Kunsten at vejlede et konstruktionsprojekt

Ken Friis Larsen, amanuensis, og Henning Niss, adjunkt, IT-Universitetet i København.

Ken Friis Larsen er uddannet civilingeniør fra Danmarks Tekniske Universitet og som ph.d. fra IT-Universitetet i København og DTU. Ph.d.-projektet omhandlede typesystemer for assembler-programmer til digital signal-processorer. Kens forskningsinteresser er inden for programmeringssprog generelt, samt hvorledes programmeringssprog-teknologi kan bringes til anvendelse. Var med til at starte IT-Universitetet i København. Siden 1995 ansat som amanuensis på IT-Universitetet og har her undervist, udviklet en række kurser og vejledt mange projekter og specialer på overbygningsniveau.

Henning Niss er uddannet datalog og siden ph.d. fra Københavns Universitet. Ph.d.-projektet omhandlede typesystemer til regionssystemer. Hennings forskningsinteresser er inden for programmeringssprog-teknologi og anvendelser af mobile og distribuerede systemer. Siden 2003 ansat som adjunkt på IT-Universitetet og har her undervist, udviklet en række kurser, vejledt projekter og specialer på overbygningsniveau samt vejledt en ph.d.-studerende.

I denne artikel beskriver vi nogle af de faldgruber ved konstruktionsprojekter, vi har observeret, studerende oftest falder i. Ved at beskrive ofte forekommende problemer er det vores håb, at både vejledere og studerende bedre kan blive opmærksomme på dem og imødekomme dem.

Muligheder og udfordringer i konstruktionsprojekter

Inden for datalogi- og IT-undervisning er det normalt at vejlede projekter, der indeholder en stor konstruktionsopgave. Det vil sige, at de studerende, alene eller i en gruppe, skal designe og implementere et større program. Konstruktionsprojekter giver gode læringsmæssige muligheder ved at give de studerende mulighed for at drage sig konkrete førstehåndserfaringer med programmering af (større) systemer. Der er dog nogle læringsmæssige udfordringer i forbindelse med konstruktionsprojekter. For det første er det vigtigt, at de studerende ikke kun fokuserer på at høste konkrete erfaringer, men også at de giver sig tid til at reflektere over deres erfaringer og er i stand til at videregive disse erfaringer og refleksioner (Kolb, David A. *Experiential learning – Experience as the source of learning and development*, 1984). Det er derfor sjældent alene programmet, de studerende bliver bedømt direkte på. I stedet er det en rapport, der beskriver programmet, der er hovedproduktionen og grundlaget for bedømmelsen. Dilemmaet for de studerende er, at konstruktionen af programmet såvel mentalt som tidsmæssigt udgør hovedparten af projektforløbet. Vores pointe er, at konstruktionsopgaver bør opfattes i den naturvidenskabelige tradition (se f.eks. Gauch, Hugh G. *Scientific Method in Practice*, 2002), hvori selve konstruktionen af programmet kan ses som eksperimentet. Konkret bør de studerende altså have et hovedspørgsmål, en *hypotese*, som de vil undersøge. For at foretage undersøgelsen skal de derfor analysere problemstillingen og på basis af dette opstille en række *eksperimenter*. De skal herefter udføre disse eksperimenter og indsamle data til at *evaluere* eksperimenterne, for til sidst at uddrage konklusioner af evalueringen. Som ved egentlige forsøg er det ikke mindst de forudgående opstillinger af hypoteser og analyser af problemstillingen og de efterfølgende evalueringer af eksperimentet, der er væsentlige at afrapportere. Dette er en idealiseret proces, og den giver ikke fuldt ud mening for alle projekter. Vi har valgt at strukturere artiklen som en række gode råd til de studerende – også selv om den primære målgruppe er andre undervisere. Mange af de råd, vi giver, vil virke oplagte for erfarne undervisere, men det er vores erfaring, at de fleste studerende har brug for vejledning i, hvordan de skal bruge den naturvidenskabelige metode til et konkret projekt, som de selv skal definere. De fleste studerende kender til den naturvidenskabelige metode fra både

folkeskolen og de gymnasiale uddannelser, men de er sjældent vant til at bruge den på projekter, hvor det ikke nødvendigvis er klart, hvad »eksperimentet« er. Derfor er vort ærinde at se konstruktionsprojekter i den naturvidenskabelige tradition snarere end at beskrive de forskellige (nødvendige) dele af den gode rapport, idet der i forvejen findes gode råd på området (f.eks. Sestoft, Peter. *Udformning af rapporter*, 2000, med henblik på datalogiske projekter, eller Rienecker, Lotte; Jørgensen, Peter Stray (2005): *Den gode opgave*, 2005, mere generelt).

Anvendelsen af den naturvidenskabelige metode består (forsimplet forklaret) i praksis i at:

0. Tage udgangspunkt i en bestemt forestilling eller model af verden (oftest ikke særlig eksplicit)
1. Formulere et spørgsmål eller en hypotese
2. Designe et eller flere eksperimenter til at illustrere spørgsmålet
3. Udføre eksperimentet og indsamle data
4. Analysere og evaluere data
5. Vurdere og konkludere på basis af evalueringen ovenfor og det samlede forløb.

Vores hovedpointe i denne artikel er, at et sådant forløb genfindes i konstruktionsprojekter – naturligvis ikke direkte – men i princippet. Punkterne ovenfor svarer til faser af konstruktionsprojektet, hvor den studerende:

0. Beskriver problemdomænet
1. Præciserer problemet og opstiller et hovedspørgsmål /hypotese
2. Planlægger og designer et program til besvarelse af hovedspørgsmålet
3. a. Rent faktisk implementerer programmet
   b. Indsamler data ved afprøvning af programmet
4. Evaluerer afprøvningen
5. Vurderer og konkluderer.

I det følgende vil vi behandle hver af disse faser. Det er interessant, at den egentlige konstruktion af programmet er hovedmålet i langt de fleste af de kurser, de studerende kan vælge at tage. De øvrige aspekter berøres ikke i nær samme grad.Vi vil gennem teksten betjene os af et opdigtet eksempel på et konstruktionsprojekt. Den nyopstartede virksomhed Coffee'R'Us, der leverer friskbrygget kaffe til ansatte i virksomheder, har bestemt sig for at gøre det muligt at bestille den friskbryggede kaffe via en webside. Coffee'R'Us' direktør, Georg Evalia, har gennem opslag fået kontakt med de studerende Mads og Lotte. Mads og Lottes opgave er at konstruere Coffee'R'Us' nye webside. Der er fra start stillet krav om, hvilken teknologi der skal bruges (i dette tilfælde programmeringssproget PHP og databasen MySQL).

Mål

Ud over de rent faglige mål har vi tre læringsmål for de studerende:

- At forstå og være opmærksomme på den naturvidenskabelige metode og tradition
- At være i stand til at udføre en saglig evaluering af deres eget arbejde
- At være i stand til at føre en videnskabelig diskussion.

Intentionen er naturligvis, at disse mål støtter tilegnelsen af den faglige viden. I konstruktionsprojekter af længere varighed, f.eks. specialer, vil analysen af problemstillingen typisk kræve, at de studerende sætter sig ind i nye teoriområder og træffer valg om, hvilke der passer bedst i den givne situation. Konstruktionen af programmet kræver dernæst, at de forstår at anvende den tilegnede viden på konkrete – og af dem selv opstillede – problemstillinger. Slutteligt leder indsamling og vurdering af data dem gennem en evaluering af denne viden og dens anvendelighed i den konkrete situation.

Konstruktion kontra rapportskrivning

Eksempel: Lotte og Mads elsker at programmere (og hader at skrive rapporter), så de giver sig straks til at udtænke og programmere en meget avanceret løsning til Coffee'R'Us. Der er hele tiden nye aspekter, de ønsker at inddrage i programmet, så de får aldrig rigtig lavet de enkelte dele af programmet helt færdig. Lige før rapporten skal afleveres, arbejder de helhjertet på at finde en fejl i den efterhånden meget avancerede programkode. I sidste ende når de aldrig at færdiggøre et program, der kan bruges til at besvare deres hovedspørgsmål, fordi de brugte al deres tid på programmering og på de avancerede detaljer.

Som nævnt er det de studerendes dilemma at balancere mellem at konstruere et tilstrækkeligt relevant program, kontra det at dokumentere programmet samt processen, der ledte til det endelige program.

Programmet behøver at være relevant for, at det kan lede til besvarelse af hovedspørgsmålet. Det skal f.eks. være færdigt nok til, at de studerende kan sige: »Ja, det var muligt at lave et program, der opfylder de indholdsmæssige mål, vi satte os«, »Ja, det var muligt at lave et program, der opfører sig korrekt« eller »Ja, det var muligt at lave et program, der har acceptabel ydelse sammenlignet med eksisterende programmer«.

Men det er ikke nok at konstruere programmet. For at vi som vejledere kan sige, at de studerende på videnskabelig vis har besvaret deres hovedspørgsmål, behøver vi mere end blot programmet. Vi bliver nødt til også at se, hvorfor der er belæg for at mene, at hovedspørgsmålet er besvaret. Med andre ord er det ikke nok at sige, at programmet indeholder den funktionalitet, de studerende var ude efter, det skal også

eftervises. På samme måde er det ikke nok at postulere, at programmet opfører sig korrekt, eller at programmet yder lige så godt som eksisterende programmer, det skal også afprøves og eftervises med målinger.

Opsummerende er vores budskab, at man ikke får så meget ud af at konstruere et godt program, hvis man ikke også vurderer, evaluerer og perspektiverer det.

Konkret udmønter dilemmaet sig i en række gode råd, vi giver vores studerende:

- Tænk rapportskrivningen ind i processen fra start
- Sørg for at dokumentere detaljer i programmet, der er ekstra smarte, gode, effektive, etc.
- Pas på med at rette den »sidste« fejl i programmet, hvis det sker på bekostning af rapporten. (Og i øvrigt er der altid en fejl mere at rette).

Konkretiser hovedspørgsmålet

Eksempel: Mads og Lotte begynder at programmere en webside med det samme. Undervejs beslutter de sig for, at det er vigtigt, at deres program tillader Coffee'R'Us løbende at ændre varesortimentet, og at løsningen til dette skal kunne betjenes af lægfolk. I realiteten har Coffee'R'Us ikke ændret i deres udbud i de seneste tyve år, og der er ingen planer om at gøre det fremover. De beslutter sig også for, at det er væsentligt, at deres løsning håndterer samtidige ordrer fra ikke mindre end 1 million kaffehungrende personer.

Oftest er opstillingen af hovedspørgsmålet eller hypotesen en del af formuleringen af de studerendes projekt og den tilhørende problemformulering. Den er måske endda udarbejdet i fællesskab med vejlederne. Det betyder, at de studerende som regel ikke behøver at starte med at opstille hovedspørgsmålet, men snarere skal starte med at konkretisere og afgrænse dette spørgsmål. En vigtig del af dette er at lægge sig fast på, hvad der er af succeskriterier for projektet. Er det nok, at der er noget, der illustrerer, hvordan kaffebestillingssiden kan se ud? Eller skal hele systemet udvikles og integreres med Coffee'R'Us' system til at levere kaffen?

Først når de studerende har fastsat de overordnede succeskriterier, giver det mening at konkretisere problemstillingen ved at opstille delproblemer, løsningsmodeller og alternative løsningsmodeller. Det er først i sidste ende, at det giver mening at diskutere tekniske aspekter af konkretisering. Med eksterne samarbejdspartnere bliver det ekstra vigtigt at afstemme forventningerne til ikke blot det konstruerede program, men også til form og indhold af afrapporteringen.

Konkretiseringen af problemstillingen er en forudsætning for at komme ordentligt i gang med at programmere. Ikke desto mindre ser vi ofte studerende, der mere eller mindre hovedløst sætter sig til at programmere, inden de har gjort sig klart, hvilket problem de præcis ønsker at løse, og ikke mindst hvilke krav løsningen skal opfylde. At opstille kravene er også en forudsætning for siden hen at kunne evaluere løsningen. Hvis man ikke på forhånd har gjort sig klart, hvad man ønsker at opnå, vil afprøvningen af løsningen ofte blive meget tilfældig. Dette giver anledning til følgende råd:

- Gør det fra starten klart, hvad succeskriterierne er, både i forhold til hvad programmet skal kunne, men også i forhold til om effektivitet er vigtigt (i givet fald i hvilken henseende)
- Vent med at programmere, til det er klart, hvad der skal programmeres
- Diskuter mulige løsningsmodeller, alternativer, og hvorledes der skal vælges mellem dem.

Fokuser på de rigtige problemer

Eksempel: Lotte og Mads bruger to uger på at sætte sig ind i et billedbehandlingsprogram, så de kan lave en dansende kaffebønne som logo til Coffee'R'Us' webside. Herefter bruger de en uge på manuelt at skrive hele Coffee'R'Us' varesortiment ind i deres testdatabase. Men de når aldrig til at forsøge at afdække, hvorfor det tager over tre minutter at bestille en kop kaffe, når der er modtaget mere end 32 ordrer.

Det er nemt at komme til at fokusere på uvæsentlige problemer i et konstruktionsprojekt, for det at løse uvæsentlige problemer minder i arbejdsmetode meget om at løse væsentlige problemer. I begge tilfælde er der tale om programmering. Ofte er det ikke lige så åbenlyst som i Mads og Lottes tilfælde, at man bruger tid på uvæsentlige detaljer, og det kan endda være mere tilfredsstillende at bruge tid på overfladepolering, da det kan få programmet til at *se* mere færdigt ud. Problemet er, at ikke alle problemer bidrager til undersøgelsen af hovedspørgsmålet. Det er en del af de studerendes opgave at demonstrere, at de mestrer at skelne de problemer, der afdækker hypotesen (de »væsentlige«), fra de, der ikke gør (de »uvæsentlige«). Det er ofte i udvælgelsen af de væsentlige problemer, at der kan opstå konflikter med eventuelle eksterne samarbejdspartnere, da det for lægmand kan være svært at forstå, hvornår noget er en opgave for en programmør, og hvornår noget er en opgave for en designer.

Råd: Vælg at fokusere på de rigtige problemer, spild ikke for meget tid på trivielle tidskrævende opgaver, men vælg opgaver med akademisk kød på.

Afprøvning af programmer

Eksempel: Mads og Lotte har hørt, at det er vigtigt at afprøve deres program, fordi det gør vejlederne glade. De bruger derfor deres løsning til at bestille en enkelt kop kaffe, og idet de fire timer senere får kaffen, konkluderer de, at deres program fungerer korrekt. De præsenterer denne afprøvning for vejlederne, der beklager sig over, at det er en meget mangelfuld afprøvning. Efter at have hørt om forskellige afprøvningsstrategier går de hjem og opstiller en mere tilbundsgående afprøvning af, at

programmet tillader at bestille de forskellige typer kaffe, Coffee'R'Us sælger, og at selve ordresystemet modtager de rigtige ordrer. På basis af dette konkluderer de, at deres program opfører sig 100 % korrekt.

Afprøvning af programmet sker for at dokumentere, at programmet opfører sig som forventet. For at kunne afgøre dette er det naturligvis nødvendigt at specificere, hvad »forventet« er. Det er ikke nok at sige, at programmet skal kunne gøre det muligt at bestille kaffe, man bliver nødt til at definere, hvad det vil sige. Afprøvningsfasen består dernæst i at identificere konkrete testeksempler, som indfanger det at kunne »bestille kaffe«, og dernæst i at afvikle disse testeksempler mod programmet for at konstatere, om programmet opfører sig som forventet.

Der findes forskellige strategier til at afprøve programmer, og de varierer m.h.t., hvad de afprøver, og hvor grundigt programmet undersøges. Det er ikke altid nødvendigt at teste alle detaljer i et program. Hvis Lotte og Mads f.eks. har sat sig for at demonstrere over for Coffee'R'Us, hvad en *mulig* webside til deres butik kunne være, er det ikke strengt nødvendigt, at programmet opfører sig 100 % korrekt. I det tilfælde kan de vælge at afprøve ud fra brugerens synspunkt og blot verificere, at programmet kan bruges til deres demonstration. Hvis de derimod var i gang med at konstruere kontrolprogrammet til en pacemaker, ville der være helt andre krav til afprøvningens grundighed. I sådanne tilfælde ville man formentlig opstille en formel specifikation af, hvilke krav programmet skal opfylde, og minutiøst eftervise, at det også er tilfældet for det udviklede program. Der er sjældent sådanne krav til de programmer, de studerende konstruerer, men der er ofte krav om mere tilbundsgående afprøvning end den først beskrevne. Om ikke andet så for at de studerende lærer, hvordan man afprøver programmer.

Det er vigtigt, at de studerende holder sig for øje, hvilke konklusioner de rent faktisk kan drage på basis af en gennemført afprøvning. Sjældent kan de sige, at »Programmet (i alle tilfælde) virker, som det skal«. I stedet vil konklusionen ofte være: »Vi har ikke været i stand til at afdække fejl i programmet. Vores teststrategi var 'brugerafprøvning', og vores testeksempler dækker kaffebestillingsdelen, men ikke sammenkoblingen med den egentlige kaffelevering.«

Afprøvningen kan naturligvis også bruges til at måle effektiviteten af det konstruerede program. Igen er det vigtigt at etablere kriterier for, hvorledes effektiviteten skal måles. Er programmet blot tåleligt i praksis, eller er det mere effektivt end et eksisterende program? Hvad vil »tåleligt« sige? Skal programmet i *alle* tilfælde være mere effektivt end det eksisterende program? Hvornår kan man sige det? I Mads og Lottes tilfælde burde de allerede ved deres indledende afprøvning, hvor det tog flere timer, før den bestilte kaffe ankom, have konstateret, at der var noget, der ikke var, som det skulle være.

Afprøvningen danner grundlag for at evaluere og perspektivere det udviklede program. Vi giver de studerende følgende råd:

- Gør det fra afprøvningens start klart, hvad den skal afdække, og vælg en strategi på basis af dette
- Konstruer afprøvningen, sådan at den giver mulighed for at evaluere programmet ud fra succeskriterierne
- Sørg for at afstemme konklusionerne af afprøvningen med den strategi, der er valgt, og undersøg, hvorvidt alle aspekter er afprøvet eller ej.

Vurdering af projektet

Eksempel: Til eksamen bliver Mads og Lotte spurgt, hvor færdigt deres program er. De forstår ikke spørgsmålet: »Hvad er det mon, vejlederne og censor fisker efter?« Har de glemt at implementere en vigtig feature, eller er det, fordi grafikken med Coffee'R'Us-logoet er gnidret? Efter eksamen snakker Lotte i telefon med Georg. I telefonen spørger Georg, om programmet er færdigt. »Ja«, svarer Lotte glad, »Vi bestod eksamen«. De aftaler derfor, at Mads og Lotte skal komme forbi Coffee'R'Us og give en præsentation af programmet på storskærm for hele firmaet. Der er ingen der siger noget, men Mads og Lotte kan mærke, at folk er lidt skuffede efter præsentationen. Hvad gik galt?

Efter at programmet er afprøvet på forskellig vis, er det passende at foretage en overordnet vurdering af programmet. Dels for at lave en samlet konklusion på alle afprøvningseksperimenterne og dels for at demonstrere, at den studerende er i stand til at give en realistisk vurdering af sit eget arbejde. Det vil sige: Hvad viser den nuværende konstruktion? Er problemet løst, eller viser konstruktionen, at der stadigvæk er nogle problemer, der skal håndteres? Er programmet færdigt med hensyn til at håndtere forretningslogikken, men mangler det stadigvæk at blive grafisk finpudset? Her er det specielt relevant at relatere løsningen til de opstillede succeskriterier (se afsnittet om konkretisering af hovedspørgsmål). Det kan også være relevant at inddrage kriterier, der ligger ud over de succeskriterier, der ellers er opsat for projekter. Hvor tæt er programmet f.eks. på at kunne sættes i produktion hos en eventuel ekstern samarbejdspartner?

Denne vurdering er vigtig for at give de studerende mulighed for at demonstrere, at de ligger »i toppen« af Blooms indlæringstaksonomi (Bloom, B. S. o.a.: *Taxonomy of Educational Objectives*, 1956), og at de er i stand til at perspektivere og opstille handleforskrifter.

Afrunding

Konstruktionsprojekter giver de studerende mulighed for at skaffe sig værdifulde førstehåndserfaringer med konstruktion af (større) programmer samt afprøvning af deres teoretiske viden. For at optimere tilegnelsen af disse erfaringer er det vigtigt at hjælpe de studerende igen-

nem de mere reflekterende dele af læringsprocessen.

Et konstruktionsprojekt kan med fordel opfattes i den naturvidenskabelige tradition, i og med at det skal undersøge et hovedspørgsmål ved et »eksperiment«. Hovedspørgsmål som »Kan det overhovedet lade sig gøre at konstruere et program til bestilling af kaffe via en webside?« og »Er det muligt at lave kaffebestillingsprogrammet effektivt nok til at det kan lade sig gøre at betjene 10.000 samtidige kunder?« kan kun besvares ved rent faktisk at konstruere et sådant program. På den måde er konstruktionsprojekter meget lig projekter i andre naturvidenskabelige discipliner. Vores pointe i denne artikel er, at der er en anden grund til at formulere projekterne på denne måde. Det hjælper nemlig de studerende (og vejledere!) til at se hvad der skal til for på videnskabelig vis at besvare det stillede spørgsmål.

Ovenfor har vi beskrevet vores erfaringer med de oftest forekommende faldgruber, og vi har givet råd til, hvordan man kan undgå dem. Ved at beskrive konkrete faldgruber og give konkrete råd, håber vi at studerende og vejledere kan få et værktøj til at undgå dem. Selv om vi har taget udgangspunkt i konstruktion af programmer i datalogisk projektarbejde, håber vi at erfaringerne kan give inspiration til lignende problemstillinger i andet projektarbejde af naturvidenskabeligt tilsnit.

Referencer

Bloom, B. S. o.a. (1956): *Taxonomy of Educational Objectives. The Classification of Educational Goals, Handbook 1: Cognitive Domain*, David McKay Company, Inc. (7. oplag 1972), New York.

Gauch, Hugh G. (2002): *Scientific Method in Practice*, Cambridge University Press. For en kort oversigt, se Wikipedia-artiklen om emnet (http://en.wikipedia.org/wiki/Scientific_Method).

Kolb, David A. (1984): *Experiential learning – Experience as the source of learning and development*. Prentice Hall P T R.

Rienecker, Lotte; Jørgensen, Peter Stray (2005): *Den gode opgave*, Forlaget Samfundslitteratur, Frederiksberg.

Sestoft, Peter. (2000): Udformning af rapporter, ikke-publiceret notat, IT-Universitetet i København. http://www.dina.dk/~sestoft/itu/rapport.pdf