# Projektforløbet: eksamens- og delafleveringer

## Overordnet om projektforløbet

Kursets undervisning og eksamen er centreret omkring jeres arbejde i grupper på et softwareudviklingsprojekt. Projektet går ud på at designe og konstruere en applikation, som opfylder de krav der er givet i det valgte projektoplæg.

De forskellige projektoplæg kan tolkes som de er, eller I kan bruge instruktorerne som repræsentant for kunden, til at afklare funktionelle krav til applikationen; i rapporten skal I redegøre for hvordan I har tolket projektoplægget, foretaget afgrænsninger, og brugt instruktor som repræsentant for kunden.

### Rapport og programkode til eksamen

Ved kursets afslutning skal I som gruppe aflevere; 1) en rapport der beskriver jeres projekt, og 2) programkode for jeres applikation via GitHub. Rapporten og programkoden danner grundlag for, og indgår i bedømmelsen af, den individuelle mundtlige eksamen. Kravene til rapporten, som er beskrevet nedenfor, er fastlagt ud fra at vi skal kunne vurdere hvad I har lært på kurset. Rapporten og de beskrevne aktiviteter afspejler derfor ikke nødvendigvis en praksis og form for dokumentation man vil finde i et typisk udviklingsprojekt.

### Delafleveringer

Delafleveringerne er **ikke obligatoriske, men stærkt anbefalede** af flere grunde; 1) I får løbende arbejdet på både jeres rapport og applikation, 2) I får konstruktriv feedback på jeres arbejde, og 3) I får arbejdet med teorier og processer præsenteret på kurset.

Jeres løbende projektarbejde struktureres som fire delafleveringer. For hver delaflevering udleverer vi en beskrivelse af hvad I forventes at lave i løbet af de følgende fire uger indtil afleveringsfristen, samt krav til en delrapport I skal aflevere. Delrapporten afleveres sammen med programkoden som en release af applikationen i GitHub.

Hver delaflevering skal beskrive arbejdet med planlægning af en eller flere iterationer i jeres udviklingsarbejde (jf. Extreme Programming-metoden som beskrevet i [Agile]). Det er dog op til jer hvordan I vil organisere jeres udviklingsarbejde, f.eks. i kortere iterationer på en eller to uger ad gangen, så længe I forklarer dette i jeres rapport og at I beskriver jeres planlægning af iterationer.

Hver delaflevering fokuserer på forskellige aktiviteter i softwareudvikling. Hver delrapport vil indeholde elementer der ligner de elementer der skal indgå i den endelige rapport. I kan vælge at inkludere dele af delrapporterne direkte

i eksamensrapporten, men det er meningen at jeres arbejde bliver løbende forbedret gennem kurset.

I vil få feedback på hver delaflevering, dels fra jeres instruktor, dels via gensidig evaluering med andre studerende. Bemærk at feedback fra jeres instruktor er formativ og ikke reflekterer bedømmelsen af den endelige rapport ved eksamen.

#### Projektets bedømmelse i forhold til kursusmålene

Uanset hvilket projekt I har valgt at arbejde med, vil vi i bedømmelsen af rapporten og programkoden finde det væsentligt:

- at I har afleveret en kørende og såvidt muligt funktionelt afrundet version af jeres applikation, som demonstrerer at I har forstået at planlægge jeres udviklingsarbejde.
- at valg af funktionalitet i den afleverede version er velbegrundet — det er derimod ikke væsentligt at I afleverer en "endelig" version af applikationen som implementerer al den funktionalitet I har afklaret som nødvendig.
- at jeres program er let at vedligeholde og tilpasse (f.eks. til nye eller ændrede krav).

Såvel ved eksamen som i jeres rapport, lægges der vægt på at I kan forklare og begrunde jeres valg i design, programmering og afprøvning i forhold til *visse* kvalitetskriterier for software. Rapporten og programkoden skal demonstrere at I kan udvikle et program:

- som er testet for om det opfylder definerede behov/krav (dvs. består accepttest) — *functionality*,
- som er pålideligt og robust overfor forkerte input, eksterne problemer såsom netværksfejl, mv. (dvs. testet for fejl og grænsetilfælde) — *reliability*,
- som er let at vedligeholde (dvs. let at finde og rette fejl) — *maintainability*,
- som er let at tilpasse til ændringer (f.eks. til platform eller omgivelser) samt at tilføje ny funktionalitet — *portability*.

Der lægges for dette kursus **mindre vægt** på andre kvalitetskriterier, som at jeres program:

- er effektivt i forhold til hvor ressourceforbrug (tid, lager, netværk) — *efficiency*,
- er let at lære og anvende — *usability*.

## Fælleskrav til delafleveringerne

Fælles for hver delaflevering er, at den skal bestå af **a)** en delrapport i PDF-format og **b)** en version af applikationen.

**Ad a.** I skal bruge gruppefunktionen på Absalon til at angive gruppens medlemmer. Derudover skal rapporten opfylde de samme formelle krav, bortset fra antal sider, som den endelige rapport beskrevet nedenfor.

**Ad b.** En version af applikationen — en release i GitHub — skal oprettes i GitHub med navnet "Deliverable-1", og mærkes som "pre-release".

## Krav til eksamensrapporten

Rapporten skal bestå af følgende dele:

### 1. Problembeskrivelse

En kort beskrivelse af jeres applikation og dens anvendelse. Væsentlige tolkninger af opgaveteksten, hvor I fandt den tvetydig, og afgrænsninger I har valgt at foretage skal beskrives og begrundes.

### 2. Krav

En beskrivelse af kravene til jeres applikation i form af brugsscenarier (*use cases*). Mindst to brugsscenarier skal skrives på fuld form (jf. [Cockburn]), resten skal kun skrives på kort form.

Jeres brugsscenarier skal omfatte den funktionalitet, som I havde planlagt for den endelige version, og kan derfor omfatte brugsscenarier som I endnu ikke har implementeret. Det skal fremgå klart hvilke brugsscenarier jeres endelige version implementerer. I skal give et rationale for valg af funktionalitet i den endelige version.

Redegør for hvilke brugsscenarier og underordnede opgaver I har arbejdet på i hver delaflevering. Dette skal inkluderes i en tabel i bilag A.

### 3. Design

Beskriv applikationens systemdesign, der tydeliggør sammenhængen med platformen og de programmeringsbiblioteker som applikationen er baseret på.

Beskriv jeres overordnede programdesign. Inkludér et klassediagram som giver et overblik over hvordan jeres applikation er designet (jf. *roadmap* i [Agile]).

Beskriv gennemgående abstraktioner og designmønstre som I har brugt i jeres design. Brug diagrammer hvor det er relevant. Eksemplificér ved at gennemgå mindst ét brugsscenarie.

Jeres design skal begrundes. I skal argumentere for hvordan jeres design opfylder ikke-funktionelle krav, herunder at jeres programdesign er modulært og løst koblet således at det er let at udvide og vedligeholde.

Redegør for de væsentligste designmæssige ændringer I har lavet i hver delaflevering; dette skal tilføjes tabellen i bilag A.

### 4. Afprøvning

Beskriv hvordan I gennem systematisk afprøvning har sikret; 1) at applikationen opfylder de beskrevne krav og 2) at den er pålidelig og robust overfor undtagelser og grænsetilfælde.

Sammenfat resultater fra accepttests og unittests for den afleverede udgave af applikationen; de fulde rapporter skal vedlægges i bilag eller være tilgængelige online (link skal tydeligt angives).

Redegør for (evt. kvantitativt) i hvilket omfang programkoden er dækket af unittests.

Endelig skal I forklare hvordan jeres afprøvningsmetode støtter udvikling og vedligeholdelse af applikationen, herunder i hvilket omfang jeres afprøvning er automatiseret og hvilken betydning det har for processen.

I skal have implementeret mindst ét brugsscenarie ved at følge en *test-first* tilgang ved iterativt at skrive først tests (der fejler), dernæst programkode (så tests virker); forklar i hvilket omfang I har fulgt denne tilgang og diskutér jeres oplevelse, herunder fordele og ulemper.

### 5. Udviklingsmiljø

Beskriv jeres udviklingsmiljø, herunder anvendte programmeringsmiljø, opsætning af projektet i GitHub, plan for versionsstyring, mv. Forklar hvordan filer i projektet er struktureret. Forklar endvidere hvordan jeres applikation bygges og afvikles.

Dette afsnit skal kunne læses af nye udviklere som vejledning i at sætte et udviklingsmiljø op og påbegynde videreudvikling på projektet.

I skal endvidere diskutere, hvordan jeres udviklingsmiljø og de værktøjer I har brugt, effektivt understøtter arbejde med programmering, refaktorering, afprøvning og afvikling af applikationen.

### 6. Diskussion af projektarbejdet

Diskutér udviklingsprocessen: erfaringer med samarbejde, planlægning, parprogrammering, etc.

Forklar hvordan I har skrevet tests, om I har gjort det løbende (evt. *test-first*), og hvilken betydning det har haft for hvordan I har programmeret og for det resulterende programdesign.

Redegør for hvordan de forskellige dele af udviklingsarbejdet var distribueret mellem gruppens medlemmer. Har gruppemedlemmer arbejdet på hver deres del af programmet eller på alle dele? Har alle gruppens medlemmer ligeligt bidraget til at skrive programkode, testkode, rapporttekst, mv.? Diskutér hvilken indflydelse dette har på programkoden, test, mv. samt på den videreudvikling.

Diskutér de væsentligste udfordringer I havde i projektet i forhold til beskrivelse af krav, planlægning af iterationer, design, programmering, afprøvning, udviklingsmiljø, mv.

#### Bilag A: Oversigt over delafleveringer

Dette bilag skal indeholde en tabel med en række for hver delaflevering og to kolonner der indeholder følgende:

1. En liste af færdigimplementerede brugsscenarier og underordnede opgaver (samt evt. estimater I har lavet) i delafleveringen.
2. De væsentlige ændringer i designet (inkl. begrundelse).

#### Bilag B: GitHub-oversigt

Rapporten skal suppleres af en release i jeres GitHub-arkiv, som kan tilgås af underviserne og censorerne. Dette bilag skal tydeligt angive hvordan arkivet tilgås og hvordan det er struktureret.

#### Bilag C: Reviews

Dette bilag skal indeholde de reviews som I har lavet af andre gruppers projekter:

1. Review af krav
2. Review af design
3. Review af kode
4. Review af release.

### Formelle krav for eksamensafleveringen

Gruppen skal aflevere en rapport via Absalon og en navngiven release af applikationen i GitHub.

### Rapporten

Rapporten skal afleveres som PDF, A4 sideformat med 2,5 cm margin, 1 1/2 linjeafstand, sidenumre (nederst i midten), og brødtekst i 12-punkts Times New Roman eller lignende skrifttype.

Rapporten må være på **maksimum 30 sider** (à 400 ord per side), eksklusive bilag.

Rapporten skal indeholde en forside med navne og e-mailadresser på gruppens medlemmer, holdnummer og navn på instruktor, og navn på GitHub-arkivet. Rapporten skal derudover indeholde en indholdsfortegnelse.

Rapporten kan være skrevet i LaTeX (f.eks. med stilfilen [DIKU-report.cls]), Word eller andre tekstbehandlingsprogrammer, efter frit valg.

Brug gruppefunktionen ved aflevering på Absalon til at angive medlemmerne i gruppen.

### Den afleverede version af applikationen

Rapporten skal følges af en version af applikationen som en release i GitHub. Derved laves en kopi af indholdet i GitHub-arkivet, som vil blive gemt som et filarkiv (.zip eller .tar.gz). Oversatte binærfiler skal ikke tilføjes til arkivet. Denne version (release i GitHub) som afleveres til eksamen skal navngives "Exam".

Den afleverede release skal kunne oversættes uden fejl således at det kan installeres og afvikles. En vejledning i hvordan man oversætter, installerer og kører applikationen skal vedlægges (f.eks. i en INSTALL.md fil i roden af arkivet); se i øvrigt krav til rapporten ovenfor (bilag B). Husk at sikre inden aflevering at jeres releasearkiv kan hentes ned og pakkes ud, at programkoden kan oversættes og installeres i det beskrevne miljø, og at applikationen kan afvikles.

## Referencer


- [**Agile**] Robert C. Martin and Micah Martin. *Agile Principles, Patterns, and Practices in C#*, Pearson Education, Inc., 2007.
- [**Cockburn**] Alistair Cockburn. Structuring use cases with goals, 1997.
- [**DIKU-report.cls**] The CPH STL, Tools used in the project, http://www.cphstl.dk/WWW/tools.html.