserte organisasjoner er dermed formulert på basis av tre ulike verdiverdener, den *industrielle*, den *kollektive* og den *markedsorienterte*:

1. Verken hackere eller ingeniører hadde gode nok løsninger i en økonomi i rask utvikling.
2. Arbeidet hadde ikke blitt tryggere eller mer komfortabelt i IKT-industrien sammenlignet med tidligere organisasjonssystemer, snarere motsatt.
3. Rasjonaliteten i økonomiske beslutninger på IKT-feltet ble trukket i tvil, fordi de ofte manglet substans. Ustabilitet, økonomisk kollaps og konkurser gjorde at investorer holdt seg unna bransjen.

37 For en norsk variant av denne kritikken, se Bjerke (2004).

8.3 Noen sentrale trekk i 2000-2010-korpuset

Korpuset jeg har studert består av 12 håndbøker i smidig utvikling eller prosjekthåndtering. Smidig utvikling eller smidig prosjektutvikling (agile) er en paraplybetegnelse på metoder eller metodologier som anvendes innen programvareutvikling og tilgrensende områder. Smidige metoder begynte for alvor å vokse fram i konsulentmiljøer i USA fra midten av 90-tallet og framover. Smidig utvikling kalles *smidig* fordi metodologiene skal håndtere IKT-bransjens stadige behov for tilpasning og endring av både form og målsetninger i produksjonen (som også var en utfordring i 1990-korpuset). Det finnes flere ulike smidige metodologier, i denne analysen tar jeg for meg tre av dem. eXtreme Programming (XP) og Scrum er mest utbredt (både globalt og i Norge), og tilhører den opprinnelige første generasjonen av smidige metodeverktøy. Begge går tilbake til midten av 90-tallet. I tillegg har jeg med en bok om Lean Development som har som ambisjon å være et utviklet og helhetlig styringsverktøy for store kunnskapsorganisasjoner. Lean har sitt utgangspunkt i Toyota Production System (TPS) som ble utviklet mellom 1948-1975, men er her tilpasset programvareutviklingsfeltet. Metodologiene inneholder begreper, og beskriver ulike navn på stillinger og organisatoriske roller som skal distingvere de ulike metodikkene fra hverandre. De er opptatte av problemer som har med lederskap, prosjektkontroll og motivasjon av arbeiderne. Tre av bøkene er ikke direkte knyttet til smidig utvikling, men inneholder likevel sentrale partier med prosjekthåndtering i IKT-organisasjoner (Berkun 2005; Gyurky 2006) eller utviklingsmetodikk (Lui og Chan 2008).

Selv om det er forskjeller i metodologiene og i korpuset, så jeg også flere likheter i de underliggende verdiene og idealene som lå til grunn for metodologiene i bøkene, og det er disse jeg har satt sammen til den ukomplekse verdiverden.

8.3.1 Bøkenes hovedproblem

Smidig utvikling og prosjekthåndtering tar mål av seg å løse problemer med organisasjon og arbeid slik denne ble formulert på 90-tallet. Lønnsomhet, stabilitet og rasjonell produksjon av programvare er gjennomgående temaer. Et dominerende trekk i 1990-korpuset var at verden ble beskrevet som i konstant forandring. 2000-2010-korpuset forkaster ikke denne idéen, forestillingen blir heller forsterket. Hovedproblemet som tegnes opp for moderne IKT-organisasjoner er i den studerte litteraturen *uforutsigbarhet* og *kompleksitet*. Det er først og fremst dette begrepsparet som er til hinder for produktivitet, som er stressende for arbeiderne, og som gjør at tidligere management-strategier ikke er optimale. Sånn sett posisjonerer bøkene seg som et tillegg, som viderefører og raffinerer den *prosjektorienterte* forståelsen av verden. Alle de tre

kritikkene, at tidligere roller og metoder assosiert med IKT ikke lenger har gyldighet, at IKT-arbeidet ikke har vært trygt og komfortabelt og at lønnsomheten og forutsigbarhet må heves for å appellere til investorer behandles i større eller mindre grad i alle bøkene i korpuset. Hvordan man skal løse disse utfordringene trekker på løsninger fra ulike verdiverdener. *Industriell* effektivitet i produksjonen tillegges mye vekt gjennom metoder for å kontinuerlig justere og tilpasse produksjon ved hjelp av planer og beregninger. *Kollektive* idealer, som tar hensyn til at arbeiderne tross alt har menneskelige begrensninger er også tydelige, gjennom referanser til "humanity", "respect for people" og et imperativ om å aldri jobbe mer enn 8 timer i døgnet. Likeledes er det flere råd som fremhever det kollektive foran det individuelle, som i (Cohn 2009:408):

> *I'm looking at an old review form right now. It asks me to rate the employee on "the degree to which the individual effectively manages tasks within budget and timeline." how would you anticipate someone to behave in regard to this factor if it were something he was rated poorly on the last time? Would the person be responsive to coworker requests for assistance? Probably not. Individual assessment factors lead to individual-focused behavior. We want instead to encourage people to do what is most beneficial for the team and product.*

Investorene, som følger *markedets* logikk, blir tatt hensyn til gjennom lovnader om "høyere avkastning" hvis metodologien blir implementert riktig. Løsninger fra den *prosjektorienterte logikken* tillegges også mye vekt, først og fremst gjennom at arbeidsgruppene skal være små og selv-organiserende, at arbeiderne skal være tilpasningsdyktige, ta beslutninger sammen med *kundene* og at organisasjoner og prosesser helst skal være så slanke som mulige. Nødvendigheten av fleksibilitet og å tilpasse seg er løftet fram nettopp gjennom begrepet *smidighet* (agility). Tilpasningsdyktighet (adaptibility) er en forutsetning for å kunne investere i tråd med den prosjektorienterte logikken.

I tekstene finnes det imidlertid også et tydelig formulert femte ekvivalensprinsipp – med utgangspunkt i ideene om det *enkle* og *ukomplekse*. Dette blir formulert på denne måten i Berkun (2005:2–3):

> *The simpler your view of what you do, the more power and focus you will have in doing it. If we can periodically maintain a simple view of our work, we can find useful comparisons to other ways to make things that exist all around us. There will be more examples and lessons from history and modern industries that can be pulled from, compared with, and*

> *contrasted against. This is similar to the concept defined by the Japanese word shoshin, which means beginner's mind, or open mind, an essential part of many martial arts disciplines. Staying curious and open is what makes growth possible, and it requires practice to maintain that mindset.*

Her er det enkle knyttet til åpenhet og vekst. Samtidig innebærer forenkling at noe skal fjernes (Beck og Andres 2004:19):

> *Improving communication helps achieve simplicity by eliminating unneeded or deferrable requirements from today's concerns. Achieving simplicity gives you that much less to communicate about.*

Jeg vil i resten av kapitlet vise hvordan prinsippet om forenkling brukes for å beskrive de ulike rollene og praksisene i bøkene, men også hvordan det utgjør et møtepunkt hvor de andre idealene kan møtes i et kompromiss og bidra til å skissere en ny moral og legitimere en "ny" type disiplinering i fagutøvelsen.

8.3.2 Rollebeskrivelser

Organisatoriske roller i 2000-2010-korpuset: antall treff	
customer	2100
programmer	1927
director	1719
user	1487
product owner	929
leader	729
scrummaster	314
coach	215
administrator	147

Tabell 10

Customer og *programmer* er de to mest brukte begrepene i bøkene, mens *user* kommer på en fjerdeplass. I scrum har begrepene *product owner* og *scrummaster* langt på vei erstattet manageren og coachen. *Director*, som er en samlekategori for ulike ledernavn i Tropes, kommer på tredjeplass. Når man tar dette i betraktning, er det noenlunde de samme ledersubjektene – eksperter, managere og ledere – som går igjen i 90-talls- og 2000-2010-korpusene. Rollene er definert på litt

ulike måter. Eksperter og ledere i den prosjektorienterte verdiverdenen har høy verdi fordi de er et knutepunkt i nettverket og fordi de gjerne har visjoner. Eksperter og ingeniører i den industrielle verdenen har høy verdi fordi de behersker målesystemer, og besitter stillinger ut fra formelle kriterier. Dette er egenskaper som forutsetter en lengre utdanning, eller en god metodologi i bunnen. Ukomplekse ledere blir fremstilt som gode fordi de klarer å omsette komplekse kunnskaper til enkle budskap som *samler* og *nærer* fram en god arbeidskultur samtidig som de *fjerner* hindringer, og har god *innsikt* i arbeidet som hver enkelt utfører.

Ledere med visjoner og selvorganiserende arbeidere er viktige også i 2000-2010-korpuset, men i tillegg trenger de å ha *oversikt, gi rammer* (boundaries), kunne fordype seg i *detaljer* og kunne noen *konkrete verktøy*. 1990-korpuset klarte ifølge Boltanski og Chiapello aldri å løse paradokset med at firmaet aldri kunne bli 100 % del av et nettverk, og verken arbeiderne eller organisasjonene kunne aldri bli 100 % fleksible. Ledernes *visjoner* ble løsningen som gjorde at ledere kunne garantere arbeidernes engasjement uten å måtte ty til tvang, visjoner som gjorde alles arbeid meningsfullt (Boltanski og Chiapello 2005:75–76). I 2000-2010-korpuset er ikke lenger visjoner nok, på langt nær. Møter, metoder og verktøy for å styre prosjekter blir introdusert. I smidig utvikling foregår møtene ofte, gjerne på daglig basis, men de holdes korte. Lederne er sterkt involverte, og viktigheten av visjoner blir nedtonet:

> *The vision shows what the product can become. The boundaries describe the realities within which the vision must be realized (Cohn 2009:126).*

> *Yet attentiveness to detail is an even more critical foundation of professionalism than is any grand vision. First, it is through practice in the small that professionals gain proficiency and trust for practice in the large. Second, the smallest bit of sloppy construction, of the door that does not close tightly or the slightly crooked tile on the floor, or even the messy desk, completely dispels the charm of the larger whole (Martin 2008:xix).*

I Scrum er begrepet *leder* ikke lenger brukt når det gjelder daglig drift, men lederoppgaver er fordelt mellom to nye roller, scrummaster og produkteier (product owner). Scrummasteren har som oppgave å "gjete" teamet (ikke lede) i riktig retning, fjerne hindringer og passe på at bevegelsen hele tiden er effektiv mot målet. Produkteierens oppgave er å definere de riktige målene, peke ut en retning, og definere rammene målene skal nås innenfor. I den grad visjoner fremdeles er viktige, skal disse komme fra produkteieren. I 1990-tallskorpuset skulle visjonene

motivere teamet til å "lede seg selv". I 2000-2010-korpuset skal de ansatte fortsatt styre seg selv, i den forstand at de i størst mulig grad får velge sine egne oppgaver og roller, men samtidig er det også tydelig at det også kreves et aktivt initiativ fra lederne. Man kan ikke alltid stole på at et team klarer å organisere seg selv (Poppendieck and Poppendieck 2009:184-185):

> *In this frame, line managers have a good knowledge of the work they manage and are deeply involved in understanding how work gets done. [...] they have a keen sense of the important issues and are willing to delve into the details.*

Dermed er det ledernes oppgave å hjelpe arbeiderne til å se problemene, hjelpe til å løse dem og spre kunnskapen om hvordan dette gjøres i hele organisasjonen. I tillegg brukes metaforer fra botanikkens verden. Flere fremstiller at det å nære (cultivate eller grow) fram en god arbeidskultur er en lederoppgave. Lederansvar og lederengasjement er tydeliggjort. Verbet gro (grow) har en dobbelhet. For det første representer dette en fornuftig tilnærming. Man begynner i det små og øker verdien, som i Lui og Chan (2008:14) "*Grow, don't build software* " eller Jeffries et al. (2000:47) "*Write functional tests as programs. Better yet, make a little scripting language that the programmers can use. Then grow it and make it easier until the customers can use it.*" For det andre representerer det som vokser noe truende. Prosjekter, systemer og relasjoner kan gro vilt. Disse trenger *trimming*. Det blir i mange tilfeller fremstilt som bedre å fjerne *mer* enn å fjerne mindre (Pichler 2010: 56): "*Simplify, prune, and strive for order—like a gardener pulling out the weeds and trimming the shrubs.*" Det å publisere et produkt som er uferdig, som krever videre tilpasning blir en strategi i seg selv (Pichler 2010: 56-57):

> *When in doubt, exclude a requirement from the release and ship quickly without it—just as Google did when the company developed the first release of Google News, an application that aggregates news from around the world. The development team could not agree whether to filter the news by date or by location. So Google decided to release the new product without either feature. Shortly after the product's launch, requests for new features started to come in. Three hundred people requested filtering by date, while only three wanted to filter by location—a clear indication of which functionality should take priority. If Google had released the product with both features, the release would have consumed more time and money and it would have been harder to get feedback on which feature was more important. By putting out an intentionally insufficient product, Google quickly discovered what to do next.*

1990-korpuset var kunden tydelig, og det er han/hun i stor grad i 2000-2010-korpuset. En rolle som ikke har vært i fokus tidligere er *brukeren,* som ikke er identisk med kunden. En kunde er oppdragsgiver og betaler for produktet (en annen bedrift, en organisasjon, en offentlig etat eller en kommune), mens brukerne er de menneskene som faktisk bruker produktet, som reagerer hvis programmet er lite responsivt eller lite brukervennlig. Hvis man vet hva brukerne ønsker, før man gjør en endring, så øker det sjansen for at et produkt blir godt mottatt. Brukerne får dermed ansvaret for å peke ut veien videre (som i eksempelet over), og representerer ellers et subjekt som legger sterke føringer for hvordan det er lurt å tenke rundt prosjektplanlegging.

8.4 Korrigerende og radikal kritikk rettet konkret mot den prosjektorienterte verdenen

Korpuset retter hovedsakelig to overordnete kritikker mot den prosjektorienterte verdensforståelsens gyldighet i IKT-organisasjoner. En korrigerende kritikk, som holder seg innenfor den prosjektorienterte verdiverden, og en radikal kritikk, som henviser til det ukomplekse ekvivalensprinsippet. Den korrigerende kritikken hevder at selve begrepet *prosjekt* ikke er fleksibelt og tilpasningsdyktig nok. Kriteriet for suksess, regnet som at et prosjekt blir fullført til planlagt tid, uten forsinkelser eller kostnadsendringer, blir ugyldiggjort. Alle metodebøkene i utvalget mitt posisjonerer seg i større eller mindre grad imot at prosjekt er den beste eller eneste måten å organisere arbeidet på. Samtidig forkastes ideen om prosjektet med en definert start og slutt: man kan aldri vite, eller planlegge nøyaktig målet i et IKT-prosjekt, og derfor kan man heller ikke vite nøyaktig hvilken sti man skal følge for å nå målet. På samme tid lanseres en radikal kritikk med henvisning til det ukomplekse ekvivalensprinsippet. For å organisere det komplekse arbeidet på en best mulig måte trenger man ikke større nettverk, flere prosjekter, eller ledere som har høy prosjektorientert verdighet (som ville innebære ytterligere kompleksitet), man trenger først og fremst oversikt, og evne til å fokusere. For å kunne *tilpasse* (verb i den prosjektorienterte verdenen) og *rasjonalisere* (verb industriverdenen) i en kompleks organisasjon med komplekst arbeid må en del arbeidsprosesser utføres *kontinuerlig*, og det må eksistere *rammer* som gjør at arbeiderne kan *konsentrere* seg.

8.5 Forenklingens verdioffentlighet (polity of simplicity)

Jeg vil nå redegjøre mer spesifikt for hva den ukomplekse verdiverdenen består av. Den første gangen jeg la merke til idealene jeg kaller "ukomplekse" var under et intervju med Craig, som jobbet som support i Charlie. Da jeg spurte ham om hvordan han definerte en god utvikler, svarte

han:

> *From my perspective I'm happy with people when they have written things that work, and when it's easy to understand and explain. [...] I think I tend to emphasize the qualities the other people don't look at. Things I appreciate with developers are for example if they maintain a bit of code, that they are aware of making changes to it that changes the way [the program] behaves. And respecting the fact people might already be using the software that they want to change, and maybe it's not ideal, or they don't think their design isn"t clean enough, or that it's something they got wrong. And they would like to fix – the really good software engineers knows about this, and knows when to accept that they made a mistake, and that it will not really be fixed in the way they like, and the less good ones don't know when to leave it alone (Craig).*

Denne definisjonen lot seg ikke kategorisere så enkelt i utgangspunktet, og jeg begynte å lete etter andre definisjoner og grensetrekkinger som fremhevet det å gjøre seg ferdig, å legge ting bort, å la ting være som de er. Sitatet inneholder essensen i den ukomplekse verdiverdenen. Her nevnes både en utfordring (endring), ekvivalensprinsipp (lett å forstå og forklare), verdighetstilstand/state of worthiness (oppmerksom), et subjekt (personer som bruker programvare), en beskrivelse av en god og en dårlig programvareutvikler, og definisjonen av en naturlig, ukompleks handling (begrense seg). Craig nevnte for øvrig ikke smidig metodologi ellers i intervjuet.

8.5.1 Siddhatta Gotama

Jeg hadde lenge en følelse av at den ukomplekse verdenen hadde røtter til østlig filosofi. Den ukomplekse moralfilosofiske verdioffentligheten kan beskrives gjennom seks aksiomer[38] ut fra kildene som beskrev livet til Siddhatta Gotama, som er kjent gjennom myten om Buddha. Kildene er samlet i *Buddha* av Karen Armstrong (2001), jeg har ikke hatt tilgang til originalverker.

Buddhas oppvåkning (enlightenment) er beskrevet som at han klarte å disiplinere sine egne indre lyster og behov gjennom meditasjon, og dette åpnet for at han kunne være bevisst på "alt". Buddha kombinerte *jhāna* (transe) med *appamana* (former for grenseløshet), og oppnådde fire hierarkiske (a4) meditasjonstilstander (a2) hvor han kultiverte og mestret sine følelser i en så stor grad at han til slutt oppnådde total sinnsro, balanse og fokus (a6). I denne tilstanden kunne han

38 For å få forklaring på hvordan en polity/offentlighet skiller seg fra verdiverden og hva de seks aksiomene betyr se side 41 eller Boltanski og Thévenot (2006:74–79).

være bevisst alt annet, samtidig som han ikke kjente verken tiltrekning eller antipatier mot noe eller noen. Dette innebar at Buddha kunne nøytralisere "*the power of that egotism that limits human potential*" (Armstrong 2001:70-72). Gjennom sinnsro, å disiplinere seg (a3), gjennom og også fjerne alle indre, egoistiske motivasjoner (a5) og unngå å vurdere hvordan andre ting og mennesker kunne være til nytte eller ulempe for seg selv, oppnådde Buddha storhet gjennom å søke det gode for alle andre mennesker gjennom klarhet og oversikt (a1). Denne tilstanden var videre oppnåelig for alle gjennom øvelse, selv om det ikke var enkelt.

Når disse seks aksiomene organiseres etter hverandre ser det slik ut: Det menneskelige potensialet for klarhet og oversikt (1) kan oppnås gjennom konsentrasjon og oppmerksomhet (a2), man kan befri seg fra indre lyster gjennom disiplinering og fokus (a3) i et hierarki av ulike nivåer (a4). Gjennom å ofre alle indre, egoistiske motivasjoner (a5) kan alle oppnå balanse (a6) og fokus.

Så, hvis Buddha hadde vært en programvareutvikler hadde han ikke vært motivert av indre lidenskap, men av at han gjennom konsentrasjon, fokus og disiplin mestret og frigjorde seg fra fra sine indre drivkrefter, behov og lyster. På denne måten kunne han være en god programmerer samtidig som han var oppmerksom ovenfor det beste for alle (brukere), på linje med Craigs innledende sitat. Hva motivasjon består av er dermed endret i forhold hackeretikken med indre lidenskap, men ansvaret for å utvikle, kultivere og mestre motivasjonen er likevel en *bevisst* og *personlig* øvelse – basert på indre disiplinering, ikke indre lyst.

I *Rethinking Cultural Comparative Sociology* beskrev Lamont, Moody og Lafaye *the green worth*, som i noen grad gjenspeiler verdier jeg har funnet i den ukomplekse logikken (Thévenot, Moody, og Lafaye 2000:256–263). Her er for eksempel *harmony* (with nature) og *sustainability* viktige, men det felles beste – the common good – gjelder ikke bare mennesker, men hele planetens økologi. The green worth er ellers nært knyttet til idéen om naturen, men sånn som jeg beskriver den ukomplekse logikken er denne formulert til et felles beste for menneskeheten. Antakelig vil det være mulig å påpeke ukomplekse idealer i tekster som for eksempel omhandler naturvern, men dette spørsmålet behandler jeg ikke nærmere.

8.6 Verdiverdenen satt opp som et grammatisk system

I det følgende skal jeg skissere grammatikken bak den ukomplekse verdiverden, sånn som jeg har funnet denne i 2000-2010-korpuset.

Den ukomplekse verdiverdenen brukes ofte for å verdsette mennesker og direkte menneskelige relasjoner framfor

teknologiske, siden førstnevnte er *naturlige* og dermed *enklere.* Høy verdi tillegges mennesker som fatter seg i korthet, og som klarer å bryte ned komplekse sammenhenger til enkle "sannheter", eller teknologier som gjør livet mindre komplisert. Logikken kan som regel forstås relasjonelt, *enklere* eller *mindre komplisert* enn noe annet. Det enkle er også et mål i seg selv hvis ting kan gjøres *selvforklarende*. Alt som vitner om at noe er ute av *balanse* eller *ute av kontroll* blir en test. Hvis det er periodevis mye overtidsjobbing, anses for eksempel dette som et seriøst problem. Overtid skaper ubalanse i flyt og produksjon, men også for arbeiderne på det personlige planet. I et ideelt arbeidsmiljø skal en observator kunne komme rett inn og iløpet av 15 sekunder forstå hvordan gruppa er organisert og om arbeidet flyter godt. Mange relasjoner er formet i opposisjon til ideen om nettverk som noe godt: Det at produksjonen overføres og spres fra et team til et annet, regnes som unødig kompliserende. Den ukomplekse logikken åpner for en legitim mulighet til å isolere, stenge alt annet ute, inkludert helheten og detaljer – sånn at man kan fokusere på nåtiden. Å ha oppmerksomheten på det nære trekkes fram som essensielt for å kunne eksistere på sikt.

Enkelhet, klarhet (HIGHER COMMON PRINCIPLE) perspektiv

(Overordnet ekvivalensprinsipp for å sammenligne verdighet mot/med andre verdiverdener)

Det overordnete prinsippet i den ukomplekse verdenen er *enkelhet* og *klarhet*. Prinsippet brukes for å vurdere om subjekter (ledere/arbeidere) lager gode løsninger, om de er gode til å formidle, eller om en prosess eller en metode er bedre enn en annen, eller om et design eller et produkt er bedre enn et annet.

Prinsippet brukes for å veie organisasjonspraksiser mot hverandre; "det er enklere å *snakke* sammen enn å skrive dokumentasjon". I tråd med dette skal kommunikasjon og organisering alltid være så enkel som konteksten tillater. Prosesser, design, funksjoner og romløsninger har verdi hvis de er selvforklarende, oversiktlige, tilgjengelige og hvis de gir *perspektiv*. Grenser og rammer regnes som noe positivt, så lenge de er tydelige, transparente og mulige å manipulere. Det enkle idealet

favoriserer små steg foran lange. "*What's the least you could do that is recognizably in the right direction?*" er et prinsipp i Beck & Andres (2004:33). Forståelsene som blir brukt om arbeidsprosessen er derfor først og fremst kortsiktige.

Tilgjengelig, smidig (STATE OF WORTHINESS) Forståelig, oppmerksom, klar, elegant, ryddig, frikoblet, redusert, fjernet

(Hva er det som gir folk/ting verdi?)

Det verdige er det som er enkelt i sin natur, det som er *tilgjengelig* eller *smidig*, det som først og fremst er enkelt å formidle, forstå eller huske. Å skape noe enkelt krever *oppmerksomhet* og *bevissthet*. Pedagogiske hjelpemidler som forenkler, og teknikker for å formidle komplisert stoff eller gjøre prioriteringer eller kontekstualiseringer kvalifiserer som verdige. Ordet *eleganse* knytter enkelhet til det som er brukbart og det som er fungerende. Kode og dokumentasjon har verdi hvis de først og fremst er leselige. Personer, ting og arbeidsprosesser som er *frikoblet* regnes som verdifulle. Å være *smidig*, som er en forutsetning for å kunne investere i nettverkslogikken, er en egen verdig tilstand i den ukomplekse verdiverden. Å være smidig brukes i denne betydningen om å klare å tilpasse seg de endringer som uansett kommer, uten å stresse. Når noe reduseres blir det ifølge den ukomplekse logikken enklere, klarere eller bedre. "*Individuals in this environment work at a slower pace, but this slightly reduced pace is more than offset by the time that would otherwise be wasted going to and from meetings*" (Gyurky 2006:127).

En viktig figur i en smidig organisasjon er *produksjonssyklusen*, som typisk har varighet på to uker eller mer. Slutten på syklusen utgjør prosjekthorisonten. Funksjoner som skal implementeres senere, eller som man ikke rakk å implementere denne gangen, forskyves til den andre siden av prosjekthorisonten. Dette bidrar til *ryddighet* og at det blir mindre å konsentrere seg om i inneværende syklus, og dermed blir nåtiden både mer komfortabel og produktiv.

Kompleksitet, kaos (STATE OF UNWORTHINESS) Motstand, distraksjoner, forstyrrelser, sabotasje,

Ting som er mer komplisert enn de kunne vært er regnet som lite verdige. Dette omfatter at alle praksiser i en IKT-organisasjon som er tunge og ressurskrevende, som for eksempel å skrive dokumentasjon eller sekvensielle planer og oppskrifter kan måles

– og anses som lite verdifulle. "*Heavy documents in a small-sized project distract the team's focus from the working software product*" (Lui og Chan 2008:60). Personlig stil og individuell *frihet* er en trussel, siden den kan føre til kompleksitet og mangel på oversikt. "*Idiosyncratic coding styles and the values revealed by them, individual freedom at all costs, don't help the team succeed*" (Beck og Andres 2004:17).

byråkrati, forsinkelser, frihet, konflikt, tung, overlesset

(Hva er det som gjør folk/ting uverdige?)

Å dele oppmerksomheten, eller å gjøre for mange ting på en gang er uverdig. *"Multitasking—attempting to work on two projects or two things at once—is one of the biggest drains on project team performance. Yet it has unfortunately become one of the busy manager's most frequently used tools"* (Cohn 2009:191-192). En "gyldig" forenkling må aldri innebære at ting i effekt faktisk blir mer komplisert. Derfor kan man heller ikke innføre enkelhet gjennom *policies* eller *byråkratiske* modeller som i sin natur medfører kompleksitet og uoversiktlighet. Stadige vurderinger (performance reviews) av arbeidernes ytelse omtales som negativt gjennom at det *forstyrrer* flyten, men også fordi det individuelle fokuset kan gi uforutsigbare *ringvirkninger* som påvirker andre deler av organisasjonen.

I korpuset omtales *konflikter* på arbeidsplassen i all hovedsak som negativt. Konstruktive forslag til endringer og forbedringer er derimot kjærkomment, og i deler av korpuset regnes dette som en forutsetning for endringer i organisasjonen.

Arbeidere som yter *motstand* mot metodologiene er kilde til forstyrrelser og kompleksitet, og er derfor lite verdige. I de verste tilfellene kan *sabotører* ødelegge *balansen* i et team eller en organisasjon, og ødelegge både produktivitet og sjansen de andre arbeiderne har til å bli lykkelige i arbeidet.

Redusere, (være) realistisk (DIGNITY)
eksponere, kunnskap, (muntlige) diskusjoner, kultivere, distinksjoner,

Alle personer kan imidlertid skape et bedre fellesskap gjennom å *redusere* kompleksitet og å disiplinere seg. Dette inkluderer å være *realistisk*. Den ukomplekse verdigheten er knyttet til menneskets naturlige behov for å forenkle, på grunn av kognitive

begrensninger og på grunn av at tid ofte er en knapp ressurs. "*Most of us have limited room in our heads, so we focus on getting our code to work more than organization and cleanliness*" (Martin 2008:139). Høy verdighet knyttes til å holde hodet klart på tross av kompleksitet og press. Realistiske planer gjør det mulig å minske gapet mellom det som er mulig i virkeligheten og det som er ønskelig. Det enkleste i organisasjonssammenheng er ofte bare å følge standarden og jobbe metodisk. Høye ambisjoner bør unngås fordi de kan skape treghet, nøling og ubalanse. Eventuelt kan de kan *deles* opp og realiseres steg-for-steg (Pichler 2010:30).

(Menneskelige trekk som gir mulighet til å oppnå storhet?)

For å oppnå storhet er man imidlertid nødt til å utfordre standarden, forenkle og forandre den på en måte som gjør den enda enklere, mer presis, mer fokusert – for alle. Verdighet kan derfor oppnås ved at man ytrer eller gjør det enkle og innlysende. Verdighet kan også oppnås gjennom å eksponere seg selv eller andre, at man tør å påpeke egne og andres begrensinger. For å klare dette kreves det *kunnskap*. Kunnskap er helt essensielt, både for å forstå hvilke alternativ man eventuelt velger vekk, og for å kunne foreta gode avgjørelser, og kunne uttale seg med autoritet.

I smidig-litteraturen nevnes mange eksempler i forhold til kommunikasjon – *muntlig* diskusjon gir ofte mer verdi enn skriftlig dokumentasjon i forhold til tiden som investeres. Det å snakke sammen direkte gir verdi gjennom at man raskere kan gi tilbakemeldinger og at det blir enklere å korrigere misforståelser. Noen mennesker har evnen til å *uttrykke seg klart* og enkelt, og *bryte ned* komplekser til bare det aller nødvendigste. Alt som *distingverer*, enten det er personer eller sammenfatninger, lister og definisjoner, er av det gode fordi de bidrar til tydelighet og klarhet. Personer og ledere som klarer å kultivere, og dyrke fram en god kultur, luke vekk "ugress" og fjerne "waste" (Lean), blir ansett som høyverdige (Pichler 2010:56; Poppendieck og Poppendieck 2009).

Ledere, brukere (SUBJECTS)

Alle har en mulighet til å forenkle. Profesjonelle gjør imidlertid grep som forenkler mer *intuitivt* og *effektivt* enn amatører, og har

dessuten mer naturlig autoritet sånn at de har større mulighet til å skjære gjennom i en organisasjon eller en forsamling. Som i den industrielle og prosjektorienterte verdenen er *ledere* og *profesjonelle* verdige subjekter, men *brukere* (av produktene) og kundene status og meninger tillegges også svært mye verdi. Det finnes to typer ledereskikkelser i scrum, *mesteren* og *eieren* – karakteristikkene går imidlertid igjen i korpuset under andre navn. Lederen, i form av en *mester* (scrummaster) har kun autoritet over arbeidsprosessene, ikke menneskene som utfører prosessene. Som en gartner, kan han/hun så frø, luke og intuitivt forstå hva som må gjøres. Å sette realistiske mål er overlatt til andre roller, for eksempel (*produkt)eieren*, brukere eller utviklerne selv. Mesteren kan ikke delegere oppgaver eller sparke ansatte. Mesterrollen blir dermed en slags blanding av å nære fram energi, lede teamet gjennom å motivere til å holde fokus, stille de riktige spørsmålene og ellers forenkle arbeidet ved å *fjerne hindringer* som måtte oppstå. Eieren er en rolle som *"unites the authority and responsibility traditionally scattered across separate roles, including the customer or sponsor, the product manager, and the project manager"* (Pichler 2009:2-3). Denne lederskikkelsen går også igjen i andre bøker under navn, lederen skal ha autoritet om det er nødvendig. Han/hun har oversikt over arbeidsoppgavene til de ansatte, og kan utøve prosjektkontroll om det trengs.

Mester, profesjonelle, gartnere, (produkt)-eiere, eksperter

(Repertoaret som beskriver verdige subjekter)

En god leder som skal formidle noe i tråd med den ukomplekse verdiverdenen må unngå *faguttrykk* og fatte seg i *korthet*, mens en dårlig formidler har for *mye innhold* på lysarkene, *snakker for fort* eller har for mange og for *lange setninger*. Profesjonelle og eksperter i den ukomplekse verdenen har høy verdi fordi de kan *prioritere* og *kontekstualisere*, og fordi de om nødvendig klarer å formidle et budskap uten å støtte seg til komplekse systemer, budsjetter og lignende. En god leder gjør store problemer om til mindre problemer.

Alle som behersker den paradoksale egenskapen å begrense og

legge bånd på seg, men som samtidig stikke fram hodet og forenkle gjennom å eliminere "waste", stoppe linja (Lean) eller uttrykke seg enkelt og *lettfattelig* om komplekse tema regnes typisk som høyverdige. Utviklere som skriver enkel kode som alle forstår, og som setter seg raskt inn i andres problemer, og som ikke fordyper seg for mye eller sløser tid på eksperimentering regnes som verdige subjekter.

Uverdige subjekter skaper kompleksitet, og prøver ikke å forbedre seg. En person som fokuserer ensidig på begrensningene – "*fussing about the constraints*" (Beck og Andres 2004:5) – blir distrahert og stresset, og gjør dermed en dårligere jobb. I denne betydningen blir forestillingen om det mistilpassede individet koblet til det amatørmessige.

Rammeverk (OBJECTS)
Fortellinger (stories), rammer, grenser, (boundaries), omfang, definisjoner, sammenfatninger, skisser, regler, prosedyrer, flaskehalser

(Repertoaret som beskriver verdige objekter og teknologier)

Viktige objekter er teknologier og metoder som gjør ting enklere for subjekter. *Prosedyrer, rammer*, *definisjoner*, *abstrakter* og *sammenfatninger* har høy verdi, fordi de er nyttige verktøy sånn at subjektene klarer å formidle eller enkelt finne og gjøre det som er nødvendig. Gode *fortellinger*, eksempler og illustrasjoner er viktige. Ordninger eller systemer som hjelper en å filtrere, sortere eller prioritere har høy verdighet. Konseptet *omfang* er svært viktig i forbindelse med oppgaver, siden det representerer noe som subjekter faktisk kan forhandle eller regulere selv i en hverdag hvor mange andre krav er bestemte på forhånd. Flaskehalser (i produksjonen) kan representere noe positivt, siden de tydeliggjør problemer, og gir mulighet til utbedring. For mange retningslinjer fra ledelsen eller bedriften er på sin side uverdige, fordi de skaper unødig kompleksitet og forstyrrelser.

Ukomplekse objekter bidrar til å forme og håndtere tiden, ressursen som er stabil og som har en konstant knapphet. Teknikker og rammeverk for å håndtere planlegging og arbeidsfordeling gis stor plass i korpuset. De analoge løsningene innebærer ofte å forkaste digitale løsninger til fordel for analoge teknologier. Både XP og Kanban (Lean) benytter et lappesystem for

oppgaver. Hver enkelt lapp inneholder en "brukerfortelling" med en tittel, og en kort håndskrevet eller tegnet *skisse* av en oppgave.

Oppmerksomhet (FORMULA OF INVESTMENT)
Bevissthet, kortsiktig perspektiv

(Hva investeres, hva kan potensielt øke verdien, og hva må ofres)

For å gjøre en god jobb og bli en høyverdig person må man ofre både *the big picture* og *detaljer* i bildet. Langsiktighet abstraheres, noe som ikke innebærer at langsiktighet forsvinner helt – det å kunne gjøre en god jobb på kort sikt er en forutsetning for å kunne eksistere på lengre sikt. Dette er en tenkemåte som ofte bidrar til å gi legitimitet til argument fra andre verdiverdener, og som også avslører det ubehagelige faktum at forenkling alltid ofrer en større eller detaljert forståelse til fordel for enhet eller fokus på det lokale. Potensial kan i seg selv være verdt å ofre, eller i hvert fall utsette, hvis det bidrar til at man blir ferdig i tide (Cohn 2009:293): "*[...] our bias should be toward adjusting scope to fit available resources and schedule.*"

Verdige personer har en egen evne til å investerere all *oppmerksomhet* på den ene oppgaven som skal gjøres. Dette betyr ikke at man glemmer alt rundt, men at man midlertidig legger det bort ved hjelp av et organiseringssystem, f.eks. i et *lappesystem* på veggen eller i en *skuff,* eller på den andre siden av *prosjekthorisonten*. Det betyr heller ikke at man kan prioritere en oppgave over lang tid Når man har investert riktig, kan man forvente at man står igjen med et resultat som er så godt som det kan bli – gitt alle begrensningene som var der, men som ikke fikk komme i veien da selve arbeidet ble gjort. Investeringen kommer alle til nytte. Når teamet får tid til å konsentrere seg i øyeblikket uten å samtidig bekymre seg for framtiden blir det mindre stress.

Det er selvsagt en risiko for at en handling, et prosjekt eller et resultat likevel ikke blir bra nok. Hvis resultatet ikke skulle bli tilfredsstillende på tross av investeringen, må man prøve å se problemet gjennom en ny *ramme* i neste syklus, og investere *annerledes* (ikke nødvendigvis mer).

Fokusere, balansere (RELATION OF WORTH)

Gjennom relasjonen *fokus* kan verdifulle tilstander (state of worthiness) omslutte mindre verdige tilstander. Begrepet fokus

beskriver selve essensen som skiller det som er relevant fra det som er mindre relevant i øyeblikket eller i situasjonen. Det verdige kan holdes skarpt og nære, og det mindre verdige holdes samtidig utydelig og på avstand – sånn at det ikke forstyrrer. Det er imidlertid i en kombinasjon med relasjonen *balanse* at fokus kan bidra til å skape noe som er varig, og godt for alle.

(Hvilke relasjoner gir verdi?)

Mennesker som behersker fokus og balanse unngår å bli for tung, overlesset og stresset, og bidrar til å gi alle andre økt verdi gjennom at de har oversikt, gjennom at de er disiplinerte, og gjennom at de skaper bevissthet rundt hva som er mulig i virkeligheten. "*Do work to get the team's velocity up. But plan based on your actual speed, not your hopes. That keeps your plan closer to reality, and keeps you focused on your most important role, steering the project to success*" (Jeffries mfl. 2000:73).

Begrepet *spredning* (scatter) brukes i Scrum og i Lean for å beskrive alle tingene som kan forstyrre en enkeltutvikler eller et team og hindre dem i å gjøre et substansielt, sammenhengende arbeid. Spredning distraherer og bryter opp arbeidsflyten på det individuelle nivået. "*Human concentration is easy to break and hard to get back. [...] Frequent interruptions are time-wasting*" (Lui og Chan 2008:20). De verdige personene forstår dette, og skaper plass og rom for alle. Enkeltutviklere kan på sin side *slå av* telefonen, e-post og alle former for sosiale medier som ødelegger konsentrasjonen.

Eliminere, ramme (inn)
(NATURAL RELATIONS AMONG BEINGS)
Skjerme, gjennomføre, fjerne, isolere, justere, konsentrere (seg), begrense, stenge ute, holde adskilt, omstrukturere

(Verb som binder sammen verdige subjekter og objekter)

De naturlige relasjonene mellom subjekter og objekter i den ukomplekse verdiverdenen hjelper den ene eller den andre å *eliminere* det som forstyrrer og *gjennomføre* (accomplish). En god arbeider deler arbeidet opp i mindre, gjennomførbare deler. Gjennom å *isolere* eller skjerme kan han/hun forløse sine planer, sine ideer og sitt kreative potensial. Ved å *ramme inn* innhold i bilder eller historier kan en leder eller utvikler illustrere hvordan enkelt-elementer i en organisasjon *flyter* samtidig som de gjør det mulig midlertidig å stenge ute innhold som "*detracts from the subject*" (Poppendieck og Poppendieck 2009:xvii). Innenfor rammen

kan man igjen velge hva man vil *fokusere* på.

Det å eliminere er alltid forbundet med effektivisering eller forbedring. Automatisering (fra den industrielle verdiverden) kan *eliminere* manuelle operasjoner, og frigjøre tid. Arbeidsoppgavene (i hvert fall i XP) skal også elimineres, man skal gjøre seg ferdig med dem og fjerne dem fra organiseringssystemet.

Å *omstrukturere* kode (refactor) vil si å forbedre koden uten å endre funksjonsmåten. Dette kan gjøres for å øke lesbarhet, hastighet og er viktig for å gjøre koden enklere, renere og raskere. Hvis man klarer å konsentrere seg kan en arbeidsoppgave utføres med minst mulig avbrytelser, noe som gir god gjennomstrømming og mulighet for å *justere* kursen. Samtidig er det viktig at arbeidet blir levert i en *jevn strøm* (even flow). Det er verdifullt å ha et jevnt arbeidstempo og investere omtrent like mye tid fra uke til uke. For å skape verdi må man være 100 % konsentrert – for oppgaven(e) man holder på med i øyeblikket, ikke for alt det andre. Man må være *innstilt* og kunne nok om emnet til å *fatte seg i korthet,* og *disiplinert* nok til å ikke flyte utover. Dette er imidlertid ikke noen hindring for å gjøre mange ulike oppgaver, så lenge man gjør dem i adskilte produksjonskontekster. Like viktig som å fokusere, er det å begrense, å stenge eller lukke ute ut fra kontekst. Mange organisasjoner stenger tilgangen til sosiale medier, eller skaper regler som regulerer bruk av Facebook i arbeidstiden m.m. En god scrummaster gir hele teamet økt verdi gjennom å *fjerne* hindringer, *løse* konflikter, på den måten kan teamet jobbe smidig og konsentrert, de får sjansen til å *fokusere*. Det som begrenser er uansett positivt i den forstand at det tydeliggjør problem, og gjør det mulig å forenkle eller forbedre.

Syklus
(HARMONIOUS FIGURE OF THE NATURAL ORDER)
Iterasjon, balanse, tid synkronitet, overgang (transition), harmoni

Den ukomplekse logikken låner mange metaforer fra naturen. Idealets harmoniske figur er konstruert rundt en figur med rot i østlig filosofi: *syklusen*, som er et uttrykk for *stabilitet*, *balanse* og mestringen av *tid*. Syklusen bringer orden gjennom at den kombinerer absolutte grenser med varighet – og den tydeliggjør

verdien av tid gjennom å dele opp prosjekter. "*Permaculture is a philosophy and practice of sustainable living in a balanced ecosystem*" (Beck og Andres 2004:103). Gjennom å arbeide i iterativt får arbeiderne et mer *bevisst* forhold til hvordan de bruker tiden på jobben. Gjennom å balansere progresjon og "sunn" vekst, skapes *stabile* organisasjoner, også i *overgangsfaser* og i tider med mange endringer. Ordet harmoni benyttes lite i korpuset, men er likevel beskrivende for det idealiserte bildet av kunnskapsarbeid som skisseres, som et tidshjul. Syklusen gjør det lett å ha oversikt, og dermed lett å innføre endringer, men i den siste uka i syklusen - når det er sprint – er det fokus på ferdigstillelse og opprinnelig målsetning. Ingen nye funksjoner legges til i denne perioden, heller ikke produkteieren kan gjøre det. Dette bidrar til at arbeiderne kan bli ferdig med oppgavene også like før ferdigstillelse, sånn at de kan dra hjem med senkede skuldre, og komme tilbake neste arbeidsdag, revitaliserte og fulle av "energi".

(Virkeligheten som forstått gjennom ekvivalensprinsippet.)

Størrelse er i seg selv en viktig figur, en organisasjon bør aldri la enheter eller avdelinger vokse seg for store. Da blir de for kompliserte og derfor trege og ineffektive. Små team har større produktivitet og mindre total innsats. Å balansere størrelse på teamene er derfor en viktig oppgave.

Kontinuerlig, responsiv (MODEL TEST)

Testene som bekrefter om organisasjoner, systemer og produkter er ukomplekse nok, er ikke bare forbeholdt kritiske situasjoner, i stedet foregår de *kontinuerlig*. Det er også poenget, et smidig og stabilt system skal være *responsivt* og reprodusere seg mer eller mindre av seg selv. Dette fremstilles som et ganske ufravikelig krav. "*Like continuous testing, continuous refactoring is not optional*" (Poppendieck og Poppendieck 2009:82). Dette gjelder også selve programmeringen. "*So the solution is to continuously keep your code as clean and simple as it can be. Never let the rot get started*" (Martin 2008:250). Om en løsning eller endring er enkel, vil den også raskt kunne implementeres og bevise sin verdi. Ubehag i form av *stress* tyder på at ikke alt er så enkelt som det burde vært. Hvis

(En test som kan brukes for å prøve gyldigheten av et argument.)

arbeidere ofte jobber *overtid*, eller hvis omgivelsene stiller stadige krav om endring, er det noe som er galt. Mål fra den industrielle og kollektive verdenen vil også kunne brukes for å kvalifisere tester om et system er enkelt nok. Har Scrum økt produktiviteten? Har HR-tiltakene hatt noen effekt?

Flyt
(JUDGEMENT)

(Hvordan avgjøres testen?)

Hvis ikke, er det en gylden sjanse til å finne begrensningen, og fjerne den sånn at *flyten* blir bedre. Om man finner begrensninger blir dette en indikator på at et system eller en relasjon har forenklingspotensiale. Når prinsippet alltid er å forenkle, så vil man avdekke begrensninger fortløpende – fram til alt flyter optimalt. Den endelige dommen på om man i tilstrekkelig grad klarer å forenkle blir imidlertid endelig avgjort av *brukerne* eller eller *mottakerne, publikum*. Mens metodologiene av og til snakker om å skjerme seg fra omgivelsene, kan det å skape flyt noen ganger innebære å fjerne organisasjonsnivåer. "*By simply removing the three levels of customer support and having developers talk directly to customers, 40% of the total time of 800 developers was freed up*" (Poppendieck og Poppendieck 2009:25).

Varighet, kontinuitet
(EVIDENCE)
Stabilitet, kvalitet

(Bevisets form.)

Beviset på om noe er enkelt nok er om planene for gjeldende syklus blir *innfridd*, om prosjektgruppa er *smidig*, og om arbeidet *flyter*. Den ultimate testen fra et produksjonsperspektiv er at neste syklus kommer, og at den i *innsats* er lik den forrige. *Stabilitet* blir en indikator på om prosesser trenger ytterligere forenkling. "*It's the full-blown cases of hero complex that you have to watch out for because their behavior may deliberately cause the project to become unstable*" (Berkun 2005). Beviset for at et argument eller budskap er enkelt nok er om det klarer å *overbevise, nå ut*, eller i hvert fall bli *spredd* videre. Det endelige beviset på om en person, en prosess, et produkt eller et team er ukomplekst og fleksibelt nok, er om det har *kvalitet*.

Simpelhet
(THE FALL)
Naiv, vanlig

Personer taper ansikt når de ikke skjønner når *nok er nok*, når man bør *begrense* seg, når man skal *stenge* døra, eller *slå av* mobilen. Når noe ikke kan gjøres tydelig er det ikke verdifullt. *Tung*, *innfløkt* og *akademisk* argumentasjon, kompliserende moralske argumenter,

(Tap av verdighet)

eller komplekse *politiske* og *langsiktige* argumenter har liten naturlig verdi. Å gå for en *innfløkt* og *uoversiktlig* løsning vil være en fallitterklæring. Alle former for kompleksitet truer en ukompleks/smidig test. Byråkrati, komplekse rutiner, søknader, kunder og for komplekse løsninger eller detaljerte målesystemer forstyrrer og hindrer oversikt, de begrenser flyten som er nødvendig for å kontinuerlig realisere og reprodusere testen. Kravet om automatisering og forenkling i organisasjoner er dermed en naturlig følge av at aktørene søker orden og ryddighet, en enklere og mer smidig hverdag – for seg selv og for alle.

Ifølge den ukomplekse logikken er man nødt til å ofre det som gjør livet mere komplisert, inntil en viss *grense*. Det å definere hvor denne grensen går er et av de store problemene for personer som refererer til ukomplekse idealer. Det store fallet i verdighet skjer når det ukomplekse rett og slett bare blir *simpelt*, når det blir noe som skjer stadig vekk, når noe ikke lenger har spenning eller utfordring.

8.7 Kompromiss mellom den ukomplekse verdiverden og andre verdener

For at sosiale kritikker skal få så stor kraft at de kan føre til sosiale endringer, krever de bred mobilisering og evne til å gi et troverdig løfte om å tilfredsstille behov som det kritiserte systemet ikke kan. Kompromiss kan potensielt rekonfigurere verdiverdener, knytte dem til nye prinsipper og institusjonalisere nye former for tester.[39] Den ukomplekse verdiverdenen brukes ofte som et tillegg for å understøtte prinsipper og tester fra de ulike verdiverdenene. Prinsippet om enkelhet harmonerer spesielt godt med ideen om at en prosess eller handling skal rasjonaliseres (industriell logikk), eller at det enkle og skisseaktige inspirerer og åpner (artistisk logikk). Enkelhet fordrer ellers direkte menneskelige relasjoner (hjemlig og prosjektorientert logikk), og kan knyttes til det å skape enkle meninger/slagord eller verdier folk kan samles rundt (kollektiv logikk).

Jeg vil avslutningsvis i dette kapitlet skissere, med utgangspunkt i 2000-2010-korpuset, hvordan den ukomplekse verdiverden inngår to kompromiss, et med den *industrielle* og et med den *kollektive* verdenen. Det første kompromisset (mellom det industrielle og det ukomplekse) ligner det jeg kalte en smidig fagverdighet i grenseanalysen min. Til slutt i kapitlet viser jeg

39 Se også om kompromiss på side 52.

hvordan dette kompromisset begrenser og danner nye muligheter for den prosjektorienterte verdiverdenen beskrevet i Boltanski og Chiapello (2005).

8.7.1 Industrielle og ukomplekse kompromiss i 2000-2010-korpuset

En kombinasjon mellom den industrielle verdenen og den ukomplekse verdenen kommer til uttrykk gjennom begreper som *enkle beregninger*, *filter* og *produksjonssykluser*. Disse gjør det mulig å kombinere høy produksjon med enkelhet og oversiktlighet. I dette kompromisset er enkelte industrielle idealer nedtonet. Spesielt idealene som fremhever verdien av å realisere et hvert (menneskelige) potensial samt å følge enhver ambisjon har mistet gyldighet. Effektivitet som ekvivalensprinsipp har også tapt legitimitet til fordel for prinsippet om enkelhet.

Verdighet i en ren, industriell verdiverden innebærer at menneskers iboende potensial for å utføre et arbeid kan realiseres. Å investere i menneskelige evner og energi er regnet som verdifullt, mens det å unnlate å bruke tilgjengelig menneskelig potensial medfører et brudd på menneskelig verdighet (Boltanski og Thévenot 2006:206). Denne formen for verdighet kan kritiseres fra et ukomplekst ståsted, som hevder at det enkle ofte er det beste, at man alltid skal søke å redusere. Ideen om iterasjon/syklus er eksplisitt posisjonert mot ideen om handling/effekt og den implisitte ekvivalensen mellom "nåtiden" og "framtiden" i den industrielle verdenen (Boltanski og Thévenot 2006:204–5). Den ukomplekse verdiverden hevder at man kan ikke vite effekten av en plan sikkert, og man kan aldri få det resultatet man planlegger uansett. Derfor må man ha *rom* for å gjennomføre endringer.

Ekvivalensprinsippet om forenkling og oversiktlighet (transparens) ser imidlertid ut til å være gyldiggjort i mange sammenhenger når det settes i sammenheng med objekter og innhold fra den industrielle verdenen. Jeg vil utdype dette gjennom noen eksempler: smidig *planlegging* og *fordeling* av oppgaver, *produksjonssyklus, kommunikasjon* og til slutt *målesystemer*.

Planlegging og fordeling

De første smidige metodene ble utviklet som en reaksjon mot detaljert planlegging. Likefullt vies tidsplanlegging og organisering av arbeidsdagen en stor plass i 2000-2010-korpuset. En av løsningene som skisseres, er å planlegge på kort sikt (Jeffries et al 2000:79): "*Inside each release, an Extreme team plans just a few weeks at a time, with clear objectives and solid estimates.*". Ved å planlegge kortsiktig slipper man å forholde seg til framtiden, og man blir bedre rustet til å håndtere problemene i nåtiden. Risikoen for å investere feil minimeres, og det blir lettere å justere kursen.

Brukerfortellinger

Konseptet brukerfortellinger (*user stories)* finnes i både XP, Scrum og Lean (under navnet Kanban). Idéen er at hver oppgave eller egenskap som skal implementeres eller endres, kan formuleres på et lite kort, typisk med en størrelse som en post-it-lapp. En brukerfortelling inneholder en kort beskrivelse av en egenskap fortalt fra perspektivet til en bruker som ønsker det: "*As a user, I want to be able to cancel a reservation"* (Cohn 2005). Brukeren kan også være en administrator, en spesifikk kunde eller ulike typer brukere. Poenget, som det fremstilles i for eksempel Cohn (2009: 238-9), er at brukerfortellinger endrer fokus fra å skrive om egenskaper (i dokumentasjon) til å diskutere dem. I hjørnet av hver lapp er det estimert hvor lang tid oppgaven vil ta.

Slike bruker-fortellinger kan for eksempel nedtegnes på starten av en prosjektsyklus, organiseres i en boks, og settes opp på en vegg, tavle eller på et bord for planlegging og organisering. Ofte henger de på en tavle mens en syklus pågår. Både utviklere, ledere og kunder kan dermed se, endre og flytte om på oppgavene som skal løses før syklus tar slutt. Lappene kan forflyttes og utsettes til den andre siden av prosjekthorisonten hvis/når det skulle kreves. Siden det er svært enkelt å lage en lapp, er investeringen liten hvis/når oppgaven må endres, utsettes eller fjernes. Det er et poeng at de håndskrevne lappene er mer effektive og enklere enn et tilsvarende digitalt system. Berkun (2005) anbefaler at alle i teamet henger sine oppgaver på synlige plasser:

> *[...] making the list visible allows people to collaborate on resolving the issues. "Oh, that's on my list, too. Should you take it, or should I?" This is one reason I"ve kept my issues list up on a whiteboard in my office or in the hallway. (A web site might work fine, but in my experience, no one ever looks at that list but the person who created it. Non-virtual and informal places work much better.)*

Alle kan plukke og utføre en brukerfortelling som ikke er påbegynt, innenfor rammene som ligger i inneværende syklus. I Lean/Kanban skal brukerfortellingene/kortene flyttes mot høyre etter hvert som de når ulike stadier. Dermed ligner prosessen litt på et samlebånd, poenget er imidlertid at dette gir oversikt (Poppendieck og Poppendieck 2009:123):

> *All kanban systems are designed to limit work-in-process, because the more work-in-process, the slower the flow. That's one of the reasons kanban systems were invented in the first place – to limit work-in-process and thus increase flow. The mechanism for limiting work-in-process in software kanban is to limit the number of items in each step of the workflow.*

Lean er imidlertid noe strengere her enn Xp og Scrum. En prestasjonsbasert ytelse, hvor enkeltutviklere konkurrerer mot hverandre blir disiplinert gjennom kanban-kortene (Poppendieck and Poppendieck 2009:127):

> *There should be no partial credit; stories are not done until they are truly done, that is, tested, integrated, documented, and ready to deploy (if not deployed). Anything less gets zero credit.*

Brukerfortellingene er et eksempel på planlegging ut fra et ekvivalensprinsipp om oversiktlighet og enkelhet. Brukerfortellingene skal være korte og konsise, de skal henge lett tilgjengelig og synlig. Alle i teamet skal kunne velge mellom de ledige oppgavene, men de får ikke jobbe utover den estimerte tiden. Derfor er det viktig at estimatene stemmer overens med virkeligheten, de må være realistiske. At planer er realistiske er viktigere enn at de er ambisiøse:

> *Reliability is more valuable than false ambition; it is the prerequisite for making realistic forecasts (Pichler 2010:99).*

> *The state of a shared plan provides clues about the state of the relationship between the people affected by the plan. A plan out of touch with reality betrays an unclear, unbalanced relationship. A mutually agreed upon plan, adjusted when necessaary to reflect changing reality, suggests a respectful, mutually valuable relationship (Beck og Andres 2004:91).*

Hvis man må endre *omfanget* i planene, men likevel kan ende opp med et stabilt produkt på enden av hver syklus, har man både orden, overskudd og et system som gjør at det mulig å implementere det man utsatte. Ikke minst har både kunder, brukere og samarbeidspartnere *oversikt*.

Produksjonssyklus

Et viktig ideal som trekkes fram flere steder i 2000-2010-korpuset er legitimeringen av *jevn, stabil* produksjon, ikke en høyest mulig produksjon. Dette er en del av kompromisset mellom det industrielle og det ukomplekse, som tar i betraktning at jevn produksjon vil være mest lønnsomt for organisasjonen på sikt. For å kunne oppnå dette, må prosjektene deles inn i biter, ofte kalt *release* eller *produksjonssyklus*, som igjen gjerne er delt inn i

iterasjoner. Beck og Andres (2005:5) lister opp noen av fordelene med denne måten å dele opp arbeidet på:

> *XP calls for short release cycles, a few months at most, so the scope of any slip is limited. Within a release, XP uses one-week iterations of customer-requested features to create fine-grained feedback about progress. Within an iteration, XP plans with short tasks, so the team can solve problems during the cycle. Finally, XP calls for implementing the highest priority features first, so any features that slip past the release will be of lower value.*

Oppdelingen gjør det enklere å planlegge, prioritere, diskutere og løse eventuelle problemer. Også her er oversiktlighet det øverste ekvivalensprinsippet, som regulerer planleggingen. Korpuset fremmer på samme tid tanken om at syklusene til ulike team bør foregå synkront eller parallellt (Cohn:343-345).

Syklusen og iterasjonene blir holdt "hellige" i alle metodologiene. Omfang (scope) er den ene variabelen som de smidige metodologiene tillater at man forhandler om eller utsetter innenfor en syklus – kvalitet, og pris er som oftest gitt på forhånd. Man kan tilpasse omfanget og utsette å implementere planlagte oppgaver hvis det er skjellig grunn, men bøkene er mer eller mindre enige i at man aldri bør forlenge syklusen – rammene og strukturen rundt syklusen har mye verdi i seg selv, ikke minst av hensyn til arbeiderne.

Kommunikasjon og koordinering

Kommunikasjon er i seg selv viktig for at prosjekter skal kunne gjennomføres, og lederne må ha en oversikt over det som foregår i organisasjonen. Derfor må møter og koordinering anta en form som gjør dette mulig. Da blir kommunikasjonen i et prosjekt for eksempel beskrevet på denne måten (Gyurky 2006:77):

> *If a project is to move ahead at high speed with a view*

> *toward conserving resources, verbal and written communication should be as open as possible. This is not to say that the team chiefs, systems engineers, and project manager are bypassed, but that technically important data and coordination flow smoothly. The project manager and the subsystem managers are kept informed through the daily coordination meetings, and through the technical writers as well.*

Det finnes altså måter å kommunisere og avholde møter som gjør at ledere kan holde seg informerte om det som foregår i organisasjonene. Så lenge møtene er korte og informative er det uproblematisk å ha mange møter.

Enkle beregninger

Enkle beregninger kan si noe om trender, og kan være verdifulle for å kunne kommunisere et budskap ut til resten av bedriften. Sidene målingene er enkle, koster de lite, og dette åpner igjen for hyppigere målinger (Cohn 2009:429):

> *Ask most people what the purpose of measuring is, and they will probably say that it is to determine how big, how heavy, how long, or how much of something there is. This is an overly ambitious definition of measuring. The real purpose of measuring is to reduce uncertainty. A measurement does not need to be exact for it to help in reducing uncertainty.*

Det er ikke så viktig om målingene er helt eksakte. Enkle beregninger gir gjerne en god nok indikator på om organisasjonen beveger seg i riktig retning eller ikke.

Lean formulerer utviklernes viktigste oppgave som å møte kundenes forventninger uten å skape økt *"failure demand"* – forstått som kundens krav om support og feilretting. Dette er også en måling. Hvis kravet på support øker etter en release, er kvaliteten for dårlig. "*The real metrics are the things that happen in the real world*"

(Jeffries mfl. 2000). Typiske måleindikatorer som nevnes i korpuset er f.eks. antallet rapporterte feil (bugs) etter leveranse, eller hvor lang tid det tar før et produkt begynner å tjene penger etter at det er ferdig, inntjening per ansatt, hvor tilfredse eller stressede de ansatte er, eller antall linjer skreven kode, eller antallet utførte oppgaver.

8.7.2 Kollektive og ukomplekse kompromiss i 2000-2010-korpuset

Et kompromiss mellom den kollektive verdenen (Boltanski og Thévenot 2006:185) og den ukomplekse verdenen kommer til uttrykk når begrensninger og forenklinger settes i relasjon til det beste for kollektivet av mennesker i teamet, organisasjonen eller faget. Enkeltpersoner har ikke høy verdighet i den kollektive verdiverdenen, men det som gjelder mange, alle, har stor betydning.

Menneskelige organisasjoner

Kent Beck skriver for eksempel at en av motivasjonene han hadde for å utvikle XP, var å gjøre arbeidsdagen mer produktiv, men også mer *menneskelig* (Beck og Andres 2004:3-4).

> *XP is my attempt to reconcile humanity and productivity in my own practice of software development and to share that reconciliation. I had begun to notice that the more humanely I treated myself and others, the more productive we all became. The key to success lies not in self-mortification but in acceptance that we are people in a person-to-person business.*

Forutsigbarhet og fritid gir energi (industrielle verdiverden) og motiverte arbeidere, og gjør at man yter optimalt når maner på jobb. Derfor fraråder XP at arbeiderne jobber mer enn 40 timer i uka (Beck og Andres 2004:41): "*Work only as many hours as you can be productive and only as many hours as you can sustain.*" Det samme argumentet om at en uke skal være delt opp i et visst antall

tidsenheter går igjen i Scrum og Lean (Cohn 2009:13-14):

> "*A lack of overtime is [...] one factor contributing to higher job satisfaction among people working on agile teams.*"

For å se målinger i et menneskelig lys, så er flere av forfatterne oppmerksomme på at målinger tradisjonelt har blitt brukt for å kontrollere arbeidere. Tall skal følgelig ikke tillegges *for* mye verdi (Cohn 2009:447):

> *Whenever numbers are collected, there is usually someone in the organization who becomes too attached to them and gives them undue significance. The metrics are then used to beat up teams. [...] the metrics [...] should be used to focus effort and aid understanding, not to place blame or enforce compliance.*

Bærekraftig arbeidstempo

Å ha et bærekraftig arbeidstempo (*sustainable pace)* betyr å jobbe omtrent like lenge hver uke, hver måned, hele året. Rammene som blir gitt av smidig prosjektplanlegging (basert på post-it-lapper og brukerfortellinger), har dermed den egenskapen at de skaper stabilitet, orden og harmoni, noe som i bøkene fremstilles som noe som er godt for *menneskene* som jobber i organisasjonene – ikke bare prosjektene i seg selv.

Likeverd, livskvalitet

Grensene og rammene i metodologien (spesielt XP) settes også i sammenheng med rettferdighet og likeverd (Beck og Andres 2004:21):

> *If members of a team don't care about each other and what they are doing, XP won't work. If members of a team don't care about a project, nothing can save it.*

> *Every person whose life is touched by software development has equal value as a human being. No one is intrinsically worth more than anyone else. For software development to simultaneously improve in humanity and productivity, the contributions of each person on the team need to be respected. I am important and so are you.*

Likeverd fremstilles som en forutsetning for at team-medlemmene skal kunne føle tilhørighet, fellesskap og forpliktelse til et prosjekt eller et team. Lean dyrker samme filosofi, og henviser til Drucker (1999), en kunnskapsorganisasjon må først og fremst behandle kunnskapsarbeiderne respektfullt. Mennesker kommer *før resultater*, fordi resultatet alltid avhengig av at menneskene fungerer optimalt (Poppendieck og Poppendieck 2009:196–203). Kapitler i korpuset som henvender seg til konsulenter eller team-managere oppfordrer gjerne å vurdere om teamene er stressete, om utviklerne engasjerer seg, om de tør å si ting høyt uten å seg seg rundt.

Hackeren og individualitet

Ordet hacker er omtalt fem ganger i korpuset, så forestillingen kan ikke sies å være særlig viktig. En av gangene ordet brukes, blir det brukt for å assosiere hackeren med individualisme og det å ha for mye å gjøre (Gyurky 2006:118).

> *What can you do about hackers, or those engineers who have a misconception about their responsibilities and a subjective view of their own personal worth? I have frequently run into very intelligent and well educated people with powerful self-images who had their own ideas about what they needed to do, and how. Asking these people nicely to remain at a meeting and participate and contribute to the substantiation of the design object is useless. Strangely enough, their answer is usually, "I have work to do. I'm already behind. I*

> *don't have time for this." With these responses, at first, even an experienced manager-architect is caught off guard, thinking, "Maybe he is right. Maybe I am wasting everybody's time."*

Denne typen personlighet har ikke bare negativ innflytelse på sin egen arbeidssituasjon, men på teamet, på lederen og på hele organisasjonen. De *skaper* støy og stress. Siden smidige metoder inneholder *krav* om endringer på mange nivå, vil det alltid være en viss *individuell* motstand (Cohn 2009:104) som krever håndtering.

8.8 Begrensning og nye muligheter for den prosjektorienterte verdenen i IKT-organisasjoner

Den prosjektorienterte logikken hadde mange praktiske begrensninger i IKT-organisasjoner, og ble derfor gjenstand for en omfattende kritikk i håndbøkene for smidig utvikling og prosjektorganisering. Individualisme, fleksible prosjekter og nettverk, som i 90-tallskorpuset ble presentert som muliggjørende og åpne bare man investerte riktig, representerer i 2000-2010-korpuset noe kaotisk, uforutsigbart og begrensende. Den prosjektorienterte verdenen bidro ikke til forutsigbarhet, den bidro ikke til lønnsomhet, og den ble til syvende og sist oppfattet som belastende for arbeiderne. Innenfor IKT hadde prosjektene en tendens til å bli langt dyrere enn planlagt, eller ikke gjennomført i det hele tatt. Å skape et organisasjonssystem som definerte suksess på en annen måte enn bare ved å fullføre og gå videre til et nytt prosjekt, og som skapte prosesser som var så stabile som mulig var derfor helt nødvendig for at fagretningen og industrien skulle kunne utvikle seg. De mest slagkraftige omveltningene kan bare skje hvis kritikker fra ulike verdiverdener skaper resonans og settes sammen til et kompromiss. Det er dette de smidige metodologiene forsøker å gjøre. I argumentene inngår både effektivitet (industriell logikk), arbeidernes kollektive verdi (kollektiv logikk), ideen om den slanke organisasjonen som deltaker i et nettverk (prosjektlogikk), og løfte om økonomisk avkastning (markedslogikk). Et kompromiss mellom disse verdiverdenene vil imidlertid ikke være stabilt hvis aktørene forsøker å bli enige om hvilket evalueringsprinsipp (henholdsvis effektivitet, trygge arbeidsplasser eller markedsverdi) som er det viktigste (Boltanski og Thévenot 2006:336). For å kombinere disse vidt ulike idealene, brukes dermed ukomplekse idealer som et trygt møtested som alle kan være enige om, og som sikrer kompromissets styrke og stabilitet.

Legitimeringsrommet for å disiplinere arbeidere, som ble åpnet når hierarki, byråkrati og

lojalitet (den hjemlige verdiverden) mistet gyldighet i management-litteraturen, blir fylt av nye imperativ. Den radikale kritikken av prosjektarbeidet, som i essens sa at nettverk, kommunikasjon og detaljerte planer skapte kompleksitet og dårlige arbeidsvilkår har blitt gyldiggjort i fagutøvelsen. En ny test, med referanse til et øverste prinsipp om forenkling, er i ferd med å bli institusjonalisert, gjennom smidig utvikling, men intervjuene mine viser at ukomplekse kritikker også blir reist av personer som ikke bruker smidig metodologi. Even henviser til det smidige ekvivalensprinsippet når han kritiserer metodologier for å være for komplekse, og Daniel bruker dette prinsippet når han skal fortelle om en kollega han ser opp til, eller definere god kildekode.

Det ukomplekse kombinerer krav om indre disiplin med lovnad om å skape stabilitet og balanse (for alle). Å forenkle og organisere arbeidet i sykluser, delt inn i iterasjoner, med arbeidsoppgavene skrevet ned som brukerfortellinger klistret på en tavle – blir foreslått som gode måter å tilnærme seg organisasjonsarbeidet på. Da har man sjansen til få *oversikt, justere* kursen og fokusere innenfor trygge *rammer*.

Forenkling og ukomplekse idealer blir dermed et virkemiddel som gjør at det igjen blir legitimt å prate om metode, målinger og hierarki i organisasjonssammenheng. Disiplineringen av arbeidet er legitimert, og forstått som muliggjørende – ut fra et kompromiss mellom det ukomplekse og det industrielle. Tester lansert med referanse til den kollektive verdenen (hensyn til de andre arbeiderne, arbeidernes fritid) bidrar til å reprodusere denne testen. Stabilitet og rutiner blir også rangert og verdsatt høyere enn lidenskap fordi den personlige gevinsten rett og slett er høyere.

9. Transformasjonen av den prosjektorienterte verdiverden

9.1 Plikt og lyst blant programmerere

Weber forklarte "den protestantiske etikken" som en selvpålagt plikt som lignet et religiøst kall, og denne motivasjonen utgjorde ifølge ham en fundamental basis i datidens kapitalistiske kultur (Weber 1930:54). Det Himanen (2001), Himanen og Castells (2002) og Boltanski og Chiapello (2005) forsøker å beskrive med the "spirit of informationalism" og "the spirit of the new capitalism" er hvordan pliktfølelsen i forhold til arbeidet er erstattet med idealer som alluderer til kreativitet og inspirasjon. Boltanski og Chiapello samler disse idealene under navnet *den prosjektorienterte verdiverden*. Et sentralt argument hos sistnevnte er at forståelser med rot i den hjemlige verdiverden, som regulerer blant annet lojalitet, tradisjon og familiære og personlige forbindelser, samtidig har mistet kraft (Boltanski og Chiapello 2005:69, 135). I ren idealform skaper dette et bilde av en verden hvor firmaer og arbeidsliv har tatt form av et nettverk, hvor alle arbeidere kan skape suksess både for seg selv og de rundt seg i nettverket gjennom å investere i prosjektarbeid. Dette er selvsagt ikke mulig i virkelighetens verden. Derfor har det åpnet seg et *legitimeringsrom* – som gir mulighet for å kategorisere og rangere arbeidere, relasjoner og objekter, og på ny gyldiggjøre idealer som kan knyttes til plikt og behovet for styring, og knytte disse til det aktørene oppfatter som virkelig. Jeg finner at idealer fra den industrielle verdiverden igjen har blitt attraktive i IKT-organisasjoner, gjennom at de løftes inn i et kompromiss med den ukomplekse verdiverden. Kompromisset mellom den industrielle og den ukomplekse verdiverden, som jeg kaller det smidige kompromisset,[40] åpner opp for en ny disiplinering i legitimeringsrommet hvor den hjemlige verdiverden mistet sin kraft. I demokratiske samfunn hvor noe oppfattes som tilstrekkelig ubehagelig, ulønnsomt eller urettferdig, vil det alltid vokse fram kritikker som viser veien mot en bedre virkelighet. Derfor er det kanskje ikke så rart at nettopp IKT-faget, med kompleks teknologi i stadig utvikling og høye

40 Det smidige kompromisset jeg snakket om i analysen *Den gode programvareutvikleren*, består av idealer med klare røtter i den industrielle og den ukomplekse verdiverdenen. Dette alluderer til innholdet i de smidige metodologiene på et personlig og kulturelt plan.

krav til produktivitet, ga næring til en eksplisitt og helhetlig formulert kritikk av de fleksible og flytende idealene som vokste fram på 80- og 90-tallet. Men verken den ukomplekse verdiverdenen, eller kompromisset mellom den ukomplekse og den industrielle verdiverden, hindrer utbredelsen eller omfanget av prosjektorientert logikk i IKT-organisasjoner. Kompromisset bidrar derimot til å kritisere, transformere og muliggjøre forståelsen og gyldigheten av den prosjektorienterte verdiverden. Det er fremdeles rom for forestillinger om prosjekter, nettverk og fleksibilitet i dagens IKT-organisasjoner, men konseptene må bestå strengere legitimitetstester enn på 90-tallet, spesielt hvis de skal kunne knyttes til arbeid og identitet. Jeg skal vise hvordan dette fremkommer i mitt utvalg gjennom konseptet med gyldiggjøring.

9.1.1 Profesjonell identitet gyldiggjort i et spenningsfelt

Fire fagverdigheter konkurrerer om å definere *meningen* med arbeidet, eller programvarefaget, i utvalget mitt. Disse er hackerverdighet, ingeniørverdighet, hybrid verdighet og smidig verdighet. Sistnevnte består av et kompromiss mellom ukompleks og industriell logikk. Alle fagverdighetene utgjør deler av det kulturelle evalueringsrepertoaret utviklere forventes å kunne beherske i en organisasjon. En typisk utvikler forventes å være både innovativ og løsningsorientert, å kunne stille opp på møter og sette seg inn i aktuelt fagstoff, skjerme seg når det trengs og organisere og planlegge sin egen arbeidsdag. Jeg finner at alle fire fagverdighetene er viktige for at arbeidet i organisasjonene skal kunne knyttes til noe meningsfullt hos informantene, og at informantene selv reproduserer verdighetene og gjennom dette evner å endre dem. Tilsammen utgjør verdighetene kimer til det jeg forstår som en profesjonell eller yrkesfaglig identitet for programvareutviklere, som muliggjør og begrenser innovasjon og lidenskap (hackerverdighet), målstyring og effektivitet (ingeniørverdighet), kommunikasjon og nettverk (hybrid verdighet), metode og disiplinering (smidig verdighet).

Det jeg tenker på som faglig identitet blir opprettholdt i *spenningsfeltet* mellom disse fire fagverdighetene. Fagverdighetene bidrar til å begrense og skape mulige handlingsrom for utviklerne i organisasjonene. Hva som regnes som verdige praksiser er imidlertid ikke gitt. Utviklerne bidrar til å endre innholdet og betydningen av fagverdighetene gjennom legitimeringer som består av virkelighetstester og kritikker, og på den måten kan man forstå den profesjonelle eller yrkesfaglige identiteten som dynamisk og mulig å påvirke for aktører. I denne forståelsen er dermed den faglige identiteten ikke hovedsakelig basert på internaliserte idealer, personlige eller tillærte normer/verdier. De "indre" motivasjonene og internalisert kunnskap og idealer utvikles, begrenses og kritiseres i et samspill med hva aktørene til en hver tid *gyldiggjør*

som verdig i organisasjonene.

Dette er spesielt tydelig i kritikkene som rettes mot hackerverdigheten i mitt materiale. Hackerverdighet reproduseres gjennom en test av inspirasjon, engasjement, jevnlige prestasjoner og en følelse av fellesskap. Den er i stor grad knyttet til ulike indre motivasjoner. Sånn som hackerverdigheten refereres i mitt utvalg, må den reprodusere forestillinger om at det å være dyktig skyldes særegne karaktertrekk hos aktøren, karisma eller at man presterer. Noen utviklere, som for eksempel Fridtjof og Dagfinn, ser ut til å knytte en hackerverdighet sterkt til sin egen karakter eller karisma. Gleden ved å prestere og konkurrere er basert på en indre lyst eller personlig stolthet, som motiverer for økt innsats. Betydningen og verdien av *hackerverdigheten* i organisasjonssammenheng er imidlertid omstridt. Jeg finner at verdigheten som ifølge litteraturen (Himanen 2001; Castells og Himanen 2002; Levy 1984) ble tillagt hackere, er marginalisert gjennom massiv kritikk i organisasjonene jeg besøkte. "Hackere" blir i utvalget mitt kritisert som ikke-metodiske og ikke-effektive. De bidrar til "ad hoc-løsninger", de er ustrukturerte, usosiale, introverte, egoistiske og de skaper uforståelige og komplekse løsninger. Fremfor alt kritiseres deres måte å investere i arbeidet på. Forestillingen om at det er verdig i seg selv å investere mye og jobbe hardt blir langt på vei ugyldiggjort. Lyst og motivasjon blir sett på som noe som begrenser, og som dessuten representerer et mål som ikke gjelder alle (testen er urettferdig) – ikke alle har lik tilgang på lyst og motivasjon. Kritikken er såpass sterk at tester basert på indre motivasjoner og beveggrunner for å engasjere seg i arbeidet ikke består på generell basis i organisasasjonene. Hackerverdigheten har dermed liten identitetsskapende kraft utenfor noen få subkulturer i utvalget mitt. Den ser imidlertid ut til å spille en fortsatt viktig rolle for rekrutteringen til faget, og for forestillingen om at programvareutvikling – spesielt fri programvare – kan være samfunnsnyttig. Hacker-forestillingen er også viktig for hele utvalgets identitetsformasjon gjennom at den representerer et grenseobjekt de andre formene for faglig verdighet kan posisjoneres i forhold til.

Ingeniørverdigheten åpner opp for en forestillingsverden hvor det å planlegge, måle effektivitet og gjennomføre arbeidet forbindes med god fagpraksis. Denne fagverdigheten gyldiggjøres gjennom en industriell test hvor det essensielle er at iverksatte planer blir fullført, at produksjonen er stabil eller økende, og at arbeidsprosesser rasjonaliseres. Alle handlinger som støtter opp om disse idealene er per se verdige, så lenge de er lovlige. Denne verdigheten er det naturlig nok mange referanser til i organisasjonene jeg besøkte. Den utgjør en del av det internaliserte repertoaret som alle forstår, og som alle forsåvidt er enige om eller ikke synes er viktig nok til å kritisere. Det at arbeidsprosesser og -praksiser skal være effektive og rasjonelle, og at de til en viss grad kan og må måles er ukontroversielt, selv om Fridtjof og noen få andre også

har noen perspektiver som innebærer eksperimentering og fantasi. Dette bidrar til at testen gyldiggjøres og reproduseres på jevn basis. Den industrielle verdiverden brukes for eksempel til å legitimere det å ansette flere, gjøre endringer i produksjonsform, skape nye produkt, salg m.m. Arvid snakker om gleden ved å rasjonalisere når han omprogrammerer tidligere brukte "ad hoc"-løsninger på en skikkelig måte, ellers er det få i utvalget mitt som åpent knytter innhold fra ingeniørverdigheten sterkt til sin profesjonelle identitet. Mange utviklere snakker derimot om lederskap og administrasjon - høy ingeniørkompetanse – som en byrde. Lederposisjon er ikke noe som umiddelbart knyttes til høy status.

Den hybride verdigheten fremhever viktigheten av både sosial og teknologisk kompetanse. Nettverkskompetanse, det å dele ressurser og unngå konflikter gir høy verdighet ifølge de hybride idealene. Det å isolere seg, å skape konflikter, henge seg opp eller markere territorium gir lav verdighet. Få informanter kritiserer den hybride verdigheten eksplisitt, og dermed ser den ut til å bli gyldiggjort i noen grad gjennom mangel på kritikk. Det er nok muligens en begrensning for hybrid verdighet i organisasjonene at ikke alle bryr seg eller *kan* spille på dette evalueringsrepertoaret, sånn som Cihan sier så diplomatisk: *"Many software developers are mainly individualistic."* Den hybride verdigheten ser uansett ikke ut til å knyttes til høy, *profesjonell* status i utvalget mitt, noe som bekreftes av at arbeidere som referer til hybride arbeidsoppgaver gjerne er gruppeledere. Flere av informantene er ambivalente til å utføre hybride arbeidsoppgaver, og beskriver at slike oppgaver gjør at de beveger seg bort fra det faglige. Supportperson Casper arrangerer konkurranser for å motivere utviklere til å fikse flere bugs. De to forutgående verdighetene er begge knyttet til det å *skape* verdi direkte gjennom produkter, og det er dermed mulig at verdien av hybride forståelser er mindre synlig. Praksiser som muliggjør og gyldiggjør en hybrid verdighet er alle former for samarbeid, utveksling, trivselstiltak og generelt det å ha en pragmatisk holdning til arbeidet.

I komplekse organisasjoner vil ukompleks logikk kunne vokse fram og bli brukt for å gyldiggjøre behovet for forenkling. Tiltak, systemer eller automatiseringer som bidrar til forenkling blir naturlig nok løftet fram som gode løsninger for å håndtere kompleksitet. Det som skaper mer kompleksitet og støy blir rangert som mindre verdig. Den smidige verdigheten er realisert med referanser til den ukomplekse verdiverden og deler av den industrielle verdiverden. Denne kommer til uttrykk gjennom både smidige metodologier, og utviklernes uttalte behov for, og stolthet over, å forenkle det komplekse arbeidet. De smidige idealene er disiplinerende på en uventet måte i den forstand at de legger noen føringer som devaluerer det å jobbe for hardt, å legge for mye i arbeidet. Det høyverdige er å *begrense* seg, å holde igjen, legge arbeidet fra seg, gjøre akkurat nok, å kode oversiktlig og ryddig. Den smidige testen er konstruert blant annet for å

reprodusere denne *begrensningen*, i form av at man ikke skal ha for høye ambisjoner, at man skal utføre ett steg om gangen, at man aldri må investere mer enn man trenger. Moderne organisasjoner er avhengige av at arbeiderne kan realisere denne typen verdighet hvis det skulle være nødvendig.

Når en situasjon, handling eller bestemt oppgave ikke lenger gir mening fordi den ikke er inspirerende, ikke samfunnsnyttig (hackerverdighet), ikke innebærer å utnytte et mulig menneskelig potensial (industriell verdighet), eller bedre kommunikasjon (hybrid verdighet) så kan den eventuelt fortsette å gi en mening gjennom at den tilrettelegger for at relasjoner eller prosesser blir enklere, at en praksis blir ryddigere, at systemet blir mer oversiktlig eller lettere å administrere (smidig verdighet). Ryddigheten kan gjerne inspirere og skape kreativt rom, men hvis det er nødvendig kan den smidige verdigheten også legitimere at man deler opp arbeidsoppgaver i mindre deler og utfører dem en etter en, som på et samlebånd eller et flytskjema. Det smidige kompromisset kan reprodusere verdighet knyttet til arbeid i IKT-organisasjoner uavhengig av om lidenskap og individualitet er en del av formelen. Man *kan* være stolt av at man er flink til å forenkle og dele opp, men stoltheten er ikke en forutsetning for å få arbeidet gjort. Humør, innstilling og vilje spiller en rolle for å regulere koordinasjon ut fra hackerverdighet, men ikke den smidige verdigheten.

9.1.2 Grenser for indre motivasjon

De fire fagverdighetene utgjør et evalueringsrepertoar som muliggjør og begrenser handling i IKT-organisasjoner og som IKT-arbeiderne benytter når de snakker om faget sitt. Alle fire eksisterer i en balanse, de er muliggjørende og begrensende på ulike felt i ulike situasjoner gjennom at de regulerer og forsvarer ulike former for verdighet i organisasjonene. En profesjonell eller faglig identitet trenger sin profesjonsmoral, sin indre forankring. Tradisjonell profesjonsteori hevder at profesjonsmoral er basert på internaliserte, tillærte koder, symboler, språk og normer – og at dette innebærer en egen faglig stolthet som står i kontrast til ytre insentiver og marked (Grimen 2008). Abbot (1988) nyanserte dette synet, gjennom å beskrive hvordan profesjoner var gjenstand for interne og eksterne kulturelle forhandlinger, som også ga seg utslag i arbeidsdelingen innad i organisasjonene. Jeg studerer hvordan programmeringsfaget blir konstruert og transformert av aktører, med henvisning til indre idealer men også i tråd med det tilgjengelige evalueringsrepertoaret i faget, i organisasjonene og i samfunnet – gjennom gyldiggjøring av ulike legitimeringer. IKT-arbeidernes indre motivasjon vil i ulik grad la seg reprodusere ut fra de fire fagverdighetene.

Den smidige fagverdigheten er imidlertid fundert i kritikker med stor resonans i utvalgene

mine, og er direkte koblet til forestillinger om produktivitet. Den smidige verdigheten, utviklet i tråd med, og sterkt formet av industriell og ukompleks logikk, har kanskje størst sjans framover til å dominere det vi kan forstå som en profesjonsmoral med et felles verdi- og normsett. Den åpner for en type indre disiplinering, men til samme tid er det klart at mange oppfatter at rammene, målingene og begrensningene er av det gode. Anders er en av utviklerne i utvalget mitt som foretrekker mulighetene som den smidige tilnærmingen til jobben gir. Han er trygg i arbeidet, får jobben gjort, og mener at han har et rikt liv utenfor jobben. Prisen er at han ikke er like fri til å realisere seg gjennom jobben, han er nødt til å jobbe innenfor noen institusjonaliserte rammer. Dette trives Anders godt med. Han lever uansett størsteparten av livet utenfor arbeidsplassen – og som han sier selv; realiserer seg gjennom trening, konkurranser og turer med familien.

Idealer basert på lidenskap og samfunnsnytte er fremdeles viktige for enkelte utviklere i utvalget mitt, og ser spesielt ut til å spille en rolle for rekrutteringen til faget. I den grad lidenskap og lek har blitt regnet som en mer generell drivkraft i IKT-bransjen, sånn som Himanen (2001) og Himanen og Castells (2002) foreslår, har imidlertid motivasjonsformen tilsynelatende fått dårligere vilkår i komplekse organisasjoner som innfører smidig metodologier. Ifølge metodebok-korpuset er ikke individuelle ferdigheter og særpreg attraktivt for de store organisasjoner, fordi det gjør at utviklerne vanskeligere kan jobbe sammen og at produksjonen blir utsatt ved sykdom eller når arbeidstakere skifter stilling. Prestasjonskulturen som ofte karakteriserte tidligere skildringer av amatørutviklere, er ikke interessant i organisasjonssammenheng fordi det gjør planlegging og ferdigstillelse av prosjektene uforutsigbare. De opprinnelige hackernes langsiktige visjoner og ambisjoner er heller ikke ønskelige, da disse gjerne kommer i konflikt med den korte prosjekthorisonten tillatt av produksjonssyklusen, og kravet om kontinuerlig endring. Den smidige vendingen i faget, med røtter i en industriell og en ukompleks logikk, begrenser spesielt personer med en sterk indre lidenskap eller motivasjon. For disse personene mister arbeidet utfordring og spenning – og den lidenskapelige formen for mening blir vanskelig å reprodusere.

Når arbeidet intensiveres i mange moderne IKT-organisasjoner, ser det dermed ut til at det blir stikk motsatt av hva Himanen (2001) så for seg. Lidenskap, dedikasjon og indre motivasjon blir ikke styrker som kan investeres og brukes på en fornuftig måte.

9.2 Forenkling som et viktig legitimerende prinsipp i samtiden

Weber (1930), Himanen (2001), Himanen og Castells (2002), Castells (2009) og Boltanski og Chiapello (2005) foreslår at idealene de beskriver kan forklare dominerende trekk i sin samtid.

Avslutningsvis skal jeg tillate meg å skrive noen ord på dette generelle abstraksjonsnivået. Verdiverdener kan forklare formasjonen av fagretninger i et historisk perspektiv. Den ukomplekse verdiverden har vokst fram i et samspill og i spenning mellom andre verdier som er viktige i det moderne samfunnet. Det er en kontinuitet mellom logikkene i tidligere produksjonsformer og de logikkene jeg påpeker i utvalgene mine. Måten elektronisk kommunikasjon og sosiale medier har inntatt deler av næringslivet på har bidratt til å tydeliggjøre og forsterke mange av poengene fra både Castells (2009) og Boltanski og Chiapello (2005). Verdien av å knytte og etablere forbindelser til andre, å ikke binde seg og være fleksibel er udiskutabel i moderne organisasjons- og arbeidsliv. Sånn sett kan man også spørre seg hvorvidt idealene i den ukomplekse verdiverden har fått økt gyldighet i kunnskapsorganisasjoner på et bredere plan, for eksempel i universitetssystemet, innen helse eller i andre offentlige institusjoner. Den prosjektorienterte logikken forklarer hvordan meningen med arbeidet mot slutten av 70-tallet og fram mot 90-tallet gradvis ble knyttet til idealer opprinnelig ble formulert som en kritikk mot rutinearbeid, hierarki og rigiditet. Mer ansvar og autonomi i arbeidsutførelsen bidro til større dedikasjon og følelse av forpliktelse (Boltanski og Chiapello 2005:8), som i kunnskapsyrker gjerne førte til en arbeidsintensivering, at arbeiderne arbeidet *mer* enn før. Tanken om at arbeid kunne være noe lystbetont og noe man først og fremst gjorde for sin egen del er viktig i å så måte. Når engasjement i arbeid gir økonomisk gevinst, og det å være kreativ blir koblet med mestringsfølelse og anerkjennelse, oppleves jobben som identitetsformende og meningsfull for aktørene. Men denne typen jobber er i virkelighetens verden ikke tilgjengelig for alle. Det moderne samfunnet og moderne organisasjoner i dag har et behov for å få utført arbeid som i mindre grad kan oppleves som lystbetont, akkurat som før.

For å forstå moderne organisasjoner og det moderne samfunnet er det også et viktig poeng at mennesket kognitivt sett ikke har mulighet til å håndtere den voksende informasjonsmengden uten samtidig økt bruk av kognitive eller kulturelle former for filtrering. Suksess i en verden som består av nettverk er betinget av å beherske en egen ukompleks logikk, som forbinder det høyverdige og det gode med å skaffe seg oversikt, filtrere, å stenge ute, balansere og prioritere oppgaver og skjerme seg. Ifølge denne logikken blir det legitimt å stenge ute både helheten og detaljer, så lenge det bidrar til nåværende situasjonen blir bedre. Det pragmatiske rammeverket og forenklingens logikk foreslår en mulig tolkning på hvordan og hvorfor mange kunnskapsarbeidere i dag ser ut til å tilpasse seg og i noen grad omfavne de nye arbeidsregimene. Visst er reformer og metodologier som disiplinerer arbeiderne i dag viktige – men de nye styringssystemene kunne aldri blitt utbredt uten at mange forbinder dem til noe som faktisk fungerer og gjør hverdagen bedre. Utvalget mitt antyder at ukomplekse idealer spiller en rolle for

å gjøre industrielle idealer, med referanse til metoder, strukturer og måling, attraktive igjen i IKT-organisasjoner. Det gjenstår imidlertid å utforske hvorvidt lignende tendenser kan observeres i andre typer kunnskapsorganisasjoner. Spesielt interessant vil det være å studere hvorvidt fagidentiteter knyttet til hardt arbeid, særegne karaktertrekk, originale ideer eller prestasjoner har gode vilkår i dag i ulike organisasjonsformer og ulike rekrutteringsprosesser.

På makroplan vil det også være svært interessant å studere konsekvenser av moderne filtrerings- og måleteknologier, og spesielt de automatiserte filtrene som i økende grad brukes Internett, basert på algoritmer eller formler (Pariser 2011). Innen mediebransjen har redaktørene forholdt seg til redaktørplakaten eller lignende etiske retningslinjer som kan diskuteres og kritiseres. Filtrene som rangerer søkeresultat eller oppslag i sosiale medier er skjult for offentligheten, og er ikke underlagt verken politisk eller demokratisk kontroll. Dette har betydning i en verden hvor folk leser færre tradisjonelle aviser (Fogt og Elvestad 2010), og baserer mer av sin verdensforståelse på informasjon fra vennekretsen eller diverse nettjenester.

Spiller ukomplekse legitimeringer noen rolle i den demokratiske offentligheten i dag? For å underbygge rødgrønne partiers argumenter om å utvikle felles standarder, eller for å støtte høyresidens avpolitisering av beslutninger og oppgaver, hevdes det (blant andre argumenter) at det politiske systemet må gjøres enklere, og at det som en følge av dette blir mer forståelig, inkluderende og *rettferdig*. Boltanski og Chiapello (2005:126) viste at den prosjektorienterte logikken åpnet for å ekskludere mennesker ut fra et kriterie om deltakelse – den som ikke investerer i nettverket og knytter kontakter har heller ingen sjans til å oppnå verdighet. *Den som ikke vil eller kan investere, eksisterer ikke.* Den ukomplekse logikken åpner opp for en annen form for ekskludering, spesielt hvis den kombineres med industriell logikk; *masse-eksklusjon gjennom automatisering og filtrering*. Schengen-avtalen, som i dag gjelder 25 europeiske land, ble legitimert med referanse til forenkling på to måter. For det første skulle den fjerne passkontroll mellom medlemslandene, og dermed gjøre mobilitet (prosjektorientert logikk) mellom medlemslandene enklere, og for det andre skulle den etablere en felles id-database (SIS) sånn at lagret personinformasjon om blant annet asylsøkere kunne være tilgjengelig på tvers av medlemslandene. Databasen inneholder blant annet informasjon om navn, fysiske karakteristikker, eventuelle anmeldelser og eventuelle anmerkninger som for eksempel manglende identitetsformasjon. Hvis en person blir avvist i et medlemsland, lagres beslutningsgrunnlaget sånn at vedkommende blir avvist på samme grunnlag i de andre medlemslandene.

Mens det på den ene siden er riktig at det moderne samfunnet kan forstås ut fra kognitive forståelser som tillegger nettverket verdi, og hvor kommunikasjon, varer og valuta fremstår og

blir oppfattet som grenseløse, er det på den andre siden risset inn nye grenser som hindrer sosial og geografisk mobilitet (se f.eks. Moses 2006) og utvikling av en større global bevissthet. Disse henter sin legitimitet fra idealer som ikke hører hjemme i informasjonssamfunnet, eller den prosjektorienterte verden – men i det ukomplekse, bak et kognitivt, men refleksivt og eksplisitt uttalt skjold, som skal beskytte mot følelsen av risiko og usikkerhet knyttet til det komplekse og uoversiktlige. Det er dermed all grunn til å være oppmerksom når ukompleks logikk blir fremmet i legitimering. Potensielt skjuler grammatikken metaforer fra andre verdiverdener: effektivisering, rasjonalisering, nedskjæring og automatisering. Industrielle argumentasjoner kan enkelt kritiseres fra både den kollektive og den artistiske verdiverdenen, men hvem klarer å argumentere mot at kompleksiteten som følger av moderne yrker og livsstil trenger forenkling?

10. Litteratur

Abbott, Andrew. 1988. *The system of professions : an essay on the division of expert labor*. Chicago: University of Chicago Press.

Abbott, Andrew. 1995. «Things of boundaries - Defining the Boundaries of Social Inquiry». *Social Research* 62(4):857–882.

Aktouf, Omar. 1996. *Traditional Management and Beyond - A Matter of Renewal*. Morin - Montreal Canada.

Allen, David. 2002. *Getting Things Done: How to Achieve Stress-free Productivity*. Piatkus Books.

Andersen, Gisle, og Marte Mangset. 2012. «Er forestillingen om det egalitære Norge resultatet av en målefeil?» *Tidsskrift for samfunnsforskning* (2).

Arbeidstilsynet. 2011. «Nyhet: Arbeidstilsynet vil føre tilsyn med legers arbeidstid i helseforetakene». *Arbeidstilsynet*. Hentet september 5, 2011 (http://www.arbeidstilsynet.no/nyhet.html?tid=229508).

Armstrong, Karen. 2001. *Buddha*. 1st utg. Viking.

Armstrong, Michael. 2009. *Armstrong's Handbook of Performance Management: An Evidence-Based Guide to Delivering High Performance*. 4. utg. Kogan Page.

Baldry et al., Christopher, red. 2007. *The meaning of work in the new economy*. Basingstoke: Palgrave Macmillan.

Beck, John, og Michael F. D. Young. 2005. «The assault on the professions and the restructuring of academic and professional identities: a Bernsteinian analysis». *British Journal of Sociology of Education* 26(2):183.

Beck, Kent, og Cynthia Andres. 2004. *Extreme Programming Explained: Embrace Change*. 2. utg. Addison-Wesley Professional.

Bell, Daniel. 1974. *The Coming of Post-Industrial Society: A Venture in Social Forecasting*. 1. utg. New York: Basic Books.

Bell, Daniel. 1976. *The Coming of Post-Industrial Society: A Venture in Social Forecasting*. 2. utg. New York: Basic Books.

Benatouil, Thomas. 1999. «A Tale of Two Sociologies: The Critical and the Pragmatic Stance in Contemporary French Sociology». *European Journal of Social Theory* 2(3):379–396.

Benner, Chris. 2002. *Work in the new economy: flexible labor markets in Silicon Valley*. Wiley-Blackwell.

Berg, Vivian Anette Lagesen. 2000. *Firkanter og rundinger. Kjønnskonstruksjoner blant kvinnelige dataingeniørstudenter ved NTNU.* Trondheim: NTNU, senter for feminist og kjønnsforskning.

Berkun, Scott. 2005. *The Art of Project Management (Theory in Practice)*. 1. utg. O'Reilly Media.

Bernstein, Basil B. 2000. *Pedagogy, symbolic control, and identity: theory, research, critique*. Rowman & Littlefield.

Bjerke, Paul. 2004. *Det nye arbeidslivet? En rapport om IKT-myter.* De Facto. Kunnskapssenter for fagorganiserte.

Blokker, Paul. 2011. «Pragmatic sociology: Theoretical evolvement and empirical application». *European Journal of Social Theory* 14(3):251 –261.

Boltanski, Luc., og Eve. Chiapello. 2005. *The new spirit of capitalism*. London; New York: Verso.

Boltanski, Luc, og Laurent Thévenot. 1999. «The Sociology of Critical Capacity». *European Journal of Social Theory* 2(3):359–377.

Boltanski, Luc., og Laurent Thévenot. 2006. *On justification : economies of worth*. Princeton: Princeton University Press.

Bourdieu, Pierre. 1984. *Distinction : a social critique of the judgement of taste*. Cambridge Mass.: Harvard University Press.

Brooks, Jr, Frederick P. 1987. «No Silver Bullet: Essence and Accidents of Software Engineering». *IEEE Computer*.

Bryson, Bethany. 1996. «'Anything But Heavy Metal': Symbolic Exclusion and Musical Dislikes». *American Sociological Review* 61(5):884–899.

Bush, C. M., og L. L. Schkade. 1985. «In search of the perfect programmer». *Datamation* 31(6):128–132.

Capretz, Luiz Fernando. 2003. «Personality types in software engineering». *International Journal of Human-Computer Studies* 58(2):207–214.

Castells, Manuel. 2001. *The Internet galaxy : reflections on the Internet, business, and society*. Oxford; New York: Oxford University Press.

Castells, Manuel. 2009. *The Rise of the Network Society: The Information Age: Economy, Society, and Culture Volume I*. 2. utg. Oxford: Wiley-Blackwell.

Castells, Manuel, og Pekka. Himanen. 2002. *The information society and the welfare state : the Finnish model*. Oxford: Oxford University Press.

Cloutier, C, og Ann Langley. 2007. «Competing rationalites in organizations: a theoretical and methodological overview». *Cahiers de recherce du Ge'PS* 3(1).

Cohn, Mike. 2005. *Agile Estimating and Planning*. Prentice Hall.

Cohn, Mike. 2009. *Succeeding with Agile: Software Development Using Scrum*. 1. utg. Addison-Wesley Professional.

Cokins, Gary. 2009. *Performance Management: Integrating Strategy Execution, Methodologies, Risk, and Analytics*. John Wiley & Sons.

Corneliussen, Hilde. 2003. «Male positioning strategies in relation to computing». i *He, She and IT revisited : New Perspectives On Gender In The Information Age*, redigert av Merete Lie. Oslo: Gyldendal Akademiske.

Corneliussen, Hilde. 2002. «The multi-dimensional stories of the gendered users». i *Researching ICTs in Context. InterMedia Report 3/2002*, redigert av Andrew Morrison. Oslo: Intermedia.

Dahlbom, Bo. 1997. «The Future of Our Profession». *Communications of the ACM.* 40(6):80.

Dahrendorf, Ralf. 1959. *Class and Class Conflict in Industrial Society*. Stanford, CA: Stanford University Press.

Demsetz, Harold. 1967. «Towards a theory of property rights». *American Economic Review* 57:347–359.

Denis, Jean-Louis, Ann Langley, og Linda Rouleau. 2007. «Strategizing in pluralistic contexts: Rethinking theoretical frames». *Human Relations* 60(1):179 –215.

Desrosières, Alain. 1998. *The Politics of Large Numbers: A History of Statistical Reasoning*. 1. utg. Harvard University Press.

DiBona, Sam Ockman, og Stone. 1999. «Open sources voices from the open source revolution».

Dijkstra, Edsger W. 1993. «There is still a war going on». Hentet august 12, 2011 (http://www.cs.utexas.edu/users/EWD/transcriptions/EWD11xx/EWD1165.html).

Douglas, Mary. 1966. *Purity and Danger: An Analysis of the Concepts of Pollution and Taboo*. 1st Edition. Routledge & Kegan Paul PLC.

Drucker, Peter F. 1999. «Knowledge Management - Knowledge-Worker Productivity: The Biggest Challenge». *California management review.* 41(2):79.

Drucker, PF. 1992. «The new society of organizations.» *Harvard business review* 70(5):95–104.

Duarte, Nancy. 2008. *slide:ology: The Art and Science of Creating Great Presentations: The Art and Science of Presentation Design*. 1. utg. O'Reilly Media.

Durkheim, Emile. 1992. *Professional Ethics and Civic Morals*. 2. utg. Routledge.

Durkheim, Emile. 1965. *The elementary forms of the religious life.* New York: Free Press.

Elster, Jon. 1993. *Local Justice: How Institutions Allocate Scarce Goods & Necessary Burdens*. Russell Sage Foundation Publications.

Epstein, C F. 1992. «Tinker-bells and Pinups: The Construction and Reconstruction of Gender Boundaries at Work». i *Cultivating differences: symbolic boundaries and the making of inequality*,

redigert av Michèle Lamont og Marcel Fournier. Chicago: University of Chicago Press.

Erickson, Bonnie. 1996. «Culture, Class, and Connections». *The American Journal of Sociology* 102(1):217–251.

Etzioni, Amitai. 1969. *Semiprofessionals and Their Organization: Teachers, Nurses, Social Workers*. Free Press.

Faulkner, W. 2000. «The Power and the Pleasure? A Research Agenda for 'Making Gender Stick' to Engineers». *SCIENCE TECHNOLOGY AND HUMAN VALUES* 25:87–119.

Fewsnet. 2011. *Famine thresholds surpassed in three new areas of southern Somalia*. Somalia: Famine Early Warning System Network Hentet august 15, 2011 (http://www.fsnau.org/downloads/FSNAU_FEWSNET_020811_press_release_030811.pdf).

Field, Tom. 1997. «When bad things happen to good projects». *CIO Magazine* 11(2):54–62.

Fitzgerald, Brian. 2006. «ISSUES AND OPINIONS - The Transformation of Open Source Software». *MIS quarterly : management information systems.* 30(3):587.

Fogel, Karl. 2005. *Producing open source software : how to run a successful free software project*. Sebastopol, CA: O'Reilly.

Fogt, Anne, og Eiri Elvestad. 2010. *Trenger vi aviser når vi har Facebook?* Oslo: Høyskoleforlaget.

Foucault, Michel. 1994. *The Order of Things: An Archaeology of the Human Sciences*. First Edition. Vintage.

Freidson, E. 1982. «Occupational autonomy and labor market shelters». S. 39–54 i *Varieties of Work*, redigert av P. L. Steward og M. G. Cantor. Beverly Hills: Sage.

Frenkel, Stephen. 1999. *On the front line : organization of work in the information economy*. Ithaca, New York: ILR press.

Friedman, Andrew. 1989. *Computer systems development : history, organization, and implementation*. New York: Wiley.

Fronda, Yannick, og Jean-Luc Moriceau. 2008. «I am not your hero: change management and culture shocks in a public sector corporation». *Journal of Organizational Change Management* 21(5):589–609.

Gansmo, Helen Jøsok. 2002a. «Samfunnsproblemet 'jenter og data'». *Kvinneforskning* (2):10–25.

Gansmo, Helen Jøsok. 2002b. «Seduced by numbers?» i *Researching ICTs in Context. InterMedia Report 3/2002*, redigert av Andrew Morrison. Oslo: Intermedia.

Gansmo, Helen Jøsok, Vivian Anette Lagesen, og Knut H. Sørensen. 2003. «Forget the Hacker? A Critical re-appraisal of Norwegian studies of gender and ICT». i *He, She and IT revisited : New Perspectives On Gender In The Information Age*, redigert av Merete. Lie. Oslo: Gyldendal.

Giacalone, Robert, og Dr. Jerald Greenberg. 1997. *Antisocial Behavior in Organizations*. Sage Publications, Inc.

Giblin, Mary, Padraig Brennan, og Chris Exton. 2010. «Introducing Agile Methods in a Large Software Development Team: The Developers Changing Perspective». *Agile Processes in Software Engineering and Extreme Programming* 48(2):184–189.

Glover, Ian, Wendy Currie, og Stephen Ackroyd. 2000. «The triumph of hierarchies over markets: information system specialists in the current context». S. 267–305 i *Professions at Bay*, redigert av Ian Glover og Michael Hughes. Aldershot: Ashgate Publishing.

Goldfarb, Brent D., David Kirsch, og David A. Miller. 2006. «Was There Too Little Entry During the Dot Com Era?» *SSRN eLibrary*. Hentet september 8, 2011 (http://papers.ssrn.com/sol3/papers.cfm?abstract_id=871210).

Goles, Tim, Stephen Hawk, og Kate M. Kaiser. 2008. «Information technology work force skills: The Software and IT Services Provider Perspective». *Information Systems Frontiers* 10(2):179–194.

Graham, Paul. 2004. *Hackers and Painters: Big Ideas from the Computer Age*. O'Reilly Media.

Gregg, Richard B. 1936. *Value of Voluntary Simplicity*. Pennsylviania: Pendle Hill Pubns.

Grimen, Harald. 2008. «Kap. 8: Profesjon og profesjonsmoral». S. 144–160 i *Profesjonsstudier*, redigert av A. Molander og L. I. Terum. Oslo: Universitetsforlaget.

Guerrier, Yvonne, Christina Evans, Judith Glover, og Cornelia Wilson. 2009. «'Technical, but not very....': constructing gendered identities in IT-related employment». *Work Employment Society* 23(3):494–511.

Gyurky, Szabolcs de. 2006. *The Cognitive Dynamics of Computer Science: Cost-Effective Large Scale Software Development*. 1. utg. Wiley-IEEE Computer Society Pr.

Hacking, Ian. 2000. *The Social Construction of What?* Harvard University Press.

Halrynjo, Hege Eggen. 2007. «Alltid beredt? Arbeids- og familiedilemmaer i møte med formelle og uformelle spilleregler i et stort konsern». i *Arbeidslivets klemmer*, redigert av Elin Kvande og Bente Rasmussen. Bergen: Fagbokforlaget.

Himanen, Pekka. 2001. *The hacker ethic, and the spirit of the information age*. New York: Random House.

Von Hippel, Eric, og Georg Von Krogh. 2003. «Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science». *Organization Science* 14(2):209–223.

Hjelseth, Arve. 2011. «Trening, knefall og forfall». *Samtiden* (2).

Hochschild, Arlie Russell. 2001. *The Time Bind: When Work Becomes Home and Home Becomes Work*. 1st utg. Holt Paperbacks.

Hyman, Jeff, Dora Scholarios, og Chris Baldry. 2005. «Getting on or getting by?» *Work, Employment & Society* 19(4):705 –725.

Håpnes, Tove, og Bente Rasmussen. 1991. «The production of male-power in computer science». i *Women, Work and computerization. Understanding and overcoming bias in work and education.*, redigert av I. Eriksson, B. Kitchenham, og K. Tjidens. Amsterdam.

Jagd, Søren. 2011. «Pragmatic sociology and competing orders of worth in organizations». *European Journal of Social Theory* 14(3):343 –359.

Jarzabkowski, Paula, og Evelyn Fenton. 2006. «Strategizing and Organizing in Pluralistic Contexts». *Long Range Planning* 39(6):631–648.

Jeffries, Ron, Ann Anderson, og Chet Hendrickson. 2000. *Extreme Programming Installed*. 1. utg. Addison-Wesley Professional.

Jordfald, Bård, og Dag Olberg. 2002. *IKT-sektoren – perspektiver på sysselsetting, arbeidsmiljø og interesseorganisering*. Fafo Hentet oktober 15, 2010 (http://www.fafo.no/pub/rapp/391/index.htm).

Jung, Carl Gustav. 1976. *Psychological Types*. A Revision by R. F. C. Hull of the translation by H. G. Baynes. Princeton: Princeton University Press.

Keirsey, David. 1998. *Please Understand Me II: Temperament, Character, Intelligence*. 1st utg. Prometheus Nemesis Book Company.

Keirsey, David. 1995. *Portraits of Temperament*. 3rd utg. Prometheus Nemesis Book Company.

Kelley, Robert E. 1985. *The Gold-Collar Worker: Harnessing the Brainpower of the New Work Force*. 1St Edition. Addison-Wesley.

Kirknes. 2011. «Universitetet i Oslo satser på Smidig». *Computerworld* Hentet oktober 20, 2011 (http://www.idg.no/computerworld/article185651.ece).

Kirkpatrick, David. 2010. *The Facebook Effect: The Inside Story of the Company That Is Connecting the World*. Simon & Schuster.

Kraft, Phillip. 1979. «The Industrialization of Computer Programming: From Programming to 'Software Production'». i *Case studies on the labor process*, redigert av Andrew Zimbalist. New York: Monthly Review Press.

Kraatz, Mattheew, og E Block. 2008. «Organizational Implications of Institutional Pluralism». S. 243–275 i *Handbook of Organizational Institutionalism*, redigert av R Greenwood, C Oliver, og K Sahlin-Andersson. London: Sage Publications.

Lagesen, Vivian Anette. 2005. *Extreme make-over? : the making of gender and computer science*. Trondheim: NTNU.

Lagesen, Vivian Anette. 2007. «The Strength of Numbers: Strategies to Include Women into Computer Science». *Social Studies of Science* 37(1):67–92.

Lagesen, Vivian Anette, og Knut H. Sørensen. 2009. «Walking the line? The enactment of the social/technical binary in software engineering». *Engineering Studies* 1(2):129.

Lakhani, Karim, og Robert Wolf. 2005. «Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects». i *Perspectives on Free and open Source Software*, redigert av Joseph Feller. Cambridge, Mass.: MiT Press.

Lamont, Michèle. 1992. *Money, morals, and manners : the culture of the French and American upper-*

middle class. Chicago: University of Chicago Press.

Lamont, Michèle. 2001. «Symbolic Boundaries (General)». Hentet november 5, 2009 (http://educ.jmu.edu/~brysonbp/symbound/papers2001/LamontEncyclo.html).

Lamont, Michèle. 2000. *The dignity of working men : morality and the boundaries of race, class, and immigration*. New York, N.Y.; Cambridge, Mass.: Russell Sage Foundation ; Harvard University Press.

Lamont, Michèle, og Virág Molnár. 2002. «The Study of Boundaries in the Social Sciences». *Annual Review of Sociology* 28(1):167–195.

Lamont, Michèle, og Laurent Thévenot. 2000. *Rethinking comparative cultural sociology : repertoires of evaluation in France and the United States*. Cambridge, UK; New York: Cambridge University Press.

Langsether, Helene. 2001. *Behov og barrierer for jenter på informatikkstudiet. SKF-rapport 3/2001*. Trondheim: NTNU, senter for feminist- og kjønnsforskning.

Larson, Magali. 1977. *The rise of professionalism : a sociological analysis*. Berkeley: University of California Press.

Lederer, Albert L., og Jayesh Prasad. 1992. «Nine management guidelines for better cost estimating». *Communications of the ACM* 35:51–59.

Lerner, Josh, og Mark Schankerman. 2010. *The comingled code : open source and economic development*. Cambridge Mass.: The MIT Press.

Levy, Steven. 1984. *Hackers : heroes of the computer revolution*. Garden City, N.Y.: Anchor Press/Doubleday.

van der Lippe, Tanja. 2007. «Dutch workers and time pressure: household and workplace characteristics». *Work, Employment & Society* 21(4):693 –711.

Lowenstein, Roger. 2004. *Origins of the Crash: The Great Bubble and Its Undoing*. 1St Edition. Penguin Press HC, The.

Lui, Kim Man, og Keith C. C. Chan. 2008. *Software Development Rhythms: Harmonizing Agile Practices for Synergy*. 1. utg. Wiley-Interscience.

Laanti, Maarit, Outi Salo, og Pekka Abrahamsson. 2011. «Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation». *Information and Software Technology* 53(3):276–290.

Martinez, Luis, Antonia Rodrguez-Diaz, Guillermo Licea, og Juan R. Castro. 2010. «Big Five Patterns for Software Engineering Roles Using an ANFIS Learning Approach with RAMSET». i *Advances in Soft Computing*, vol. 6438/2010, *Lecture Notes in Computer Science*.

Martin, Robert C. 2008. *Clean Code: A Handbook of Agile Software Craftsmanship*. 1. utg. Prentice Hall.

Mauss, Marcel, og Ian Cunnison. 1970. *The gift: forms and functions of exchange in archaic societies*. Taylor & Francis.

May, Christopher. 2000. *A global political economy of intellectual property rights : the new enclosures?* London; New York: Routledge.

Merton, Robert K. 1979. *The Sociology of Science: Theoretical and Empirical Investigations*. University Of Chicago Press.

Moløkken-Østvold, Kjetil, og Magne Jørgensen. 2005. «A Comparison of Software Project Overruns-Flexible versus Sequential Development Models». *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING* 31(9):754–754–766.

Moses, Jonathon W. 2006. *International Migration: Globalization's Last Frontier*. large type edition. Zed Books.

Myers, Isabel Briggs. 1995. *Gifts Differing: Understanding Personality Type*. 2nd ed. Davies-Black Publishing.

Myers, Isabel Briggs. 1998. *Introduction to Type: A Guide to Understanding Your Results on the Myers-Briggs Type Indicator*. 6th utg. Center for Applications of.

Nachi, Mohamed. 2006. *Introduction à la sociologie pragmatique*. Armand Colin.

Nowack, K. 1996. «Is the Myers Briggs Type Indicator the Right Tool to Use? Performance in Practice». *Performance in Practice, American Society of Training and Development* 6.

Olson, Mancur. 1971. *The Logic of Collective Action: Public Goods and the Theory of Groups, Second printing with new preface and appendix*. Revised. Harvard University Press.

Ophir, Eyal, Clifford Nass, og Anthony D. Wagner. 2009. «Cognitive control in media multitaskers». *Proceedings of the National Academy of Sciences*. Hentet (http://www.pnas.org/content/early/2009/08/21/0903620106.abstract).

Pariser, Eli. 2011. *The Filter Bubble: What the Internet Is Hiding from You*. London: The Penguin Press.

Parkin, Frank. 2001. *The Social analysis of class structure*. London: Routledge.

Parsons, Talcott. 1951. *The Social System*. New York: Free Press.

Pavalko, Ronald. 1971. *Sociology of Occupations and Professions*. Ithasca IL: F E Peacock Pub, 2d. ed.

Perlow, Leslie A. 1998. «Boundary Control: The Social Ordering of Work and Family Time in a High-Tech Corporation». *Administrative Science Quarterly* 43(2):328–357.

Perrons, Diane, Colette Fagan, Linda McDowell, Kath Ray, og Kevin Ward. 2007. *Gender Divisions and Working Time in the New Economy: Changing Patterns of Work, Care and Public Policy in Europe and North America*. Edward Elgar Publishing.

Pichler, Roman. 2010. *Agile Product Management with Scrum: Creating Products that Customers Love (Addison-Wesley Signature Series*. 1. utg. Addison-Wesley Professional.

Poppendieck, Mary, og Tom Poppendieck. 2009. *Leading Lean Software Development: Results Are not the Point*. 1. utg. Addison-Wesley Professional.

Power, Michael. 1997. *The audit society: rituals of verification*. Oxford University Press.

Rao, Leena. 2011. «Zuck Confirms That Facebook Now Has 750 Million Active Users». *TechChrunch*. Hentet august 16, 2011 (http://techcrunch.com/2011/07/06/zuck-confirms-that-facebook-now-has-750-million-users/).

Rasmussen, Bente, og Birgitte Johansen. 2002. «Kunnskapsarbeidere i dot.com-økonomien». *Tidsskrift for Arbejdsliv* 4(2):25–44.

Rasmussen, Bente, og Birgitte Johansen. 2005. «Trick or treat? Autonomy as control in knowledge work». i *Management, labour process and software development: reality bytes*, *Routledge research in employment relations, 13*, redigert av Rowena Barrett. New York: Routledge, 2005.

Raymond, Eric. 2003. «How to Become a Hacker». *Database and network journal.* 33:8–9.

Raymond, Eric. 2001. *The cathedral and the bazaar : musings on Linux and Open Source by an accidental revolutionary*. Beijing; Cambridge, Mass.: O'Reilly.

Reay, Trish, og C.R. Hinings. 2009. «Managing the Rivalry of Competing Institutional Logics». *Organization Studies* 30(6):629 –652.

Reich, Robert. 1991. *The work of nations : preparing ourselves for 21st-century capitalism*. 1st ed. New York: A.A. Knopf.

Reilly, Brendan M, Arthur T Evans, Jeffrey J Schaider, Krishna Das, mfl. 2002. «Impact of a Clinical Decision Rule on Hospital Triage of Patients With Suspected Acute Cardiac Ischemia in the Emergency Department». *JAMA: The Journal of the American Medical Association* 288(3):342 – 350.

Reilly, Brendan M, Arthur T Evans, Jeffrey J Schaider, og Yue Wang. 2002. «Triage of patients with chest pain in the emergency department: a comparative study of physicians' decisions». *The American Journal of Medicine* 112(2):95–103.

Reynolds, Garr. 2007. *Presentation Zen: Simple Ideas on Presentation Design and Delivery*. 1. utg. New Riders.

Ritzer, George. 2004. *The McDonaldization of society*. Rev. new century ed. Thousand Oaks Calif.: Pine Forge Press.

Roch, Joanne M. 2004. «A new way to look at the integration challenge: The reconciliation of collective representations». S. 23–50 i *Advances in Mergers & Acquisitions*, vol. 4. Bingley: Emerald.

Sandler, Todd. 1992. *Collective Action: Theory and Applications*. University of Michigan Press.

Scarbrough, Harry. 1999. «Knowledge as Work: Conflicts in the Management of Knowledge Workers.» *Technology Analysis and Strategic Management* 11(1):5–16.

Schumacher, E. 1973. *Small is beautiful : economics as if people mattered*. New York: HarperPerennial.

Sennett, Richard. 1998. *The Corrosion of Character: The Personal Consequences of Work in the New Capitalism*. 1. utg. W. W. Norton & Company.

Shi, David Emory. 2001. *The Simple Life: Plain Living and High Thinking in American Culture*. University of Georgia Press.

Silber, Ilana Friedrich. 2003. «Pragmatic Sociology as Cultural Sociology». *European Journal of Social Theory* 6(4):427 –449.

Simmel, Georg. 1950. *The sociology of Georg Simmel : [by] Georg Simmel*. New York: Free Press.

Simon, Herbert A. 1971. «Designing Organizations for an Information-Rich World». S. 37–72 i *Computers, communications, and the public interest*, redigert av Martin Greenberger. The Johns Hopkins Press.

Skarpenes, Ove. 2007. «Den 'legitime kulturens' moralske forankring». *Tidsskrift for samfunnsforskning* (04):532–560.

Smith, D. C. 1989. «The personality of the systems analyst: an investigation». *ACM SIGCPR Computer Personnel* 12:12–14.

Spaeth, Sebastian, Matthias Stuermer, og Georg Von Krogh. 2010. «Enabling knowledge creation through outsiders: towards a push model of open innovation». *International Journal of Technology Management* 52(3/4):411 – 431.

SSB. 2010a. «Fritid og kultur». *SSB.no*. Hentet (http://www.ssb.no/sosind/tab-2010-12-06-09.html).

SSB. 2010b. «Fullførte høyeregradsstudier i 2008/2009, etter antall år siden studenten første gang var registrert i høyere utdanning, kjønn og studiets varighet.» *SSB.no*. Hentet (http://www.ssb.no/emner/04/02/40/hugjen/tab-2010-11-25-02.html).

Stallman, Richard. 1999. «The GNU Operating System and the Free Software Movement». i *Open Sources: voices from the open source revolution*, redigert av Chris Dibona, Sam Ockman, og Mark. Stone. Cambridge: O'Reilly.

Standish. 1994. *Chaos, Technical Report*. The Standish Group International Inc.

Star, Susan Leigh, og James R. Griesemer. 1989. «Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39». *Social Studies of Science* 19(3):387–420.

Stonier, Tom. 1983. *The Wealth of Information: Profile of the Post-industrial Society*. London: Thames Methuen.

Strathern, Marilyn. 2000. *Audit cultures: anthropological studies in accountability, ethics, and the academy*. Routledge.

Strathern, Marilyn. 1988. *The gender of the gift : problems with women and problems with society in Melanesia*. Berkeley: University of California Press.

Striebeck, Mark. 2006. «Ssh! We Are Adding a Process...» S. 185–193 i *Proceedings of the conference on AGILE 2006*. Washington, DC, USA: IEEE Computer Society.

Stuedahl, Dagny. 1997. *Jenter og informatikkstudiet – en rapport om jenters studiesituasjon ved Institutt*

for Informatikk, UiO. Oslo: Kirke og utdanningsdepartementet.

Stuermer, Matthias, Sebastian Spaeth, og Georg Von Krogh. 2009. «Extending private-collective innovation: a case study». *R & D Management* 39(2):170–191.

Svendsen, Roy Hilmar, og Gunhild Agdesteen. 2011. «Studenter orker ikke skrive for hånd». *NRK.no*, mai 26 Hentet (http://www.nrk.no/nyheter/distrikt/hordaland/1.7648562).

Svensson, Lennart, og Julia Evetts. 2010. *Sociology of professions : continental and Anglo-Saxon traditions*. Göteborg: Daidalos.

Thévenot, Laurent. 2001a. «Organized Complexity». *European Journal of Social Theory* 4(4):405 –425.

Thévenot, Laurent. 2001b. «Pragmatic Regimes Governing the Engagement with the World». i *The practice turn in contemporary theory*, redigert av Theodore R. Schatzki, Karin Knorr-Cetina, og Eike von Savigny. New York: Routledge.

Thévenot, Laurent. 2007. «The Plurality of Cognitive Formats and Engagements». *European Journal of Social Theory* 10(3):409 –423.

Thévenot, Laurent, Michael Moody, og Claudette Lafaye. 2000. «Forms of valuing nature: arguments and modes of justification in French and American environmental disputes». S. 229–272 i *Rethinking Comparative Cultural Sociology*, *Cambridge Cultural Social Studies*, redigert av Michèle Lamont og Laurent Thévenot. Cambridge University Press.

Thompson, Paul. 2003. «Disconnected Capitalism: Or Why Employers Can't Keep Their Side of the Bargain». *Work, Employment & Society* 17(2):359–378.

Thompson, Paul, og David McHugh. 2001. *Work Organisations*. 3rd Revised edition. Palgrave Macmillan.

Thoreau, Henry David. 2008. *Walden*. Reissue. Oxford Paperbacks.

Thornton, Patricia H. 2004. *Markets from culture: institutional logics and organizational decisions in higher education publishing*. Stanford University Press.

Thornton, Patricia H., og W Ocasio. 2008. «Institutional logics». S. 99–129 i *Handbook of Organizational Institutionalism*, redigert av R Greenwood, C Oliver, og K Sahlin-Andersson. London: Sage Publications.

Turkle, Sherry. 1984. *The second self : computers and the human spirit*. New York: Simon and Schuster.

Turkle, Sherry, og Seymour Papert. 1990. «Epistemological Pluralism: Styles and Voices within the Computer Culture.» *Signs: Journal of Women in Culture and Society* 16(1):128–57.

Vallas, Steven Peter. 2001. «Symbolic boundaries and the new division of labor: Engineers, workers and the restructuring of factory life». *Research in Social Stratification and Mobility* 18:3–37.

Veblen, Thorstein. 2004. *The Theory of the Leisure Class*. Neeland Media LLC.

Wagner, Peter. 1999. «After Justification». *European Journal of Social Theory* 2(3):341 –357.

Wagner, Peter. 2001. *A history and theory of the social sciences: not all that is solid melts into air*. SAGE.

Watts, Jacqueline H. 2009. «'Allowed into a Man's World' Meanings of Work-Life Balance: Perspectives of Women Civil Engineers as 'Minority' Workers in Construction». *Gender, Work & Organization* 16(1):37–57.

Weber, Max. 1978. *Economy and Society: An Outline of Interpretive Sociology*. University of California Press.

Weber, Max. 2001. *The Protestant Ethic and the Spirit of Capitalism*. 2nd edition. London: Routledge.

Weber, Steve. 2004. *The success of open source*. Cambridge, MA: Harvard University Press.

West, Dave, og Tom Grant. 2010. *Agile development: Mainstream Adoption has Changed Agility*. Forrester Research, Inc.

Westenholz, Ann. 2006. «Identity, Times and Work». *Time & Society* 15(1):33 –55.

Wilhelmsen, Marit. 2010. *Holdninger til norsk bistand*. Oslo-Kongsvinger: Statistisk sentralbyrå Hentet august 30, 2011 (http://www.ssb.no/emner/00/01/30/rapp_uhjelphold/rapp_201113/main.html).

Zeitlyn, David. 2003. «Gift economies in the development of open source software: anthropological reflections». *Research Policy* 32(7):1287–1291.

Aakvaag, Gunnar C. 2011. «Å sette samtiden på begrep: Noen utfordringer for en samtidsdiagnostisk sosiologi». *Sosiologisk tidsskrift* 19(3).