

Råd og regler for projektarbejde på 1. semester

Mads Rosendahl

31. august 2004

Projektarbejde spiller en central rolle i vore uddannelser. Cirka halvdelen af tiden går med projektarbejde og det gør også at væsentlige aspekter af de kompetencer man opnår får man gennem projektarbejdet. I vort fag er projektarbejde og arbejdet med rapportskrivning ikke blot en undervisningsform men en central faglighed man skal beherske bagefter. På første semester møder mange studerende for første gang denne type projekt og der er derfor ofte en hvis usikkerhed om hvad der forventes og hvordan man griber projektarbejdet an. Denne note samler en række råd om projektarbejdet og projektrapporten. Den henvender sig primært til 1. semesterstuderende men skulle også gerne læses af vejlederen så vi har nogle rimeligt ensartede forventninger til projektet.

Noten har tre dele. Første del omhandler de krav og forventninger studieordning og studienævn har til projektarbejdet. Anden del drejer sig om råd til projektarbejdet og i tredje del gives nogle retningslinier for projektrapporter.

1 Formelle krav

Første semester projektet skal opfylde en semesterbinding "Softwareudvikling" med følgende ordlyd:

"Projektarbejdet skal [...] give de studerende erfaring i at planlægge, implementere, afprøve og dokumentere middelstore programmeringsopgaver ved anvendelse af et højere, generelt programmeringssprog."

(§8, stk. 2 og 3 i OB-studieordningen og §3 stk. 2 og 3 i TIT-studieordningen)

Semesterbindingen er den samme for TIT og Datalogi så det er helt uproblematisk at danne grupper på tværs af de to linier.

Projektet har et omfang af 1/4 semesters arbejde. Det svarer til ca 4 ugers fuldtidsarbejde til det egentlige projektarbejde. Hertil kommer så projektetablering og evaluering.

Projektet udarbejdes i grupper af normalt 2-5 studerende. Studienævnet kan godkende grupper på 6 studerende, men sådanne grupper kan af censor og eksaminator kræves delt ved eksaminationen. Eksaminationen er en gruppeprøve og tager udgangspunkt i den udarbejdede projektrapport. Denne mundtlige prøve varer normalt 20-30 minutter per studerende. Der gives ikke separat karakter for rapporten, men den indgår i eksamensgrundlaget gennem den mundtlige præstation.

Eksaminationen kan inddrage metoder og teknikker som de studerende har, eller med rimelighed kunne have benyttet i projektarbejdet. Hvad der på denne måde kan inddrages må afhænge af en konkret vurdering fra censor og eksaminator.

I projektarbejdet forventes de studerende at have et fagligt niveau med kendskab til softwareudvikling svarende til at de f.eks. har fulgt kurserne Indledende Programmering, Begreber og Redskaber i Programmering og Interaktive Systemer og Projektledelse. Der er ikke krav om at man specifikt har fulgt disse kurser idet andre kurser, projektarbejder, mm kan bidrage til et sådant fagligt niveau.

Fortolkning af krav

Omfang. Det væsentligste aspekt i projektarbejdet er at man selv skal udvikle et program af en sværhedsgrad der går noget ud over det helt elementære. Et løst skud vil sige et program på 300-500 linier afhængig af gruppestørrelse og kommenteringsstil. Det centrale i denne vurdering er at det kræver løsning af nogle reelle datalogiske problemstillinger - det skal altså ikke blot være 300 linier "println".

Der kan ikke gives en simpel anvisning på hvad der er en "reel datalogisk problemstilling", men det kan dreje sig om at det kræves overvejelser om hvordan data skal struktureres og repræsenteres eller om hvordan data skal behandles. Hvorvidt der indgår reelle overvejelser af denne art er noget man må diskutere med sin vejleder.

Det er ikke et absolut krav at programmet faktisk fungerer. At visse funktionaliteter ikke virker vil normalt ikke være et større problem. Er der derimod stadig syntaksfejl i programmet er det ret alvorligt. Det skal fremgå tydeligt af rapporten i hvilket omfang man mener programmet fungerer.

Ud over krav til at der faktisk foreligger et program skal man også "planlægge", "afprøve" og "dokumentere" det i rapporten. Det forventes derfor at projektrapporten bl.a. indeholder følgende dele:

- en analyse del med overvejelser med diskussion og argumentation for hvad der skal løses og hvordan det løses. Der kan også være overvejelser om planlægning af selve projektforløbet.
- en beskrivelse af programmets opbygning og virkemåde. Der skal som minimum ligge et velstruktureret program og ekstra information der er nødvendig for at forstå programmets opbygning og for at kunne bruge det i praksis.
- en afprøvning af programmet. Det skal som minimum indeholde overvejelser om hvordan man vil afprøve det og om muligt dokumentation fra eksempler på udførelser af programmet. Som afprøvningsstrategi kan f.eks. benyttes ekstern afprøvning af centrale funktionaliteter for programmet.

Rapporten vil typisk fylde 25-40 sider plus bilag med programtekst og evt andre udskrifter.

Niveau. De studerende forventes at kunne perspektivere og problematisere projektet i en større sammenhæng inden for modulets emnekreds. Det kan derfor være relevant at overveje og diskutere andre måder at løse programmeringsopgaven på og andre måder at dokumentere og afprøve programmet. Det forventes ikke at man ved "alt" om softwareudvikling, men efter cirka et semesters studium har man visse erfaringer at trække på.

Alle medlemmer af gruppen forventes at være fortrolig med alle dele af rapporten og projektarbejdet. Alle medlemmer må således kunne forklare programmets virkemåde. Det er så godt som umuligt at forklare et program man ikke selv har været med til at skrive hvis man ikke er en meget erfaren programmør.

Der stilles ikke krav om at der skal benyttes et bestemt programmeringssprog (java f.eks.), en særlig programmeringsstil (OOP, f.eks.), særlige analysemetoder (OOA/D,...), eller andre værktøjer (f.eks. referencelinier, UML). Det er naturligt i projektarbejdet at inddrage metoder og teknikker man har erfaring med fra kurser, men der er ikke krav om det.

2 Råd om projektplanlægningen

Gruppedannelse. Første fase af projektforløbet er at danne en gruppe. Vi anbefaler en gruppestørrelse på første semester på 3-4 studerende. Med flere studerende bliver den interne koordinering og arbejdsdeling noget mere kompliceret.

Det anbefales at man søger at danne en gruppe af studerende med nogenlunde samme forudsætninger. Er der en i gruppen som har væsentlig mere programmerings erfaring risikerer man let at den person løber afsted med programmet og resten bruger så uforholdsmæssigt meget tid på at prøve at forstå det denne person har programmeret.

Diskuter jeres ambitionsniveau tidligt. Regner I med at bruge 10 timer om ugen og er holdningen at alt over 6 er spildt arbejde? Eller vil man afsætte fuld tid (40 timer per uge) i den projektintensive periode.

Det kan være en ide at lave en egentlig aftale om hvordan I vil arbejde og hvad I vil forvente af hinanden.

Emnevalg. I god tid inden den projektintensive bør I foretage en afgrænsning af problemfelt. Det vil sige en klar afgrænsning af det felt man ønsker at arbejde med - hertil anbefales det at man inddrage vejlederen af projektet, som skal være med til at sikre at problemstillingen er en man kan nå at komme i dybden med på den afgrænsede tid, men samtidig også at der er nok kød på den til at projektgruppen kan komme rundt om de vigtigste aspekter

I den første fase er vejlederens hovedopgave at rådgive gruppen så den vælger et problemfelt der rejser nogle reelle datalogiske problemstillinger og som vil resultere i et program af en realistisk størrelse. Erfaring viser at det er nyttigt at udnytte vejlederens råd i denne afgrænsning så man undgår ubehagelige afgrænsninger senere i projektforløbet. Et godt projektemne til et 1. semesterprojekt kræver ikke de mange overvejelser om afgrænsninger senere hen i projektforløbet.

Man behøver ikke at have lagt sig fast på et emne ved vejlederansøgningen eller eksamenstilmeldning. Det kan dog være en god ide at diskutere forskellige emner med nogle vejledere ifm. projektbørs og tiden op til vejlederansøgningen. Få gerne kommentarer om rimelighed af emnet fra flere vejledere - de behøver ikke være enige.

Vejledende rapporteksempler. Man kan hente megen inspiration ved at læse andres rapporter. Man kan få ideer til andre emner, der kan omtales i en rapport enkelte afsnit, eller til en anden måde at gribe rapportarbejdet an på. Begrænsningen i sådanne eksempler er, at hvad der er godt i en rapport kan være indeligt ligegyldigt i en anden. Man kan finde de sidste mange års rapporter i bibliotekets kælder.

Planlægning af det intensive forløb. Prøv at estimer hvor lang tid i vil bruge på programmering og rapportskrivning. Det er svært og kommer sikkert ikke til at passe, men kan alligevel være nyttigt at man har en ide om hvormeget panik der nu er. Det kan f.eks. være en

referencelinieplan med Sørg for at afsætte tid til sidst til uforudsete ting. Programmering tager som regel noget længere tid end man forventer.

Det er naturligvis vigtigt at få skrevet et godt program. Man skal imidlertid også bruge tid på at skrive rapport og man skal nok passe på at forsøge på at rette nogle sidste dumme fejl eller uhensigtsmæssigheder ikke tager for meget tid fra afrunding af rapporten.

Vejledning. Vi lægger vægt på projektarbejde og vejledningsprocessen. På første semester kan man forvente at holde ca 6 vejledermøder af en halv time per studerende. Dette inkluderer møder i etableringsfasen og før en eksamen.

3 Råd om rapportudformingen

Kriteriet for en god rapport er egentlig ganske enkelt: Den skal være *anvendelig*. Begrundelsen for retskrivning, sidetal osv. er at det gør det lettere at læse rapporten. Tekstafsnittene skal indeholde de forskellige oplysninger og diskussioner fordi en mulig læser kan være interesseret i at læse om dem og gerne skulle kunne forstå jeres overvejelser.

Tænk på næste gang du sidder med en manual for en videobåndoptager, et program eller et programmeringssprog at det er dokumentation for et projektarbejde. Når du så forgæves bladrer rundt for at finde en eller anden oplysning skyldes det oftere gruppens manglende evne til at vide hvad en læser ønsker end din uvidenhed. For gruppens medlemmer er disse ting banale og oplagte, men for læseren betyder det at en barriere af indforståethed skal gennembrydes.

I den gode rapport har man tænkt på læseren. Den har et budskab til en læser, den har tænkt på at rapporten skal læses og ikke bare skrives. Det kan lyde banalt men er ikke spor nemt i praksis. Det er ikke nok at noget står et eller andet sted hvis det ikke er det naturlige sted for en læser.

Man behøver ikke være nogen ørn til at programmere for at skrive en god 1. semester rapport. Det hjælper derimod hvis man som forfatter engagerer sig i rapportskrivningen og faktisk har noget man gerne vil fortælle en læser. Nogle interessante budskaber gør det meget mere spændende at læse en rapport.

Hold læseren i hånden igennem rapporten. Lav korte indledninger til kapitler hvor det fremgår hvad der står og måske referencer fremefter til hvad man venter med til et senere kapitel. Afrund kapitler så læseren er klar over at nu har man altså sagt det man ville om emnet. Dette skal man nødig finde ud af ved at der er en kapitel overskrift eller et bilag på næste side. Sørg for at give korte introduktioner til afsnit så man ikke ryger direkte ned i detaljerne. Indsæt hjælpeoverskrifter (rubrikker) med jævne mellemrum i længere tekstafsnit så man ikke taber læseren. Mange læsere vil have glemt detaljer i emner omtalt mere end en halv side så undgå løse referencer til punkter omtalt længere fremme eller tilbage.

Læsergruppe. Når man skriver rapport skal man skrive rapport til *nogen*. For store dele af rapporten kan man have censor og eksaminator - eller måske den gode medstuderende (lidt over middel) som målgruppe. Erfaring siger at selv om man henvender sig til en målgruppe kan det i skrivefasen være en fordel at forestille sig en bestemt læser man skriver til. I rapporter på dette niveau er det vigtigt ikke at træde rundt i trivialiteter men forvente et vist niveau hos læseren uden at det bliver indforstået. En brugervejledning kan henvende sig til en fiktiv bruger.

3.1 Standarddisposition

Som udgangspunkt til rapportskrivning kan man benytte følgende standarddisposition.

- Forside
- Resume
- Indholdsfortegnelse
- Indledning.
- Problemanalyse.
- Beskrivelse af programmet.
- Brugervejledning.
- Afprøvning.
- Konklusion.
- Litteraturliste
- Bilag med bl.a. programudskrift.

På kapitelniveau kan det være en ide - men ikke noget krav - at bruge standarddispositionen. På afsnitsniveau skal det særlige ved projektet gerne komme frem. Her skal man tænke på om man får emnerne frem i en naturlig rækkefølge så man undgår for mange klodsede referencer fremad ("som vi vil diskutere senere..").

Når man disponerer og senere skriver indholdsfortegnelse bør man tænke i to niveauer med kapitler og afsnit. Flere niveauer giver nok ikke mening i så korte rapporter. Hvis man finder vigtige underafsnit man gerne ville have med i en indholdsfortegnelse var de måske så vigtige at det burde være på afsnitsniveau. Undgå mange korte kapitler da det gør læsningen for opbrudt. Man skal gerne nå en balance hvor læseren har overblik over hvor man er i rapporten og samtidig kan nå den passende fordybelse.

3.2 Forside og resumé

Forside skal indeholde titel, navne på forfattere og angivelse af i hvilken sammenhæng rapporten er skrevet (1. semester rapport på Dat eller TIT uddannelsen). Man bør nok også skrive hvem der har vejledt rapporten. Så slipper man for at det bliver fortolket som en kritik af vejlederen hvis man ikke gør det.

Et resume i starten skal give et overblik over rapportens indhold. Ca. 10 linier vil nok være passende for denne type projekt.

3.3 Indholdsfortegnelsen

Indholdsfortegnelsen har to formål. Man skal kunne finde det man søger og man skal kunne få et indtryk af hvad rapporten indeholder. Er den for detaljeret vil den forvirre mere end forklare. Indeholder den kun kapitler, kan man ikke se hvad rapporten indeholder.

3.4 Indledning

En indledning skal både kunne bruges til at finde ud af om rapportens emne og niveau var det man søgte, og den skal give et overblik over rapportens indhold. Leder man efter et bestemt emne blandt mange bøger, tidsskriftsartikler eller rapporter er det ofte indledningen og indholdsfortegnelsen man skimmer igennem for at danne sig et indtryk af om det var det man søgte.

Kunsten i at skrive en god indledning er at få det hele med uden at det bliver for langt. Man skal gerne få et overblik uden for megen læsning. Den følgende huskeliste indeholder punkter man nok skal overveje at medtage i en indledning.

- Hvem har skrevet rapporten og i hvilken sammenhæng.
- Introducer til emnet. Hvad drejer det sig om? Hvorfor skal man læse denne rapport? Sørg for at centrale stikord indgår.
- Centrale konklusioner og erfaringer. Formuler det så det kan forstås før man har læst rapporten.
- Status af arbejdet. Virker programmet? Redelighed i videnskabeligt arbejde er vigtigt. Hvis der er alvorlige fejl må det ikke skjules i en kort paragraf i afprøvningsafsnittet.
- Læserforudsætninger og læsergrupper. Er der enkelte kapitler eller bilag som henvender sig til andre grupper end den øvrige rapport bør det fremgå her.
- Oversigt over rapportens indhold. Hvad kommer hvor og i hvilken rækkefølge? Formålet er at give en vejledning til hvordan rapporten skal eller kan læses. Hvis indholdsfortegnelsen giver det fornødne overblik kan det udelades.
- Særlige typografiske, layout, mm bemærkninger. Er det noget man bør vide før man læser rapporten så bør det stå her.

3.5 Problemanalyse

Der er tre hovedformål med denne del. I større rapporter kan det være naturligt at opdele det i flere kapitler, men det må afhænge af en konkret vurdering. De tre mål er

- Introduktion til problemfeltet. Introducer de centrale begreber og notationer. Hvad er det for en verden programmet skal bruges i? Hvem er aktørerne/brugere og hvad laver de? Sæt scenen så man ikke gennem resten af rapporten skal forklare de centrale begreber man bruger.
- Afklar problemstillingen i projektet. Det kan sjældent koges ned til en 1-linies problemformulering men er snarere en diskussion der afgrænser anvendelser, funktionaliteter, brugssituationer mm. Denne del kan munde ud i en kravspecifikation. Overvejelser om hvad der skal løses og argumentation for foretagne afgørelser.
- Designovervejelser. Hvordan realiserer man ønskerne i et program? Hvilke centrale valg foretager man og hvad er de relevante alternativer? Overvejelser om hvilke metoder der skal bruges til at løse det.

En væsentlig udfordring i denne del er at få lavet en fornuftig disponering af materialet. Begreber skal introduceres før de bruges. Valg skal diskuteres før konklusionerne bruges i anden sammenhæng osv.

Introduktion. I denne del skal man gerne kort og klart beskrive hvad det er for et emne man arbejder med. Her introducerer man begreberne så man kan bruge dem frit siden hen.

I større rapporter kan introduktionen til problemfeltet inkludere en diskussion af andres arbejde med emnet. Er emnet behandlet i litteraturen bør dette refereres. I det omfang man har eller kunne/burde have baseret sig på andres arbejde bør det kommenteres og der skal være referencer til denne litteratur. For 1. semester rapporter vil sådanne diskussioner ofte ikke være relevante. Man kan måske nøjes med en enkelt reference til en java bog og så iøvrigt ikke bruge det som anledning til at diskutere hvad java er. Egentlige litteraturstudier vil normalt ikke være en del af et 1. semester projekt og der forventes heller ikke en teoretisk diskussion af begrebet "softwareudvikling". Blot fordi andre også har lavet et kryds-og-bolle spil på en computer behøver man ikke diskutere det i en første års rapport. Kommer inspirationen til projektemnet fra en artikel eller bog, så skal man dog angive det.

Hvis projektet er inspireret af emner fra et andet fag kan det være at en del af terminologien ikke kan forventes kendt af en almindelig læser, f.eks. eksaminator, censor eller en medstuderende. I sådanne situationer bør man nok bruge lidt plads på at introducere de begreber og emner man skal kende til for at kunne læse rapporten. Kræver projektet et dybere kendskab kan man være nødt til at forudsætte denne viden hos læsergruppen og angive dette i indledningen. Det er dog sjældent nødvendigt i en 1. semester rapport.

Afklaring af problemfelt. I analysen bør I overveje alternative løsninger og beslutninger til det I har valgt. Forsøg ikke at gøre valg I fortryder "logiske", men undersøg resultaterne i konklusionen. Det er fint nok hvis erfaringen fra arbejdet at man med fordel kunne have løst det noget anderledes. Når I vælger at bruge java er det ikke overbevisende med lange diskussioner om forskellige sprogs fortræffeligheder hvis læseren godt ved at det er de eneste sprog I kan programmere i. At noget er nemmest for jer er vel en glimrende begrundelse for et valg.

I analysen er det vigtigt særligt at koncentrere sig om de generelle, grundlæggende og helt fundamentale valg man foretager og ikke straks diskutere nogle efterfølgende meget specifikke valg. Når man skriver rapport kan man let glemme de centrale valg man foretog i starten af projektførløbet. Det er derfor en god ide at lave en huskeseddel gennem hele forløbet.

Det er vigtigt at afgrænse problemstillingen skarpt. Som hovedregel får man et bedre projekt ud af at løse et mindre problem godt end at løse et mere ambitiøst problem halvgodt.

Designovervejelser. I analysen bør man overveje anvendelsen af centrale algoritmer eller datastrukturer. F.eks. hvilke datastrukturer der kunne bruges, hvilke algoritmer der var relevante, hvilke klasser og objekter man har valgt at lave - samt hvorfor. Hvilken begrundelse ligger der bag f.eks. valget af datastrukturer? Overvej alternativer, for ofte vil alternativerne være mindst lige så interessante for en læser som jeres valgte løsninger.

Bliv ikke for programnær. Det er det centrale design man skal diskutere og ikke specifikke overvejelser om forskellige faciliteter i sproget. De tekniske muligheder og problemer man har med at få programmer til at virke er mange overvejelser værd når man endnu ikke er så erfaren programmør, men det er nok ikke så interessant at læse om.

3.6 Programbeskrivelse

Formålet med programbeskrivelsen er at give læseren den forståelse af programmets virkemåde I har haft under programmeringen. Bag sådanne programmer er der nogle ideer som ikke direkte

fremgår af programteksten og som er nødvendige at forstå for at kunne læse programmet. I nogle programmer er det centrale nogle datastrukturer hvor centrale informationer repæsenteres gennem programudførelsen. Andre gange er der en kodestump som styrer hele programudførelsen.

Er der dele af programmet som er særligt komplicerede bør man bruge lidt plads på at forklare de dele eller på anden måde støtte læseren i at forstå dem.

I det følgende gives en liste over nogle punkter man kan lade indgå i en programbeskrivelse. Listen skal dog tages med det forbehold at noget der er centralt for et program vil være inderligt ligegyldigt i en anden sammenhæng.

- oversigt over centrale underprogrammer
- Klassestruktur
- Centrale datastrukturer.
- System diagram. Oversigt over forskellige komponenter og deres forbindelse.
- Testudskrifter. Hvordan slår man dem til og hvad viser de.

Det der skal give overblikket står i hovedrapporten. Har man nogle flere oversigter som kan være nyttige for den ivrige programmør kan de placeres som bilag til rapporten. Man kan godt lave 'javadoc' oversigter over klasser men de er sjældent velegnede til at give et overblik over et programs struktur.

3.7 Brugervejledning

Det vanskellige ved at skrive en vejledning til brugere af et program at beskrive de ting som brugeren har brug for og beskrive det i den rækkefølge som brugeren vil forvente. Alt for ofte virker brugervejledningen ganske logisk — hvis man har forstået det hele, men den nye bruger bladrer forvirret rundt.

En brugervejledning kan i en mindre rapport blot være en kørselsvejledning - altså hvordan kaldes programmet og hvor og på hvilket format skal inddata være. Det kan være nødvendigt at forklare hvilke komponenter og anden software der skal bruges for at udføre programmet. Bliver denne del for specialiseret bør det nok placeres i et bilag.

Den lidt større brugervejledning kan indeholde

- Indledning med introduktion og beskrivelse af læserforudsætninger
- Kørselsvejledning. Hvordan startes programmet.
- Eksempel på udførelse. Det viser ofte mere end lange forklaringer.
- Fejlmeddelelser. Hvis brugeren ikke kan få programmet til at fungere kan der jo være nogle forklaringer på hvad man så skal gøre.
- Litteraturliste. Vejledningen skal måske kunne fungere uafhængig af rapporten.

3.8 Afprøvning

Vis udskrifter fra kørsler af programmet med udvalgte data. Som afprøvningsstrategi kan f.eks. benyttes ekstern afprøvning af centrale funktionaliteter for programmet. Man kan måske overveje andre afprøvningsformer som f.eks. intern afprøvning, bevisførelse, usability test med brugere. Sørg dog for at det ikke bliver så ambitiøst at der ikke bliver tid til at skrive et program.

Afprøvningsafsnittet skal gerne overbevise en læser om at programmet virker. Det sker nemmest ved at man viser man har arbejdet systematisk og ikke bare præsenterer nogle tilfældige kørsler. Hvis programmet ikke virker eller ikke virker helt så kan man stadig planlægge en afprøvning som det endelige program skal opfylde.

Det er en positiv egenskab ved en afprøvning hvis den har været i stand til at afsløre fejl i programmet, også selv om man ikke når at rette alle fejlene.

Yderligere beskrivelse af hvordan man kan dokumentere en afprøvning i en rapport findes i noten “om afprøvning” på kurset Begreber og Redskaber i Programmering.

3.9 Konklusion

Afsluttende kommentarer om projektarbejdet og opgaven. Stikord til denne del kan være:

- Hvor godt er det stillede problem løst?
- Hvilke erfaringer har implementeringen givet mht. til designet?
- Fremhævelse af eventuelle interessante delløsninger.
- Hvor vil det være oplagt at sætte ind med forbedringer og udvidelser?

3.10 Bilag

Brug bilag til det der ikke vil være en del af en normal gennemlæsning af rapporten men som man har glæde af til opslag eller en nærmere fordybelse i dele af rapporten.

Man vil normalt placere programteksten i bilag da det jo ikke er noget man staver sig igennem i en normal gennemlæsning. Programmet bør være velstruktureret og med fornuftige kommentarer der letter læsningen - både for læseren og jer selv. Kommentarer skal gerne gøre det nemmere at finde og forstå programdele - ikke skjule programteksten. Systematiske indrykninger er også med til at gøre et program overskueligt. Det helt afgørende for et læseligt program er dog at det er baseret på et fornuftigt og gennemtænkt design.

3.11 Typografi og “lay-out”

Dette afsnit indeholder en liste over en række mere formelle ting som bare skal være i orden. For meget sjusk giver et dårligt indtryk og irriterer læseren.

Skriv dansk. Hvis rapporten skrives på dansk, er det vigtigt ikke at forfalde til engelske betegnelser, når der findes gode danske oversættelser. Det kan lyde smart med en teknisk jargon, men det kan også skabe en distance til læseren og er ofte udtryk for dovenskab. Hvorfor sige at man compiler sit program når man kan sige at man oversætter det. Vil man bruge et engelsk ord, kan man enten skrive det i citationstegn, eller skrive det oversat i parentes. F.eks.

.. vi har brugt et “array”. ..
.. vi har brugt en tabel (eng. array). ..

eller

Adskil variabelnavne mm. fra den øvrige tekst. Brug citationstegn eller en anden skriftstype til variabelnavne mm. så det klart adskilles fra den øvrige tekst. F.eks. skal man ikke skrive

.. variabelen tegn indeholder nu (eller \n og .. men i stedet
.. variabelen tegn indeholder nu (“” eller et linieskift (“\n”) og

Tekst skal kunne læses fra nederste eller højre kant af papiret. Hvis en rapport kommer i et ringbind kræver det akrobatisk behandling at læse fra andre vinkler. Hvis rapporten er trykt to-sidet eller med to a5-tekstsider på et a4-ark skal man tænke over hvordan læseren skal læse rapporten uden at dreje den for meget rundt.

Paginer siderne og nummerer kapitler og afsnit. Nummerering af kapitler gør det nemmere at finde rundt, slå op og henviser. Det gør det også nemmere for læseren at have et overblik over hvor man er i rapporten.

Luftigt, men læseligt “lay-out”. Sørg for at teksten ikke præsenteres gnidret, men samtidig uden så meget overflødig luft at man mister sammenhængen i teksten.

Nyt kapitel på ny side. Det giver et luftigt “lay-out” og hjælper læseren med at orientere sig.

Mindst en overskrift per side. Inden for et afsnit er det tilladt at have hjælpende overskrifter der gør det lettere at overskue teksten. Sådanne overskrifter behøver man ikke medtage i indholdsfortegnelsen.

Husk korrekturlæsning. Den sidste korrekturlæsning er meget vigtig for indtrykket af rapporten. Ret stavfejl. Check om der er fejl i henvisninger (til afsnit, figurer, referencer i litteraturlisten) - det er irriterende for læseren. Husk også at kontrollere efter kopiering. Den sidste side kunne være glemt i en kopimaskine.

Sammenfatning

Kravene til rapportering i forbindelse med softwareudvikling er ret specielle. Der er mange detaljer som en læser kan være interesseret i - meget kan misforstås - meget kan være indforstået af en læser. En del af en rapport drejer sig om at formidle en kompliceret fælles forståelse og intuition om problemområdet og løsningsmodeller. Andre dele drejer sig om at formidle nogle komplicerede tekniske løsninger. Rigtig godt bliver en sådan rapport først hvis man gør sig klart at rapporteringen i sig selv er en krævende formidlingsopgave og ikke bare er noget dokumentation som også skal være der ved siden af en programtekst.

4 Kun til censorer og eksaminatorer

Projektarbejdet på 1. semester er en del af programmeringsundervisningen på vore uddannelser. Der afholdes kurser som understøtter det, men en central del af undervisningen er projektorienteret. Projektet på 1. semester skal derfor ses som en del af træningen i at programmere og ikke noget man laver når man har lært at programmere.

Formuleringen med at rapporten kun er udgangspunkt for eksamen men ikke det der bedømmes giver ofte anledning til forvirring. Der er ikke noget i vejen for at man som voteringsmetode bruger at man giver et bud på karakter for rapporten før eksamen og så f.eks. lader den mundtlige præstation kunne trække en karakter op eller ned. Dette er i orden så længe det kun er et voteringsmetode som man kan fravige i helt særlige tilfælde hvor f.eks. en eksaminand er usædvanligt blank i store dele af rapporten. Brugte man voteringsmetoden “altid” var bedømmelsen af rapporten og dermed af den samlede eksamen ikke længere individuel og dermed ulovlig jfr. eksamensbekendtgørelsen.

At en studerende er usædvanligt blank i store dele af rapporten er ikke i sig selv eksamenssnyd - også selv om rapporten er afleveret alene. Formelt set er den egentlige eksamen den mundtlige og snyd kunne der være at sende en substitut til eksaminationen. Hvis det kan bevises at en studerende har afleveret en rapport som en anden har skrevet kan det nok give anledning til en eller anden form for disciplinær sag, men jeg er ikke sikker på at det vil blive behandlet som eksamenssnyd.

Jfr eksamensbekendtgørelsen skal censor og eksaminator huske at tage notater om præstationerne. Det kan være en god ide med et groft overslag over hvor lang tid hver eksaminand får lov at komme med oplæg og blive udspurgt.

Diskussionen under eksamen skal have en “rimelig” relevans til projektet eller semesterbindingen. Hvad der er “rimeligt” er naturligvis en vurderingssag men som eksaminator og censor skal man være parat til at kunne forsvare det i en evt. klagesag. I sidste ende er det ankekommissioner der er dommeren i sådanne afgørelser og ikke studienævn eller studieleder. Detaljerede spørgsmål ud i pensum fra kurser de studerende har fulgt men som ikke er “rimeligt” relevante kunne give anledning til problematiske klager. På samme måde må man ved en kursuseksamen heller ikke spørge dybt ned i pensum fra et andet kursus blot fordi man ved at den studerende har fulgt det.

Som eksempler på hvad der med rimelighed kan inddrages er et kendskab til de centrale faciliteter i det programmeringssprog man har benyttet. En censor (Erik Meilling) kom i et brev i januar med nogle eksempler hvor man kan være i tvivl. Han nævnte et datalogisk træ, et array, filhåndtering og designmetoder. Jeg vil umiddelbart sige at de to midterste er relevante hvis man programmerer i java. Jeg vil nok sige ja til at man bør kende begrebet en træstruktur, men ikke forskellige særlige rød/sorte, balancerede, mm træer. For designmetoder kan man ikke forvente kendskab til OOA/D. Jeg er ikke lige sikker på om der er andre designmetoder man rimeligt kan forvente kendskab til. Situationen er nok i virkeligheden ikke meget anderledes end i den gamle studieordning hvor man kunne forvente et overblik over kursets pensum - men man kunne ikke forvente kendskab til specifikke dele af pensum.