ProjDat2015 - Bibliotek udlånssystem

Oliver Sejling Kogut 010694
Thomas Nyegaard-Signori 141093
Casper Helms 260294

Instruktor: Markus Wittorf

Marts 23, 2015

Indholdsfortegnelse

1 Problemformulering

1.1 Problemet

Et lille bibliotek på Nørrebro i København vil gerne have et udlåningssystem. På nuværende tidspunkt styres udlån og udsending af rykkere igennem manuel nedskrivning og opslag i bøger, og der kræves et manuelt opslag i en række bøger for at tjekke om udlånsdatoer er overskredet, hvorefter man kontakter låneren. Det er meget tidskrævende. De vil derfor gerne have en forsimpling og automatisering af dette i form af et it-system, som blandt andet skal kunne fungere som selvbetjening, ved at benytte en computer i biblioteket. Således kan en låner returnere eller låne en bog uden at være i kontakt med en bibliotekar, så der principielt set ikke behøver at være en ansat til stede.

Vi har altså et anvendelsområde som består af biblioteket og de ansatte som skal bruge en effektiv måde til at holde styr på deres bøger, udlån og udsending af rykkere til deres lånere. Derudover vil lånerne også være en del af anvendelsesområdet, da de selv skal kunne låne og returnere en bog ved brug af systemet. Ligeledes har vi et problemområde som består af alle disse bøger, udlån, lånere og rykkere som et it-system skal simplificere og overskueliggøre.

1.2 Scenarios

Et scenario er en overskueliggørelse af brugen af et system i konkrete og præcise skridt. Det er ofte en illustration af en mindre del af et system, og ikke af systemet i sin helhed[s. 48, kilde 1].
Vi har brugt scenarios til at vise hvorledes bibliotekets udlån og rykker tjek fungerer på nuværende tidspunkt.

Lån af bog på nuværende tidspunkt:

Scenario name	LånAfBog
Participating actor instances	bob: Låner alice: Bibliotekar
Flow of events	1. Bob finder en bog han vil låne og går op til Alice med den. 2. Alice finder en bog frem hvori hun skriver Bobs mobil nr., dato, og info omkring bogen. 3. Bob forlader biblioteket med bogen og har nu lånt den.

Fig. 1.1:
Det ses at det nu er en meget ineffektiv måde udlån af bøger fungerer på, da bibiliotekaren manuelt er nødt til at nedskrive oplysninger på både bøger og bruger ved hvert udlån. Dette vil bestemt kunne effektiviseres, ved brug af et it-system hvori et register over bøger, lånere og udlån er registeret(dette er nærmere beskrevet med en usecase (fig. 3.1, s. 9)).

Tjek og udsending af rykker på nuværende tidspunkt:

Scenario name	RykBog
Participating actor instances	alice: Bibliotekar
Flow of events	1. Alice finder alle bøger frem hvori udlån står skrevet ned i. 2. Alice går manuelt alle bøger igennem for at tjekke om nogle udlån har overskreddet returneringsdatoen. 3. Alice ringer til alle de lånere som skal rykkes for deres bog. 4. Alice ligger bogen tilbage, og gennemgår den igen på et andet tidspunkt for at gøre det samme igen.

Fig. 1.2:
Det ses at det er meget tidskrævende og frustrerende at henholdsvis tjekke for udlån der ikke er returneret og at skulle ringe hver enkelt person op for at rykke dem, som det ser ud nu. Her vil et system i høj grad aflaste de ansatte da det hele vil foregå automatisk. Det kunne eventuelt gøres ved at systemet automatisk tjekker hvor lang tid en bog har været udlånt, og hvis datoen er overskredet sendes en rykker på mail(dette er nærmere beskrevet med en usecase(fig. 3.2, s. 10)).

1.3 Funktionelle minimuskrav

Funktionelle krav er de deciderede funktioner, som et it-system skal understøtte. Så som hvad programmet skal kunne[s. 12, kilde 1]. Både de funktionelle og ikke-funktionelle krav, er lavet for at skabe overblik over hvad systemet skal kunne.

- Systemet skal fungere således, at det skal kunne selvbetjenes af lånere både ved udlån og returnering af bøger.
- Bibliotekarer skal yderligere kunne registrere henholdsvis bøger og lånere i systemet.
- Systemet skal kunne tjekke for og udsende rykkere/påmindelser på email, til lånere der har overskredet returneringsdatoen.

Når minimumskravene er opfyldt, vil det være muligt for kunden at ønske mere funktionalitet, hvis tiden er til det og det ønskede krav ikke er for stort. Ovenstående er blot tænkt som værende det simpleste endelige system.

1.4 Ikke-funktionelle krav

Ikke-funktionelle krav, fortæller noget om design af systemet, effektivitet, sikkerhed, sprog osv. altså krav, som ikke er direkte relateret til systemets funk-

tioner[s. 12, kilde 1].

- Det skal være en hjemmeside, som let kan tilgås fra alle styresystemer via en browser.
- Hjemmesiden skal kun kunne tilgås, hvis man er koblet på bibliotekets netværk.
- Systemet skal generelt sikres mod misbrug udefra, hvad end det er rettet mod bruger oplysninger eller ved input af skadelig tekst i indtastningsfelter.
- Systemet skal være brugervenligt og simpelt. Dette kan gøres ved at udføre brugertest, og kvalitativ forståelse ved interview med brugere.

2 Projektplan

2.1 Overordnet arbejdsstruktur

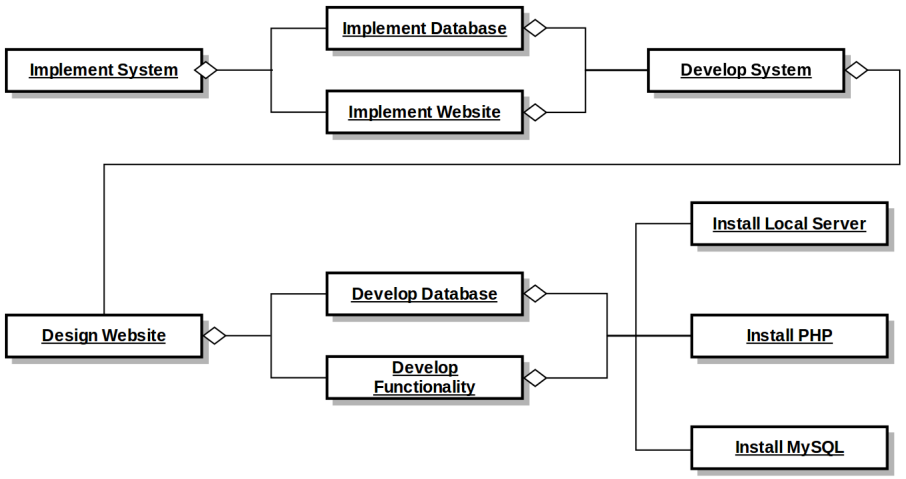

Fig. 2.1:
Figur 2.1 herover viser en simpel skitse af det arbejde der vil foregå og i hvilken rækkefølge. Figuren er en Work Breakdown Structure for hele projektet, lige fra hvad der kræves installeret (på udviklers computer), for at man kan gå i gang med at udvikle systemet, designe det, for derefter at implementere det hos kunden.

Den skal forstås således at arbejdsprocesserne løber fra slutningen af figuren (install local server, install php, install mySQL), og følger pilen mod venstre. Det er ment sådan at det der står på højre side af en pil kræves for at kunne udføre det på venstre side.
Eks:
For at man kan 'Develop Database' og 'Develop Functionality', kræves det at man først har opfyldt 'Install Local Server', 'Install PHP' og 'Install MySQL'.

I forhold til figuren bruger vi 'develop' som betegnelse for den deciderede udvikling, hvorimod 'implement' er måden hvorpå det udviklede skal sammensættes og installeres på det, som systemet skal køre på.
Boksen 'Develop Functionality' skal forstås således, at det er systemets funktionalitets aspekt, så som knapper, tekstfelter, funktioner osv. Det er tænkt således at det visuelle design ikke hører under denne boks, men derimod under 'Design Website'. 'Develop Database' og 'Develop Functionality' afhænger selvfølgelig af hinanden og bør udvikles side om side.

2.2 Tidsplan

Foreløbig skitse af vores tidsplan indholder tre iterationer i form af prototyper. Alle tre iterationer forventes at være færdige umiddelbart før deres tilsvarende delrapport ((23/4, 21/5, 4/6)det er dog ikke fuldstændig fastlagt). Altså første iteration/prototype forventes at være færdig umiddelbart før delrapport 2 skal afleveres osv.

Første iteration:
Første iteration forventes at indeholde alle minimuskrav (listet i del 1.3 tidligere), uden synderligt henblik på design og brugervenlighed. Denne iteration forventes at være den mest krævende såvel tidsmæssigt som arbejdsmæssigt.

Anden iteration:
Prototypen for første iteration vil blive testet på biblioteket, hvis der bliver mulighed for det. Ud fra resultatet af testen vil vi udvikle en ny prototype, eventuelt i forhold til brugervenlighed, og muligvis nye eller forbedrede funktioner, hvis kunde eller brugere hare forslag.

Tredje iteration:
Prototypen for anden iteration vil igen blive testet, og en ny prototype vil blive udviklet hvis behovet viser sig at være der. Det kan som det simpleste være det visuelle design eller nogle få ekstra funktioner, hvis der er behov for det.

2.3 Budget for ressourcer

Det eneste vi umiddelbart har behov for, for at implementere systemet, er et domæne, en webhost/database. Dette ejer biblioteket dog allerede, så det vil derfor være oplagt blot at implementere systemet under dette. Andre nødvendige ressourcer kan naturligvis opstå undervejs i projektet, og budgettet må fastlægges på det tidspunkt.

2.4 Roller og ansvar

2.4.1 Kunde

Vores kunde er biblioteket. De har det ansvar at fastslå så detaljeret som muligt, hvad it-systemet skal kunne, hvad de skal bruge de til og hvorfor. Dette kan som udgangspunkt formidles videre til Product Owner, Simon Shine. Vi vil dog også selv være i direkte kontakt med kunden, da de trods alt selv bedst ved hvad de vil have systemet skal kunne.

2.4.2 Product Owner

Vores Product Owner er som førnævnt Simon Shine. Han har sammen med biblioteket fastlagt hvilket system, der er brug for og hvad det skal kunne.

2.4.3 Team

Vi (Casper, Thomas, Oliver) er udvikler teamet. Vores ansvar er at udvikle et system til biblioteket, som opfylder deres ønsker, krav og behov. Både i form af visuelle design-ønsker, funktionalitet og sikkerhed.

2.4.4 Scrum Master

Da teamet kun består af tre personer, forventes Scrum Master rollen ikke at være nødvendig, men hvis det modsatte skulle vise sig er Thomas Scrum Master. Han vil ved uenigheder, sørge for at alle har noget at skulle have sagt, og derved få løst de opståede problemer så godt som muligt.

2.5 Risikovurdering og behandling

Med enhver form for projekt følger naturligvis en række risici, men særligt IT-projekter er nok dem hvor der er flest risici, og der hvor der kan opstå flest problemer. IT-systemer er ofte meget komplekse, og et problem kan tit løses på hundrede forskellige måder. Der vil ofte være delte meninger om hvilken metode der er mest effektiv, og det kan derfor lægge op til problemer og uenigheder. Der er også ofte risiko for, at det først går op for en i løbet af projektet, at det man er i gang med at lave, ikke vil fungere i praksis eller, at der findes en meget lettere og bedre løsning på problemet.

Ovenstående er naturligvis svært at undgå, hvis ikke umuligt. Vi forsøger at forebygge og undgå så mange af disse problemer som muligt, ved at planlægge projektet godt på forhånd, og fuldt ud slå fast hvad kundens nuværende problem er, hvorfor de vil have et it-system, og hvordan det skal fungere efter implementeringen.

Hvis der alligevel opstår problemer, hvilket der sandsynligvis vil gøre, må vi drøfte problemerne i fællesskab, eventuelt med kunden og/eller product owner, og finde ud af hvordan vi bedst muligt løser det. Altså få fastlagt hvor problemet ligger, hvorfor det er et problem, og sammen vurdere muligheder for at løse det, og forhindre, at det sker igen.

Derudover er der naturligvis risici omkring sikkerhed og datatab. Der er risiko for at lånere går ind og registrerer lån på en lang række bøger, blot for at lave uorden i systemet, og skabe en masse fejlagtig data, så der ikke længere er styr på hvem der har lånt hvad osv. Dette er svært at forhindre, men vi har gjort det således at man kun kan tilgå systemet, når man befinder sig på biblioteket, og håber så på at det kun er folk med interesse for biblioteket der bruger det, og at de gør det fornuftigt. Dette er vendt med kunden og de er enige i, at det bør kunne gøres således, da de kender de folk der kommer på biblioteket.

3 Skitse af system- og softwarearkitekturen

3.1 Skitse af undersystemer

Det overordnede system kan opdeles i 5 undersystemer der her skitseres med overodnet funktionalitet og interfaces.

3.1.1 Lån og aflevering af bøger

Dette undersystem står for opdateringen af statussen på bøger, altså går udlån og returnering igennem dette system. Brugeren af undersystemet vil indirekte kommunikere med databasen såfremt deres input er korrekte. Bøgernes status ændres afhængigt af brugerens input. Ydermere skal dette undersystem have et interface til Rykker undersystemet så denne kan holde styr på udlånsdato mm.

3.1.2 Rykkere

Dette undersystem står for den semi-automatiske rykker der bliver sendt pr. e-mail til en bruger af bibliotekssystemet. Kun bibliotekaren har adgang til dette undersystem, dog sker ændring af database ganske automatisk. Systemet skal kunne sende e-mails automatisk dog skal bibliotekaren bekræfte hvem rykkeren skal sendes til og hvornår.

3.1.3 Oprettelse af bøger og brugere

Dette undersystem står for indskrivning af nye bøger samt nye brugere. Dette undersystem er kun tilgængeligt for bibliotekaren. Her har bibliotekaren direkte indflydelse på hvad der står i databasen, derfor er det vigtigt at være opmærksom på at de indtastede informationer er korrekte.

3.1.4 Server og database

Dette undersystem behandler al data som måtte opstå ved brug af systemet. Det skal sørge for bøgernes status er konsistent igennem forløbet, altså at en bog kun kan lånes af én person af gangen og lignende situationer. Data i form af udlånsdatoer, brugere, og bognavne skal også sikres og må ikke gå tabt i tilfælde af fejl eller nedbrud(dette står nærmere beskrevet under afsnittet 2.5 Risikovurdering og behandling). Kontakten mellem input dataen fra brugeren og serveren vil foregå ved brug af serverside sproget PHP og vil gennem database-håndteringssproget mySQL indskrives i databasen.

3.1.5 Hjemmeside

Dette undersystem vil være den generelle brugerflade for alle brugere af systemet, låner såvel som bibliotekar. Ved brug af markup sproget HTML konstrueres det basale udseende af siden og tilgang til server og database gennem inputfelter og knapper. Derudover vil der blive anvendt en smule CSS til yderligere visuelt design af siden, hvis dette viser sig at blive nødvendigt.

3.2 Use Cases

Use case name	LånAfBogNy
Participating actor instances	bob: Låner
Flow of events	1. Bob finder en bog han vil låne og går hen til selvbetjenings computeren med den. 2. Bob vælger funktionen lån bog. 3. Bob indtaster sin mail(hvis han er registreret), og nummeret på bogen. 4. Bob har nu registreret lånet, og forlader biblioteket.

Fig. 3.1:
Det ses at ved brug af systemet kræver det kun låneren selv for at kunne låne en bog. Derudover forhindres menneskelige fejl, da en bog kun kan lånes hvis den er registreret i systemet, så den accepteres ikke hvis der tastes forkert bognummer.

Use case name	RykBogNy
Participating actor instances	alice: Bibliotekar
Flow of events	1. På sin computer vælger Alice funktionen der tjekker for udlån der har overskredet returneringsdatoen. 2. Alice får en liste over alle de udlån der har overskredet returneringsdatoen. 3. Alice klikker på knappen send rykkere. 4. Systemet sender automatisk en rykker ud på email, til hver af de mails der er registreret på udlån af bøger der endnu ikke er returneret.

Fig. 3.2:
Med denne automatisering kan man ved sammenligning med scenariet figur 1.2, se at rykkersystemet i den grad vil hjælpe de ansatte, da de ikke længere selv behøver manuelt at søge en række bøger igennem, for derefter at ringe lånerne op for at rykke dem.

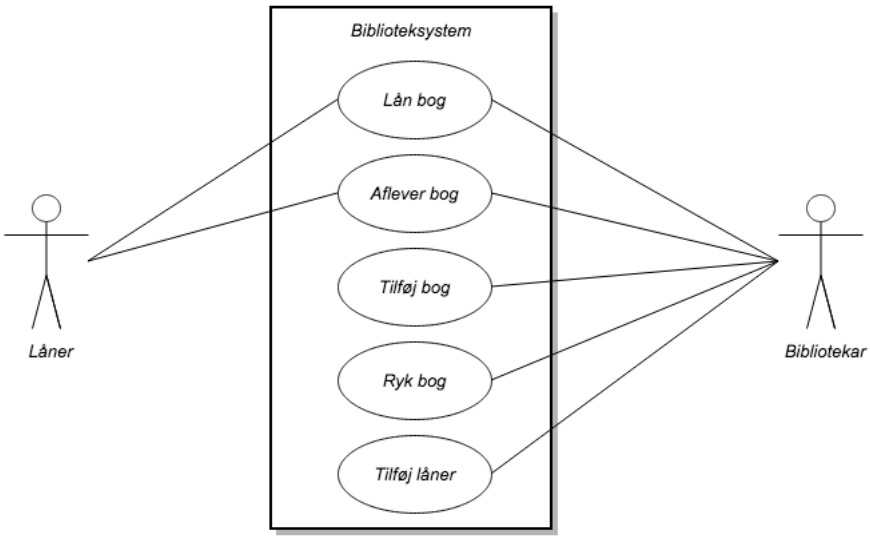

Fig. 3.3:
Figur 3.3 herover er en illustration af hvilke muligheder, henholdsvis låner og bibliotekar har i systemet. Det skal ikke forstås således at de to aktører interagerer med hinanden, men blot at de har forskellige funktioner i samme system.

4 Projektaftalen

Aftalen er indgået mellem os, biblioteket og product owner Simon Shine.

4.1 Overordnet aftale

Aftalen går ud på, at vi skal udvikle et it-system, som skal simplificere udlån, returnering og henholdsvis tjek og udsending af rykkere. Derudover skal det ligeledes fungere som et register over bogbeholdningen, hvor man kan både tilføje og fjerne bøger.
Systemet skal ligeledes fungere som selvbetjening, således at lånere vil kunne gennemføre et lån og/eller en returnering af en bog, selvstændigt og uden hjælp fra en ansat.

4.2 Implementering

Biblioteket har både domæne og database til rådighed, og det vil derfor være oplagt at lave systemet i form af en hjemmeside, som kan ligge som et underdomæne til deres nuværende side. Da systemet så vil komme til at ligge på nettet, er det et krav at underdomænet kun kan tilgås, hvis man er koblet på bibliotekets netværk. Der vil det så være op til biblioteket, at beslutte om der skal stå en eller flere computere til rådighed til selvbetjening, eller om man vil give lånere adgang til deres netværk således, at de kan foretage lån og returneringer fra deres egen computer, smartphone eller anden enhed.

Der vil være mulighed for senere i forløbet at tilføje yderligere funktioner til systemet, hvis tiden er til det, såsom at lånere har mulighed for at søge på bøger, reservere bøger eller andre idéer der måtte opstå under udviklingsprocessen.

4.3 Muligheder for test af prototyper

Vi har aftalt at der højst sandsynligt vil være mulighed for, at teste prototyper på biblioteket, for evt. at forbedre systemet, eller rette eventuelle fejl der måtte opstå. Det vil også være oplagt at overvære udlån og returneringer fra lånere, for at se om der er noget ved designet, der skal ændres så låneren lettere finder de knapper han/hun har brug for. Der vil det ligeledes være oplagt at høre fra lånerne hvad de synes om systemet, i forhold til forbedringer eller andre funktioner.

4.4 Deadline

Som udgangspunkt vil vores arbejde ophøre d. 14/06-2015, sammen med vores kursus på Københavns Universitet. Det skal dog siges, at der sagtens kan være mulighed for at forlænge arbejdet, hvis vi kan blive enige om det. Det forventes at der som minimum vil være en fungerende prototype, som biblioteket kan gå videre med, hvis ikke vi vælger at fortsætte. I forhold til en endelig implementering af systemet, er der ikke aftalt noget specifikt omkring, men det bliver formegentlig aktuelt så snart, der er udviklet en fungerende prototype af systemet, som er klar til at blive testet.

5 Intern Projektetablering

5.1 Arbejdsform

Overordnet set er vores arbejdsform baseret på iterationer som beskrevet under del 2.2. Vi vil under arbejdet med projektet benytte os af et par værktøjer, som er beskrevet herunder. Dette gøres både for at danne overblik, uddelegere opgaver og deling af kode. Dette giver også mulighed for at kunne arbejde hver for sig ved at dele opgaver op, hvilket vil være oplagt at benytte sig af, da vi ikke alle nødvendigvis har tid til at arbejde samtidigt. Vi vil naturligvis mødes omkring hver anden uge i løbet af projektarbejdet, og drøfte hvorledes det står til i forhold til arbejde og tid.

5.1.1 SCRUM

Vi vil benytte os af den agile udviklingsmetode SCRUM igennem vores projektforløb. SCRUM er en iterativ udviklingsprocess/arbejdsmetode. Man fastslår en række sprints på en 1-4 uger af gangen, hvorefter man mødes og taler om hvad man har lavet, og hvad man har fundet ud af. Vi har valgt sprint længder af 2 uger, så omkring to for hver af vores tidligere nævnte iterationer. Ideen med SCRUM er at man løbende holder styr på hvor langt man er nået, hvor meget der er tilbage, og hvis der skal foretages ændringer[s. 10, kilde 2]. En beskrivelse af vores SCRUM roller, findes under sektion 2.4 Roller og ansvar.

5.2 Værktøjer

5.2.1 Github

Under arbejdsforløbet vil vi i udvikler teamet, benytte os af github således, at vi kan dele nyttige dokumenter og arbejde på kode uden rent fysisk at sidde sammen. Github er ligeledes et godt værktøj til at danne overblik over alle ændringer i kode, og system versioner. For at dette skal kunne fungere optimalt, kræver det naturligvis god koordinering således at to personer ikke sidder og laver eller redigerer det samme. Til dette bruges Trello.
Link til Github: www.github.com

5.2.2 Trello

Trello benytter vi os af til at skabe overblik over hele projektsituationen. Ved inddeling af arbejdsopgaver (underopgaver osv.) og registrering af hvem der er igang med at arbejde på hvad, hvad der er færdigt, hvad der mangler. Dette er et super godt værktøj til at overskueliggøre et stort projekt. Det kan sammenlignes med et Scrum board[s. 23, kilde 2](kaldes i kilden for task-board), hvorpå man hænger sedler med hvad der skal laves osv.
Link til Trello: www.trello.com

6 Bibliografi

1. Bruegge, Bernd og Dutoit, Allen H. 2014. "Object-Oriented Software Engineering Using UML, Patterns, and Java third edition"

2. Jeff Sutherland. 2010. "SCRUM Handbook"

7 Afleveringsopgaver

7.1 OOSE 1-6

Functional requirements:
"The TicketDistributor must enable a traveler to buy weekly passes."
"The TicketDistributor must provide(!) a phone number to call when it fails."
(læg mærke til at den 'provider' et nummer. Derfor er det funktionelt. Hvis nummeret blot var skrevet på selve maskinen ville det være nonfunctional.)
Nonfunctional requirements:
"The TicketDistributor must always be available."
"The TicketDistributor must be written in Java."
"The TicketDistributor must be easy to use."

7.2 OOSE 1-8

Når der tales om application domain, menes der den reele account som man har hos banken, hvilket er de to første gange det nævnes. Hvorimod når der tales om solution domain, menes der systemets repræsentation og håndtering af account, altså de to sidste gange det nævnes.

7.3 OOSE 2-6

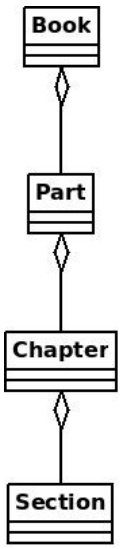

Fig. 6.1:
Klasse-diagrammet herover illustrerer at en book består af parts som består af chapters som består af sections. Med andre ord; Parts, Chapters og Sections danner sammen en bog.

7.4 OOSE 2-7

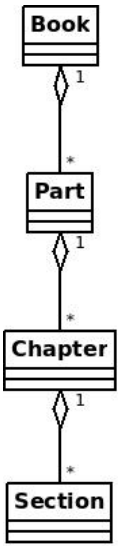

Fig. 6.2:
Klasse-diagrammet herover svarer til figur 6.1, her er der dog tilføjet multiplicity, som skal forstås således at til 1 bog kan der være flere parts. Til 1 part kan der være flere chapters, og til 1 chapter kan der være flere sections.

7.5 OOSE 2-9

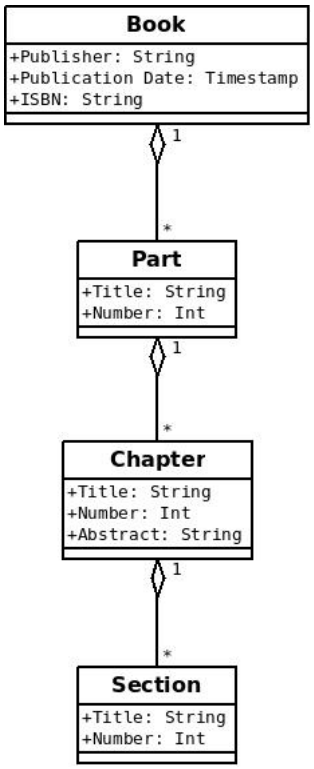

Fig. 6.3:
Klasse-diagrammet herover svarer til figur 6.2, her er der dog tilføjet nogle attributter til hver enkelt klasse.

7.6 OOSE 2-10

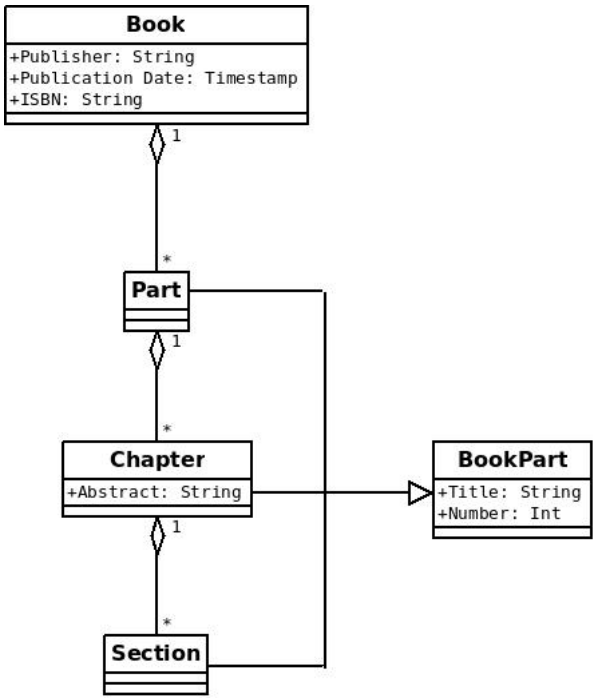

Fig. 6.4:
Klasse-diagrammet herover er konstrueret ud fra figur 6.3, da part og section har de samme attributter er der indraget en superklasse BookPart som Part, Chapter og Section nedarver fra, og derigennem får både Title og Number. Chapter og dog stadig sin egen attribut Abstract.

7.7 OOSE 5-3

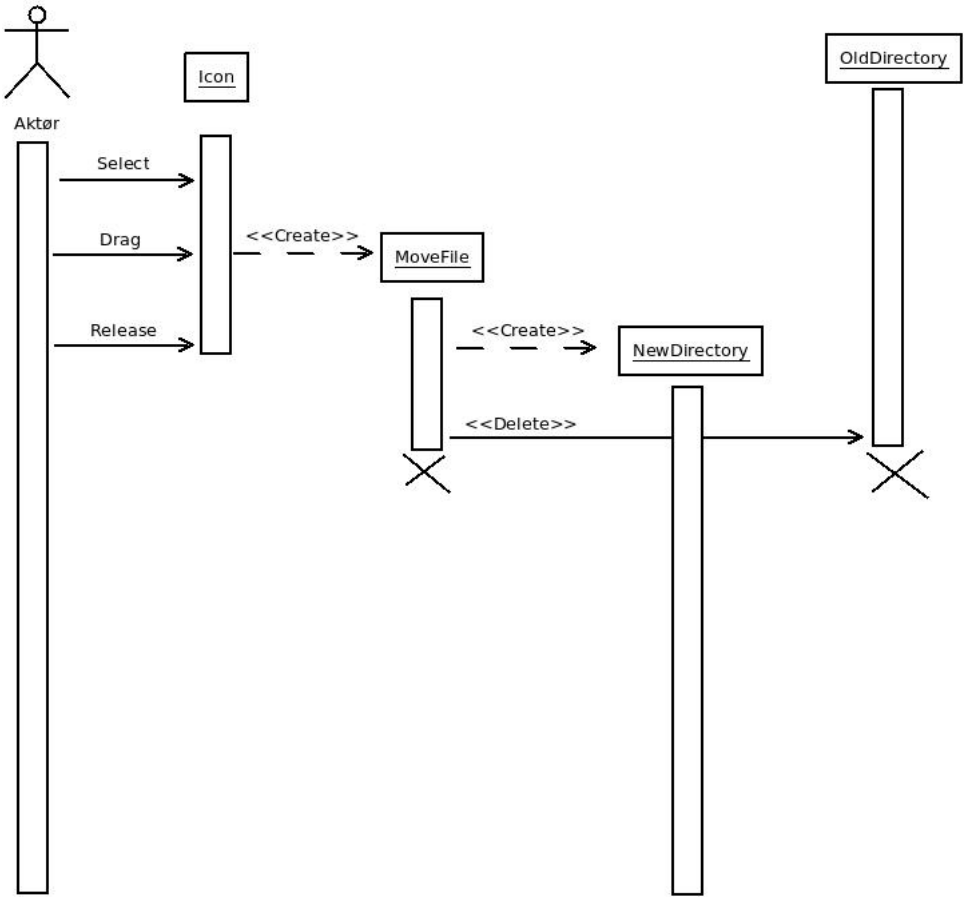

Fig. 6.5:
Sekvens-diagrammet herover illustrerer en flytning af en fil til en anden mappe. Aktøren starter med at selecte ikonet, derefter trække det til den ønskede lokation, for herefter at slippe den. Herefter udføres funktionen MoveFile som laver den nye directory og sletter den gamle.

7.8 OOSE 7-1

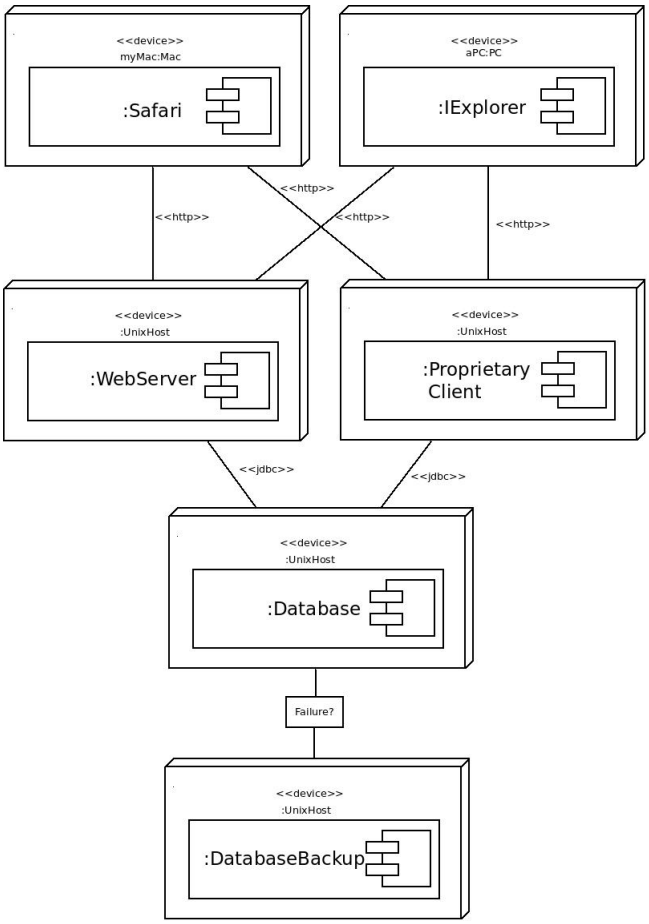

Fig. 6.6:
Deployment-diagrammet herover illustrerer hvordan man kan tilgå en database fra en mac eller pc. Enten via en webserver eller en proprietary client. Der er ligeledes to identiske databaser, hvor den ene fungerer som backup af den anden. Det skal forstås således at hvis det fejler med den første database, forbindes der til backuppen.

8 Bilag

8.1 Mødereferat 09/03-2015

Deltagere:
Udviklere: Thomas Nyegaard-Signori, Casper Helms, Oliver Sejling Kogut
Kunde: Simon Shine(Product Owner)

Da biblioteket alligevel ikke havde mulighed for at deltage i mødet på det aftalte tidspunkt, holdt vi i stedet mødet med Product Owner Simon Shine.

Vi fik igennem mødet fastlagt og forklaret hvorledes udlåning og administrering med det nuværende manuelle system fungerer, og samtidigt klargjort hvor meget et velfungerende it-system vil gavne biblioteket.
Der er umiddelbart ikke nogen faste computere til at stå på biblioteket, hvilket gør det udfordrende at lave et internt lokalt it-system. Derfor bliver vi enige om at en hjemmeside vil være en god løsning. Det skal dog kun være muligt at tilgå hjemmesiden hvis man er koblet til bibliotekets netværk. Det vil sige, at lånere vil kunne tilgå hjemmesiden fra deres egne computere eller smartphones (hvis de er koblet på bibliotekets netværk), og det ligger i særdeleshed op til at man kan forestille sig en form for selvbetjening i form af udlån og returnering af bøger. Derudover skal de ansatte naturligvis have adgang til yderligere funktioner, såsom at registrere bøger og lånere, og tjekke for og udsende rykkere på mail.

Simon udtrykker at funktionaliteten vægter langt højere end design, og at der sandsynligvis vil være mulighed for at afprøve prototyper og lignende på biblioteket.

Vi bliver i fællesskab enige om minimumskravene for dette system, hvilket står listet i første del af denne rapport, men aftaler desuden også at der sagtens vil kunne tilføjes flere funktioner og detaljer i systemet i takt med at arbejdet skrider frem, og systemet bliver afprøvet.

Biblioteket har i forvejen et domæne og en database som det vil være oplagt at lægge systemet på.