Udformning af rapporter

Peter Sestoft, IT-højskolen i København
IT-C udgave 1, 2000-05-01

Denne note indeholder forslag til opbygning af rapporter over mindre programmeringsprojekter. Når man konstruerer et program til at løse en bestemt opgave, så skal man beskrive og begrunde programmets opbygning, forklare hvordan det bruges, og argumentere for at det løser opgaven. Den resulterende rapport er en meget væsentlig del af projektet, lige så væsentlig som et fungerende program.

1 Rapportens hovedafsnit

Forsiden skal angive forfatter, titel, dato, kursus/aktivitet, IT-højskolen i København. Herefter følger disse hovedafsnit:

- Forord og indledning
- Problemstilling og baggrund
- Problemanalyse
- Brugervejledning og eksempel
- Teknisk beskrivelse af programmet
- Afprøvning
- Konklusion
- Evt. litteraturliste
- Bilag: afprøvningsdata, programtekst, skærmbilleder og lignende

I et fireugersprojekt med 3 deltagere kan afsnittene være på f.eks. 1, 5, 5, 5, 4, 4, 1 side, i alt ca. 25 sider, hvortil kommer litteraturliste og bilag. Der er ikke nogen fast øvre grænse, men jo længere en rapport er, jo vanskeligere bliver det at overskue og forstå den.

1.1 Forord og indledning

Forordet angiver hvor, i hvilken periode og i hvilken sammenhæng arbejdet er lavet. Skriv det her hvis rapporten henvender sig til en speciel læserkreds, og sig tak til folk der har hjulpet. Resten af indledningen beskriver og motiverer den opgave der løses af det påtænkte program, uden detaljer.

Eksempel:

> Denne rapport er udarbejdet i maj 2000 i et fireugersprojekt på IT-højskolen i København under vejledning af .... Tak til NN for hjælp med programpakken til håndtering af TIFF-billeder.
>
> I projektet udviklede vi et Java-program til analyse af mikroskopibilleder af mælkesyrebakterier. Programmet analyserer en TIFF-fil produceret af et digitalkamera tilkoblet et mikroskop, finder konturerne af alle bakterier over en vis størrelse i billedet, og udskriver disse konturer til en fil i REG-format. Filer i REG-format kan derefter indlæses i standardprogrammet BactEdit fra Yoyodyne Corp, og man sparer altså den besværlige manuelle indtegning af bakteriekonturerne i BactEdit.

1.2 Baggrund og problemstilling

Her gives en omtale af den problemstilling (administrativ, biologisk, økonomisk, ...) som programmet behandler, og der opstilles krav til programmet. Der skal være baggrund og forklaring nok til at man kan forstå hvad det er programmet skal gøre, og hvordan det kan gøre det: formler der skal beregnes, osv. Her kan være henvisninger til relevant faglig litteratur.

I en større opgave kan dette afsnit opdeles i flere afsnit, f.eks. Baggrund (hvad er problemområdet?) og Problemformulering (hvad skal programmet kunne?).

Brugsscenarier ('use cases') kan benyttes til at indkredse kravene til programmet. Man gennemgår hvordan programmet benyttes af en tænkt bruger: hvilke muligheder skal der være (f.eks. ikke bare bestilling, men også afbestilling af billet), hvad kan gå galt (f.eks. forsøg på at bestille en billet når der ingen ledige pladser er), hvordan skal programmet reagere på sådanne fejl, osv.

Overordnet systemdesign hører normalt også til i dette afsnit. Det kan f.eks. være beslutningen om man skal have et server-tungt system (al databehandling foretages på en central server og brugeren ser kun en HTML-baseret grænseflade) eller et klient-tungt system (kun data forvaltes centralt i en database, mens al anden databehandling foretages ude hos brugerne).

1.3 Problemanalyse

Det foregående afsnit har fastlagt hvad programmet skal kunne. I dette afsnit diskuteres hvordan et program kan laves så det løser opgaven. Problemanalysen kan bestå af en beskrivelse og diskussion af alternative løsninger, f.eks. forskellige måder at repræsentere programmets data, og forskellige måder at beregne ønskede resultater.

De mulige løsninger præsenteres på et begrebsmæssigt plan med den terminologi og notation der er relevant for problemstillingen, og med gængs terminologi for algoritmer og datastrukturer. Man kan benytte mindre stykker egentlig programtekst i præsentationen når særlig præcision behøves.

Det er i orden at beskrive både en simpel men langsom algoritme og en indviklet men hurtig algoritme, og derefter beslutte sig for at programmere den simple af tidsårsager.

Hvis et program benytter en database, så kan man i problemanalysen diskutere databasedesign: hvilke tabeller skal man have, hvilke nøgler, og hvilke relationer mellem tabellerne.

1.4 Brugervejledning og eksempel

Her forklarer man hvordan programmet anvendes, uden henvisning til dets indre opbygning.

Hvis programmet er vinduesorienteret bør man have billeder eller tegninger af brugergrænsefladen, og en beskrivelse af brugen af de forskellige komponenter (knapper, menuer, vinduer).

Hvis programmet læser og skriver tekstfiler, bør man beskrive format og betydning af inddata, eventuelle programparametre, og format og betydning af uddata.

Hvis der er begrænsninger i programmet, eller særlige krav til inddata, skal de anføres i brugervejledningen. F.eks. 'Programmet kan kun finde ydre konturer, og virker derfor ikke korrekt på hule bakterier (men de forekommer heller ikke i virkeligheden)'.

Forklar hvilke fejlmeddelelser programmet kan give, og hvad de betyder.

Giv et komplet eksempel på en programudførelse, detaljeret nok til at læseren kan køre eksemplet selv (det er en god måde at lære et nyt program at kende). Eksemplet kan f.eks. vise (del af) inddata, forklare hvordan eksemplet køres, og vise (del af) de resulterende uddata.

1.5 Teknisk beskrivelse af programmet

Her beskrives de vigtigste moduler (klasser), datastrukturer (objekter, tabeller), og algoritmer (metoder). Beskrivelsen kan inddeles i underafsnit:

- Interne datastrukturer: hvordan repræsenteres data i programmet
- Interne algoritmer: hvordan behandles data i programmet
- Brugergrænsefladen:
  - for vinduesorienterede programmer: hvilket lay-out, hvilke komponenter, og hvilke lyttere består brugergrænsefladen af;
  - for programmer der arbejder på tekstfiler: hvordan læses og skrives inddata og uddata.

Beskriv det overordnede formål med hver klasse. Hvis en klasse bruges til at danne objekter med, så beskriv også de væsentligste (offentlige) metoder i klassen. Det er normalt ikke hensigtsmæssigt slavisk at beskrive hver klasse i alle detaljer. Den tekniske beskrivelse af programmet skal ikke gentage det som man uden videre kan læse i programmet, men skal være en støtte til at forstå det.

1.6 Afprøvning

Her godtgøres at programmet løser den stillede opgave. Det kan gøres med en brugerafprøvning, eller med en systematisk (intern eller) ekstern afprøvning. Man kan nøjes med systematisk afprøvning af særlig interessante dele af programmet. Eksemplet i brugervejledningen kan eventuelt fungere som brugerafprøvning.

I alle tilfælde beskrives givne inddata, forventede uddata, faktiske uddata og observerede uoverensstemmelser. Afsnittet skal indeholde en konklusion med en vurdering af afprøvningens resultater og dens grundighed: hvor stor er risikoen for at der stadig er (ikke påviste) fejl i programmet?

1.7 Konklusion

Konklusionen skal resumere arbejdet, og vurdere hvor godt det er lykkedes. Virker programmet? Er det praktisk i brug? Er det hurtigt nok? Er der noget der burde have været lavet anderledes? Er der nogle forbedringer eller udvidelser der burde laves i en evt. fremtidig udgave af programmet?

1.8 Bilag

Programteksten skal vedlægges som bilag. Den skal udskrives i et skriftsnit med fast bredde, f.eks. Courier, ikke Times. Sørg for at programteksten har et fornuftigt lay-out (indrykning) så den er læselig.

2 Rapportens form

Husk på at rapporten er et formidlingsredskab, ikke et litterært eksperiment, og ikke en tilfældig samling beskrevne sider.

- Skriv målrettet: for at meddele noget, ikke for at gøre rapporten længere.
- Skriv til en målgruppe: hav en læser i tankerne. De fleste afsnit henvender sig til den person der har bestilt programmet, men brugervejledningen henvender sig til personer der ønsker at bruge programmet, og den tekniske beskrivelse henvender sig til personer der ønsker at ændre (udvide, rette) programmet.
- Skriv enkelt: undgå fyldord; undgå tunge pseudo-videnskabelige formuleringer; undgå tågede formuleringer (prætentiøst og uklart: ‘systemets arkitektur kan karakteriseres som værende opbygget efter et distribueret client-server paradigme med multiple klienter’; ligefremt og oplysende: ‘hver bruger kører en applet som kommunikerer med en fælles database’); undgå variation for variationens skyld; beskriv beslægtede ideer med beslægtede formuleringer.
- Skriv korrekt: slang, stavefejl, sprogfejl, uforklarede forkortelser, uforklarede begreber og mærkelig tegnsætning forstyrrer læseren og hæmmer formidlingen.
- Lav en indholdsfortegnelse. Nummerér afsnit og underafsnit hvis det øger overskueligheden.