Softwareteams i Programmering B

Materiale og opgaver til et forløb om softwareudvikling.

*I programmering B arbejdes med udvikling af større og mere komplekse systemer end på C-niveauet. I dette forløb skal eleverne samarbejde i projektteams, der efterligner dele af arbejdsformen hos et udviklingsteam. Eleverne får indsigt i de forskellige kompetencer der er i spil, når et virkeligt program skal udvikles, i samarbejde mellem forskellige interessenter.*

*Forløbet er designet med øje for den didaktik der er beskrevet i læreplanen, i særdeleshed use-modify-create og stepwise improvement.*

Indholdsfortegnelse

[Resumé 2](#_Toc511121733)

[Abstrakte dokumentationsformer 2](#_Toc511121734)

[Brugerhistorier 3](#_Toc511121735)

[Sekvensdiagrammer 3](#_Toc511121736)

[Klassediagram 4](#_Toc511121737)

Resumé

I lærerplanen til Programmering B findes faglige mål, der handler om at udvikle virkelighedsnære programmer med høj kompleksitet:

1. Anvende avancerede konstruktioner i et programmeringssprog.
2. Redegøre for arkitekturen af programmer på forskellige abstraktionsniveauer, herunder relation mellem brug og funktion.
3. Rette, tilpasse og udvide avancerede programmer.
4. Arbejde inkrementelt og systematisk i programmeringsprocessen

I dette forløb arbejdes med de fire mål, ved at:

1. Bruge objektorienteret programmering
2. Samarbejde om at udvide et komplekst system, og tage udgangspunkt i brugeren ønsker til programmet.
3. Udgangspunktet for projektet er et fungerende program, der naturligt kan udvides i flere forskellige retninger.
4. Udvidelserne til programmet sker i en struktureret proces.

Kernestoffet i forløbet er fortrinsvis:

1. Programmeringssprog og elementer i programmers opbygning, herunder variabler, typer, udtryk, kontrolstrukturer, parametrisering/abstraktionsmekanismer, rekursion, polymorfi og algoritmemønstre
2. Arbejdsgange og systematik i programmeringsprocessen, herunder test of fejlfinding.
3. Abstrakte programmeringsbeskrivelser og dokumentation.

De abstrakte dokumentationsformer danner udgangspunkt for elevernes samtaler og planlægning af arbejdet.

Abstrakte dokumentationsformer

I dette forløb anvendes tre forskellige dokumentationsformer til at specificere de opgaver, som eleverne skal arbejde med. De tre dokumentationsformer har en kobling til de tre lag i trelagsmodellen[[1]](#footnote-1), som tit anvendes til beskrivelse af IT-systemer.

Præsentationslag:

Brugerhistorier

Applikationslag:

Sekvensdiagrammer

Datalag:

Klassediagram

Brugerhistorier

Brugerhistorierne beskriver en afgrænset interaktion mellem brugeren og systemet. Følgende kendetegner en effektiv brugerhistorie:

* Den er skrevet i naturligt sprog, idet den skal fungere som kommunikationsmiddel mellem programmets bruger (Der har domænekendskab, men ikke nødvendigvis teknisk indsigt) og udviklingsteamet.
* Den er evaluerbar. Når brugerhistorien er implementeret, kan brugerhistorien bruges som udgangspunkt for en test af programmets funktion.
* Detaljegraden giver mulighed for at implementere brugerhistorien på en entydig måde, uden at blive så omstændig, at den hæmmer forståelsen.
* Brugerhistorierne nummereres - eller navngives på anden vis - så eleverne kan henvise til dem på en struktureret og entydig måde.

I forløbet er der forberedt et dokument med brugerhistorier, men man kan også lade eleverne formulere nye brugerhistorier.

Eksempel på en brugerhistorie:

|  |
| --- |
| Brugerhistorie 1: Bestil bøger Denne brugerhistorie bruges, når butikken har brug for flere eksemplarer af en bestemt bog.   1. Brugeren vælger en bog fra listen af bøger. 2. Brugeren klikker på knappen ”Bestil flere”. 3. Programmet viser en dialog, hvor brugeren kan se:    1. Bogens titel    2. En Spinbox, hvor man kan vælge det antal bøger man ønsker at bestille. (0-10)    3. En ”OK”-knap 4. Brugeren vælger det ønskede antal bøger og klikker ”OK”.    1. Hvis antallet er større end 0, oprettes en transaktion, hvor bøgernes pris trækkes fra kassen, og bøgerne tilføjes lageret.    2. Hvis antallet er 0 oprettes ingen transaktion, og der sker intet med lageret. 5. Programmet lukker dialogen. |

Sekvensdiagrammer

For at dokumentere programmets applikationslag bruges sekvensdiagrammer. Et sekvensdiagrambeskriver de beskeder der sendes gennem programmet for at implementere en bestemt brugerhistorie. Fordelen ved sekvensdiagrammet er, at det er tættere på koden end for eksempel et flowdiagram. Til gengæld er det en del sværere for eleverne at producere.

Derfor bør sekvensdiagrammerne stilladseres på den måde, at eleverne

* først implementerer en brugerhistorie hvor sekvensdiagrammet er givet på forhånd.
* derefter implementerer de evt. en brugerhistorie hvor de starter med at skrive koden, og derefter producerer det tilhørende sekvensdiagram.
* til sidst kan de stærkeste elever selv producere et sekvensdiagram til en brugerhistorie, inden de har skrevet koden.

Eksempel på et sekvensdiagram:

|  |
| --- |
|  |

Klassediagram

Til at dokumentere programmets data bruges klassediagrammet. Det giver et overblik over programmets dele, og koblinger mellem dem. Det giver et strukturelt overblik, som man ikke kan få ved at kigge på koden. Dette overblik er det ofte svært for eleverne at skaffe sig, når der er tale om komplekse systemer.

Klassediagrammet skal løbende udvides når eleverne arbejder med projektet.

Boghandelens klassediagram:

|  |
| --- |
|  |

Projektteamets sammensætning

I projektet inddeles eleverne i grupper, hvor hver elev får en rolle, der svarer til en rolle man kunne finde i et traditionelt udviklingsteam. Der er flere formål med at give eleverne roller:

* Det sætter struktur på deres arbejde
* Det tvinger dem til at kommunikere om programmet og om deres løsning
* Det giver eleverne et indtryk af de mange forskellige kompetencer der er i spil, når et program skal udvikles. Det styrker deres karrierekompetence ved at give dem mulighed for at se sig selv i en eventuel fremtidig rolle indenfor faget.

Roller

* Projektleder: Har det sidste ord, når gruppen skal vælge hvilken rækkefølge brugerhistorierne skal implementeres.
* Arkitekt: Har det sidste ord når gruppen tager beslutninger angående programmets opbygning. Skal der laves nye klasser? Hvor skal snitfladen mellem klasserne placeres?
* Programmør: Har ansvaret for koden. Programmøren skal ikke nødvendigvis skrive al koden, men beslutter hvilke kontrolstrukturer, datatyper, osv., der skal bruges i koden. Programmøren skal kende alle tekniske detaljer i koden.

Boghandelen: Et forløb i programmering b

Her præsenteres et undervisningsforløb hvor eleverne i projektteams - og ved hjælp af abstrakte dokumentationsformer - skal lære at videreudvikle på et komplekst system: Et program til en boghandel. Programmet skal sætte de ansatte i boghandelen i stand til at overvåge og administrere alt i butikken: Indkøb og salg af bøger, løn til de ansatte, lagerbeholdning osv.

Der er beskrevet et antal brugerhistorier, og til nogle af dem er er lavet sekvensdiagrammer. Der er lavet et udgangspunkt for programmet, som eleverne arbejder videre på.

Lektionsplan

Forløbet er ca. berammet til moduler på 2x50 minutter. Tiden pr lektion betyder egentlig ikke så meget, men kan nemt justeres efter behov. Antallet af lektioner kan også justeres efter klassens tempo og ambitioner.

## Modul

Eleverne fordeles i grupper på 3-4 personer.

Programmet udleveres, og eleverne begynder at orientere sig i brugerfladen. Formålet er at lære programmets funktioner at kende. Læreren svarer på spørgsmål, og efter lidt tid kan man sammen gennemgå klassediagrammet. Forhåbentlig vil eleverne kunne forstå, hvordan den overordnede struktur i programmet er.

Nu skal eleverne præsenteres for arbejdsformen i projektet. Hver gruppe skal agere udviklingsteam, og hver elev får en rolle, som minimum:

1. Projektleder: Har det sidste ord, når gruppen skal vælge hvilken rækkefølge brugerhistorierne skal implementeres.
2. Arkitekt: Har det sidste ord når gruppen tager beslutninger angående programmets opbygning. Skal der laves nye klasser? Hvor skal snitfladen mellem klasserne placeres?
3. Programmør: Har ansvaret for koden. Programmøren skal ikke nødvendigvis skrive al koden, men beslutter hvilke kontrolstrukturer, datatyper, osv., der skal bruges i koden. Programmøren skal kende alle tekniske detaljer i koden.

Eleverne skal nu læse alle brugerhistorierne igennem, så de ved, hvad boghandleren forventer, at de implementerer i løbet af projektet.

Sammen gennemgår klassen en udvalgt brugerhistorie, hvor sekvensdiagrammet gennemgås og sammenholdes med koden. (Forslag: ”Brugerhistorie 1: Bestil bøger”, der allerede er implementeret i koden.)

Herefter skal gruppen implementere en udvalgt brugerhistorie. Vælg en, hvor sekvensdiagrammet er inkluderet, men koden endnu ikke er skrevet. (Forslag: ”Brugerhistorie 2: Justér markup”. Den er simpel, der skal laves et GUI-element, og den vil have direkte effekt på programmets funktion.)

## modul

Nu skal projektgrupperne til at planlægge udviklingen af resten af programmet. Gruppen skal forsøge at implementere så mange af brugerhistorierne som muligt, men de skal afslutte én brugerhistorie, før de starter på den næste.

**Øvelse i grupperne: Vurdering af tidsforbrug for de enkelte brugerhistorier.**

Gruppemedlemmerne tager hver et sæt ”planning-poker”-kort.[[2]](#footnote-2) Kort til print kan findes mange steder, blandt andet: <http://www.hennebert.org/web/download/63-planning-poker>

Man kan også vælge at bruge et online værktøj, f.eks. <https://www.planitpoker.com/> (gratis ”quick play”-funktion, der vil passe fint til dette formål)

Gruppen gennemgår brugerhistorierne en efter en, og estimerer tidsforbruget. Laveste og højeste bud får lov at fortælle om deres begrundelse. Projektlederen noterer den tid gruppen kommer frem til.

Efter pokerspiller kan gruppen vælge hvilken brugerhistorie de vil implementere først. **Projektlederen** tager den endelige beslutning.

Hvis der ikke findes et sekvensdiagram i forvejen, skal gruppen nu lave et sekvensdiagram. Det er vigtigt, at de prøver at strukturere opgaven, før de begynder at kode. **Arkitekten** har det endelige ord. Klassediagrammet skal opdateres, hvis der tilføjes nye klasser eller foretages ændringer i eksisterende klasser.

Så skal koden skrives. Gruppen kan fordele opgaverne mellem sig, men **programmøren** har ansvaret for alle detaljer i koden. Desuden er det **programmøren** der har ansvaret for at teste, og programmet opfylder brugerhistorien.

Resten af forløbet består af, at grupperne forsøger at nå så langt som muligt med programmet til boghandelen. Efter hver brugerhistorie skal gruppen fortælle læreren om status (Læreren kan tage rollen som boghandler)

Med tiden, og hvis læreren ønsker det, kan man lade gruppen skrive sine egne brugerhistorier, så de kan arbejde videre med programmet som de ønsker.

1. <http://iftek.dk/client-server-og-trelagsarkitektur> [↑](#footnote-ref-1)
2. https://en.wikipedia.org/wiki/Planning\_poker [↑](#footnote-ref-2)