Udformning af rapporten (GRPRRO)

Dette dokument tager afsæt i Peter Sestofts *"Udformning af rapporter" (2000-05-01)* med henblik på at gøre det lettere at skrive rapporterne til projektet i grundlæggende programmering. Således udleveres det originale notat ikke, men derimod fremhæver vi nogle af de (mange) gode pointer, som Peter kommer med. Vi anbefaler man læser det oprindelige notat senere, da I vil støde på at skulle skrive sådanne rapporter under jeres uddannelse.

0 Indholdsfortegnelse

1 Rapportens struktur

Her beskrives strukturen for projektrapporten samt projektdagbogen. De enkelte afsnit beskrives i den efterfølgende sektion:

Projektrapporten
Forside; Introduktion; Design; Testning; Konklusion.

Projektdagbogen
Tema 1 (Resumé, Design, Testning); Tema 2 (Resumé, Design, Testning); Tema 3 (Resumé, Design, Testning); Tema 4 (Resumé, Design, Testning).

Commented [MOU1]: (Systematisk) Afprøvning?

Commented [MOU2R1]: Flere steder…

2 Projektrapporten

Indholdet i projektrapporten opsummerer og diskuterer, på overordnet plan, projektet samt systemet. Her gennemgås afsnittene enkeltvist. For alle afsnit gives et estimat på hvor langt hvert afsnit skal være. Dog kan det variere om de er lidt kortere eller længere, afhængigt af hvor meget man har at sige. Det er dog vigtigt at man altid forsøger at fatte sig koncist (kort og præcist), og derfor anbefaler vi at sidetalsforslagene er et maksimum, da der er et maksimum sideantal for rapporten.

Introduktionen (1 side)

Dette afsnit skal indeholde en præcisering og afgrænsning af hvad jeres system understøtter ud fra de krav som er stillet. Her er det også vigtigt at kontekstualisere jeres løsning, altså hvad skal programmet overordnet (trods at dette står beskrevet i projektbeskrivelsen).

Design (4 sider)

Design sektionen skal beskrive jeres overordnede design (eventuelt med brug af diagrammer) og begrunde de største, og væsentligste, designbeslutninger i løsningsdomænet vedrørende systemet (klasser, interfaces, mv. samt relationer mellem disse). Hvad der er de væsentligste beslutninger, afhænger af hvad man er løbet ind i undervejs som gruppe, dog skal hele systemet kort beskrives – fra indlæsning af data til simuleringen er færdig.

Man kan her forestille sig, at en læser, ud fra denne sektion, skal forstå hvordan systemet hænger sammen, og kan forstå (de vigtigste) valg der er blevet foretaget.

Testning (1 side)

Testningsafsnittet skal indeholde en kort diskussion af hvorvidt systemet opfylder de opstillede krav, og i hvilken grad. Her er det vigtigt at argumentere for hvorfor man mener dette, hvilket bør understøttes af systematisk testning (hvis man faktisk *har* gjort dette). Sektionen bør også beskrive den testningsstrategi, altså hvordan der er blevet testet. Her er det vigtigt at man beskriver hvad man faktisk *har* gjort, og ikke hvad man *ville* have gjort.

Konklusion (1/2 side)

Konklusionen bør indeholde en kort gengivelse af de vigtigste elementer fra de tre ovenstående afsnit.

3 Projektdagbogen

For hvert tema skal projektdagbogen indeholde de sektioner som beskrives i dette afsnit. Der vil således være fire af de følgende afsnit i projektdagbogen når projektet afsluttes. Det er her vigtigt at disse afsnit skrives samtidigt med udviklingen, eller ved afslutningen af ugen (temaet). Således kan det også bruges indledningsvist til jeres vejledninger. Disse afsnit skal repræsentere de overvejelser gruppen har undervejs. De kan tænkes på som værende en form for ”dagbog”. Fra uge til uge vil det variere hvad der fylder mest, men samlet set bør de tre afsnit ikke fylde mere end 2 sider. Nedenunder beskrives de forskellige afsnit.

Resumé

I stil med afsnittet Introduktion, skal der her redegøres for den funktionalitet der er blevet arbejdet på undervejs i den specifikke uge (temaet). Hvis der er taget nogle væsentlige

beslutninger angående bestemte krav som I mener *skal* forklares, kan det her kort nævnes. I kan her også nævne de vigtigste opdagelser fra jeres objekt-orienterede analyser.

Design

Design sektionen bør indeholde en beskrivelse og argumentation for de *vigtigste* designmæssige beslutninger, I har foretaget og implementeret den sidste uge. Dette kan både være nye tilføjelser til systemet (som eventuelt også påvirker tidligere dele af programmet) eller modificeringer af dele af programmet fra tidligere. Denne sektion kan med fordel skrives *når* I tager beslutninger. **Det er her vigtigt at I ikke går ned og beskriver enkelte metoder i detaljen**. Det handler om at præsentere *designmæssige* overvejelser, ikke hvordan I har skrevet jeres `if`-statements. **I skal derimod skrive (væsentlige) implementations væsentlige beslutninger ind som Java dokumentation.** I må her også gerne anvende diagrammer.

Testning

Testning sektionen her, bør indeholde en beskrivelse af hvordan I har tilgået testning af jeres system denne uge, og i hvilket omfang I har testet. Her kan I også beskrive opdagelser af fejl I gjorde jer ved specifikke test-cases. I behøver her ikke beskrive selve strategien hvis denne har været den samme hele vejen igennem og er beskrevet tidligere i projektdagbogen. Dog skal I eksplicitere at denne er det samme, hvis dette er tilfældet.

4 Rapportens form

Det er vigtigt at huske at rapporten er et formidlingsredskab, ikke et en tilfældig samling beskrevne sider. Formålet er således at sørge for at rapporten er sammenhængende og fører læseren gennem en sammenhængende fortælling.

Generelle råd til hvordan du skriver

Her følger en serie af råd til hvordan du skriver, baseret på Peter Sestofts *"Udformning af rapporter"*:

- **R1)** **Skriv målrettet: Du ønsker her at meddele noget, ikke at gøre rapporten længere**. Det at få enkelte pointer eller argumenter ned på én eller få sætninger er utroligt svært, men styrker budskabet.
- **R2)** **Skriv til en målgruppe: hav en læser i tankerne.** Målgruppen for projektets rapport er individer med samme tekniske forståelse som dig selv. Specifikt kan du forestille dig en person som kun har adgang til de dokumenter og filer som er udleveret i forlængelse med projektet.
- **R3)** **Skriv enkelt: undgå fyldord; undgå tunge formuleringer; undgå tågede formuleringer.** Undgå variation for variationens skyld; genbrug de samme termer for samme ideer således at læseren let kan genkende ting på tværs af din rapport.
- **R4)** **Skriv korrekt**: uforklarede forkortelser, slang, stavefejl, sprogfejl, begreber og mærkelig tegnsætning forstyrrer læseren og hæmmer formidlingen.
- **R5)** **Lav en indholdsfortegnelse**. Anvend afsnit og underafsnit til at strukturere din rapport, hvis det øger overskueligheden.

Om argumentation og diskussion

At argumentere for et design valg, for herefter at diskutere det, er typisk svært for softwareudviklingsstuderende. Den specifikke form vi søger i projektrapporten følger *de generelle råd til hvordan du skriver*. Specifikt er det værd at fremhæve **R1)** og **R3)**. Målet er *tydeligt* at formidle de pointer vi har. Det eksempel som gives nedenunder, itererer over et stykke tekst, således at det bedre forklarer skribentens pointe. Med det i mente, er det ikke nødvendigvis denne type refleksion der er vigtigst for dit projekt (og det er heller ikke sikkert at selve pointen er rigtig!). Derfor bør du i større grad bruge dette afsnit til at hjælpe dig med at udfolde dine pointer (altså formen) og ikke så meget indholdet. Eksempelvis kunne man se følgende i en rapport:

*Vi valgte at designe vores klasser således at alle dyr nedarver fra én abstrakt klasse **dyr**. Vi kunne her også have lavet flere abstrakte klasser, men valgte én abstrakt klasse fordi det virkede bedre i systemet*.

I dette eksempel er der flere ting galt. Først og fremmest forklarer vi aldrig rigtig *hvorfor* vi tog den beslutning, vi mangler altså at argumentere for valget. Dertil er det svært at se værdien i netop dette skriv – hvad er budskabet (vi følger altså ikke **R1)**)? En forbedring kunne derfor se således ud:

*Vi valgte at designe vores klasser således at alle dyr nedarver fra én abstrakt klasse **dyr** for at reducere kodeduplikation, da alle dyr har fællestræk; de kan være i live, dø, skal spise, og sove. Alternativt, kunne vi have anvendt flere abstrakte klasser, men vi valgte én abstrakt klasse fordi det virkede bedre i systemet*.

Vi er nu tættere på noget som ville fungere godt I en rapport – vi argumenterer nu for vores valg med afsæt i vores løsning. Dog diskuterer vi stadigt ikke, hvorfor vi nævner alternativet *flere abstrakte klasser* hvis vi ikke vælger det? Hvis det ikke er relevant, skal det ikke med (ifølge pkt 1.). Dog kunne det være, at skribenten i virkeligheden prøver, at forklare, specifikt hvorfor de ikke valgte dette:

*Vi valgte at designe vores klasser således at alle dyr nedarver fra én abstrakt klasse **dyr** for at reducere kodeduplikation, da alle dyr har fællestræk; de kan være i live, dø, skal spise, og sove. Dog varierer de enkelte subklassers implementation af disse nedarvede metoder, og derfor overvejede vi her, at have flere abstrakte subklasser, således, at hver subklasse ikke selv skulle implementere variationen. Vi fravalgte dette, da der ikke var nok delt funktionalitet mellem typerne af dyr, og det derfor havde ført til unødigt mange abstrakte klasser.*

Dette stykke tekst argumenterer langt bedre for valget og diskuterer alternativer skribenten havde overvejet. Da rapporten blandt andet handler om jeres refleksioner, er det vigtigt at du her også diskuterer de overvejelser I har haft, og hvorfor en given beslutning blev taget. Dog skal du ikke bruge tid på irrelevante eller mindre designovervejelser – du har ikke så meget plads!