

ProjDat2015 - Bibliotek udlånsystem

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

Instruktor: Markus Wittorf

April 22, 2015

Indholdsfortegnelse

1	Abstract	3
2	IT-projektets formål og rammer	4
2.1	FACTOR	4
3	Kravspecifikation for IT-løsningen	5
3.1	Funktionelle- og ikke-funktionelle krav	5
3.2	Use case model over system-funktionaliteten	5
3.3	Use cases	6
3.3.1	Lån af bog	7
3.3.2	Udsending af rykker	8
3.3.3	Tilføjelse af bog	9
3.4	Problemområde	9
3.5	BCE-Model	10
3.6	Sekvens diagrammer	11
3.6.1	Lån af bog	11
3.6.2	Tilføjelse af bog	12
3.6.3	Udsending af rykker	13
4	Systemdesign sammenfatning	14
5	Program- og systemtest	15
6	Brugergrænseflade og interaktionsdesign	16
7	Projektsamarbejdet	17
7.1	I gruppen	17
7.2	Kunden	17
8	Bibliografi	18
9	Litteratur Review	19
9.1	A rational design process: How and why to fake it	19
9.2	Designing for usability: key principles and what designers think .	21
10	Bilag 1: Versionsstyring	22
11	Bilag 2: Changelog for projektrapporten	23
12	Bilag 3: Timeline	23

1 Abstract

This system is being developed for a library, who now handle their books, loans and customers manually. Our goal is to optimize and simplify both the book and loaner registration, and the actual loans. Besides that, it will be automatizing the creation of reminders, and send them to the loaners by email if necessary. The system will rely on a database where all necessary data are stored, and the actual program will be a website, where you can interact with the system. The system can also work as a self-service system, so the loaner may register a loan or return a book, without talking to an employee at the library.

2 IT-projektets formål og rammer

Vores IT-projekts formål og rammer er beskrevet ved FACTOR-begrebet[s. 39-41, kilde 2]. FACTOR er et værktøj man kan bruge, til at beskrive/definere et IT-projekt kort og præcist.

2.1 FACTOR

Functionality:

Systemet har funktioner, således at man kan registrere brugere, bøger, udlån og udsende rykkere.

Application Domain:

Systemet administreres af ansatte på biblioteket, men kan ved udlån og returnering anvendes af brugerne selv.

Conditions:

Systemet vil blive udviklet i samarbejde med, og efter kundens ønsker.

Technology:

Systemet vil blive udviklet på henholdsvis mac og linux computere. Det bliver udformet som en hjemmeside, som bliver kodet i HTML, CSS og PHP.

Objects:

Bøger og brugere er hovedobjekterne i dette system.

Responsibility:

Systemet skal have ansvaret for at registrere og overskueliggøre brugere, bogbeholdning og udlån.

Det kan læses ud af ovenstående FACTOR beskrivelse af systemet, at formålet med projektet er at udvikle et system, som kan holde styr på brugere, bogbeholdning og de udlån der måtte være. Et andet formål med systemet, er at aflaste de ansatte eller med andre ord, simplificere arbejdet for dem.

Rammerne for projektet er i bund og grund ret fleksible. Vores kunde har naturligvis en række minimumskrav, som de gerne vil have opfyldt. Når dette er gjort vil der være mulighed for videreudvikling eller at foretage ændringer på det allerede udviklede system, hvis der er behov for dette.

3 Kravspecifikation for IT-løsningen

3.1 Funktionelle- og ikke-funktionelle krav

Funktionelle krav:

Funktionelle krav fortæller noget om hvordan systemet fungerer i forbindelse med brugere af systemet og andre eksterne systemer.[s. 119, kilde 1].

- Bibliotekarere 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.

Ikke-funktionelle krav:

Ikke-funktionelle krav, er krav der ikke er direkte relateret til funktionaliteten af systemet. Det kan være mange ting, så som systemets ydelse, hvilket programmeringssprog det skal laves i, sikkerhed og lignende.[s. 120, 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, således at det også kan bruges som selvbetjening. Dette vil vi gøre ved at udføre brugertest, og kvalitativ forståelse ved interview med brugere.

3.2 Use case model over system-funktionaliteten

En use case model bruges til at overskueliggøre et system, og bliver i dette tilfælde brugt til at illustrere hvilke funktionaliteter de forskellige aktører har i systemet.[s. 42, kilde 1]

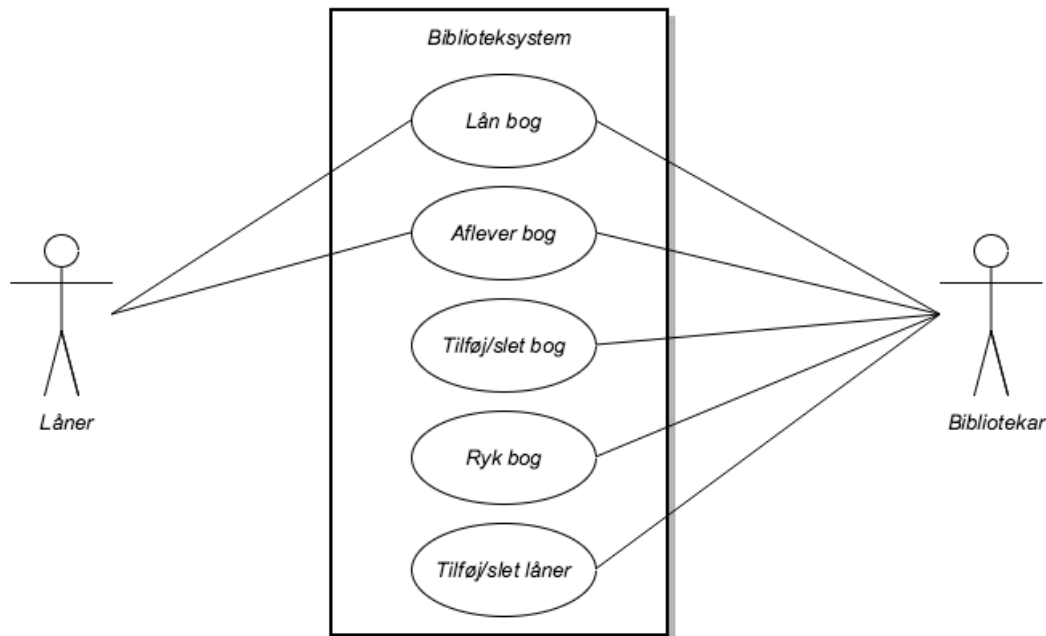


Fig. 3.1: Figuren herover er en illustration af hvilke muligheder, henholdsvis låner og bibliotekar har i systemet.

Figur 3.1 herover skal ikke forstås således, at de to aktører interagerer med hinanden, men blot at de har adgang til forskellige funktioner i samme system. I figuren kan man ligeledes se, at en almindelig låner har mulighed for at låne en bog, eller aflevere en bog igennem IT-systemet. Hvorimod en bibliotekar(eller anden ansat), har samme muligheder, men kan også tilføje og slette både bøger og lånere til databasen, og de har mulighed for at rykke en bog. Med rykke bog menes der en påmindelse, der sendes som mail til en låner, hvis vedkommende har overskredet returneringsdatoen. Hvordan systemet skal kende forskel på disse, bliver formegentlig ved brug af adgangskode eller anden form for identificering af de ansatte, der måtte bruge systemet.

3.3 Use cases

Use cases er en måde at simplificere en del af systemets funktionalitet set fra en aktørs/brugers perspektiv. Altså en beskrivelse af i skridt hvad aktøren skal foretage sig, i hvilken rækkefølge og hvad det resulterer i.[s. 42, kilde 1].

Herunder er specificeret tre use cases, som har en vigtig rolle i vores system. Det er henholdsvis lån af bog, udsending af rykkere, og tilføjelse af bøger til systemet.

3.3.1 Lån af bog

<i>Use case name</i>	<u>LånAfBogNy</u>
<i>Participating actor instances</i>	<u>bob: Låner</u>
<i>Flow of events</i>	<ol style="list-style-type: none"> 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.
<i>Entry condition:</i>	Brugeren skal være registreret
<i>Exit condition:</i>	Brugeren får en bekræftelse, og bogen er udlånt
<i>Quality requirements:</i>	Bogen kan ikke udlånes, før den er blevet returneret

Fig. 3.2: Figuren herover viser, de skridt der skal tages for at låne en bog.

I figur 3.2 herover kan 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 lånet accepteres ikke, hvis der tastes forkert bognummer. Derudover kræves det er brugeren er registreret i systemet, før denne kan foretage et lån.

3.3.2 Udsending af rykker

<i>Scenario name</i>	<u>RvkBogNy</u>
<i>Participating actor instances</i>	<u>alice: Bibliotekar</u>
<i>Flow of events</i>	<ol style="list-style-type: none"> 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.
<i>Entry condition:</i>	Brugeren skal være logget ind som bibliotekar
<i>Exit condition:</i>	Bibliotekaren får bekræftelse, og rykkerne er blevet sendt
<i>Quality requirements:</i>	Mailen forventes at være ved låneren indenfor 30 sekunder

Fig. 3.3: Figuren herover viser, de skridt bibliotekaren skal tage for at udsende rykkere.

Ved et kig på figur 3.3 kan det ses, hvorledes automatiseringen af henholdsvis tjek og udsending af rykkere fungerer. Det er en klar forbedring af den tidligere metode, hvor de ansatte selv manuelt skulle søge en række bøger igennem, for derefter at ringe lånerne op for at rykke dem. Det kræves naturligvis at det er en bibliotekar der benytter systemet, for at denne funktion er til rådighed.

3.3.3 Tilføjelse af bog

<i>Use case name</i>	<u>TilføjBog</u>
<i>Participating actor instances</i>	<u>alice: Bibliotekar</u>
<i>Flow of events</i>	<ol style="list-style-type: none"> 1. Alice klikker på funktionen tilføj bog 2. Alice indtaster oplysninger på bogen (nr. titel, forfatter, år osv.) 3. Alice bekræfter de indtastede oplysninger 4. Systemet fortæller Alice, om bogen blev succesfuldt registreret. 5. Alice kan hvis den blev registreret, sætte bogen på hylden, og fortsætte med den næste
<i>Entry condition:</i>	Brugeren skal være logget ind som bibliotekar
<i>Exit condition:</i>	Bibliotekaren får bekræftelse, og bogen er tilføjet
<i>Quality requirements:</i>	Bogen kan udlånes lige efter den er blevet tilføjet

Fig. 3.4: Figuren herover viser, de skridt bibliotekaren skal tage for at tilføje en bog til systemet.

På figur 3.4 herover kan man se hvorledes en bibliotekar tilføjer en bog til systemets database. Hvilke oplysninger på bogen der bør kunne indtastes, er endnu ikke fastlagt, så ovenstående er derfor blot et eksempel. Det kræves igen naturligvis at det er en bibliotekar der bruger systemet.

3.4 Problemområde

Et problemområde er det som systemet skal effektivisere, muliggøre eller løse på en eller anden måde. I vores tilfælde omhandler det altså de bøger, brugere, udlån og rykkere der skal administreres, registreres og lagres i databasen. Alle de dele som systemet er bygget op omkring.[s. 39, kilde 1] Et klassediagram kan bruges til at visualisere problemområdet, på en sådan måde at man kan forbinde de forskellige dele af systemet med hinanden, så man kan se hvordan det er bygget op.[s. 48, kilde 1]

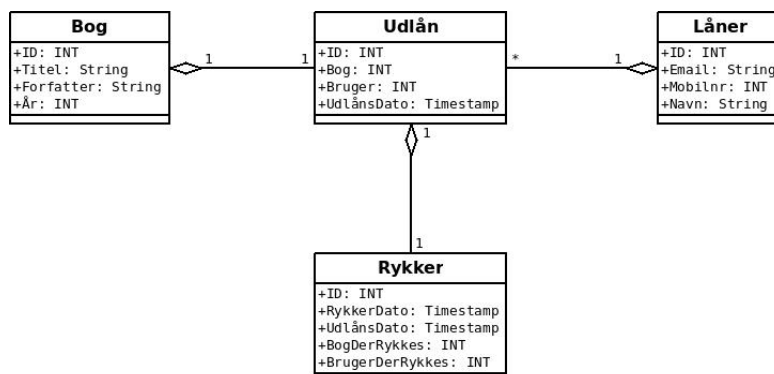


Fig. 3.5: Figuren herover er et klassediagram over de klasser, som vores problemområde, og vores system er bygget op omkring.

De givne attributter i objekterne i figur 3.5 herover, er ikke fastlagt, men vi forventer, at det bliver noget lignende det, der er anført. Figuren hænger sammen således, at et udlån består af en bog og en låner, og en rykker består af et udlån(hvori bog og låner er implicit). Alle multipliciteter er 1 til 1, udover at en låner godt kan have flere udlån.

3.5 BCE-Model

Boundary-, Control- og Entity-objekter er begreber, der bliver brugt til at beskrive sammenhængen mellem forskellige dele af et it-system. Ofte forbindelserne mellem brugergrænseflade og de bagvedliggende funktioner og data.

Boundary-objekter er de dele af systemet som en aktør direkte interagerer med. En brugergrænseflade bliver ofte forbundet med en boundary, da dette er hvad en bruger benytter sig af ved brug af et program.

Entity-objekter er de dele af systemet, som det er bygget op omkring. Det er ofte dele af systemet som gemmes på den ene eller anden måde. Det kan ofte forbindes med de entiteter der også normalt findes i en database.

Control-objekter er ikke en decideret del af systemet, men mere et abstrakt begreb for hvad der sker, fra aktøren udfører en handling i et boundary-objekt, til at der igen er noget aktøren skal foretage sig. Man kan forklare det ved, at en aktør eksempelvis klikker på et menupunkt(boundary), herefter sørger et control-objekt for, at der sker det der nu skal ske, som nok ville være at bringe aktøren hen til et andet boundary-objekt, som svarer til det der blev klikket på.[s. 171, kilde 1]

Boundary-objekter:

Brugergrænsefladen i sig selv, altså alle sider og undersider der er i systemet, og de formularer der skal udfyldes ved henholdsvis tilføjelse af bruger og bog, lån af bog osv.

Entity-objekter:

Entity-objekter vil være bøger, brugere, udlån og rykkere, i form af data, som

hele systemet afhænger af.

Control-objekter:

Control-objekterne er de abstrakte objekter, som sørger for at man bliver ledt de rigtige steder hen, ved valg i menuen. Og derudover også ved afsending af formularer, og derved også kontakt mellem websiden og databasen.

3.6 Sekvens diagrammer

Et sekvensdiagram er en nærmere beskrivelse af en use case, som illustrerer hvordan de forskellige dele af systemet arbejder sammen med hinanden og aktøren, og hvilken rækkefølge det foregår i. Det giver en fornemmelse af hvad en bruger af systemet aktivt skal gøre, og hvad systemet selv sørger for.

3.6.1 Lån af bog

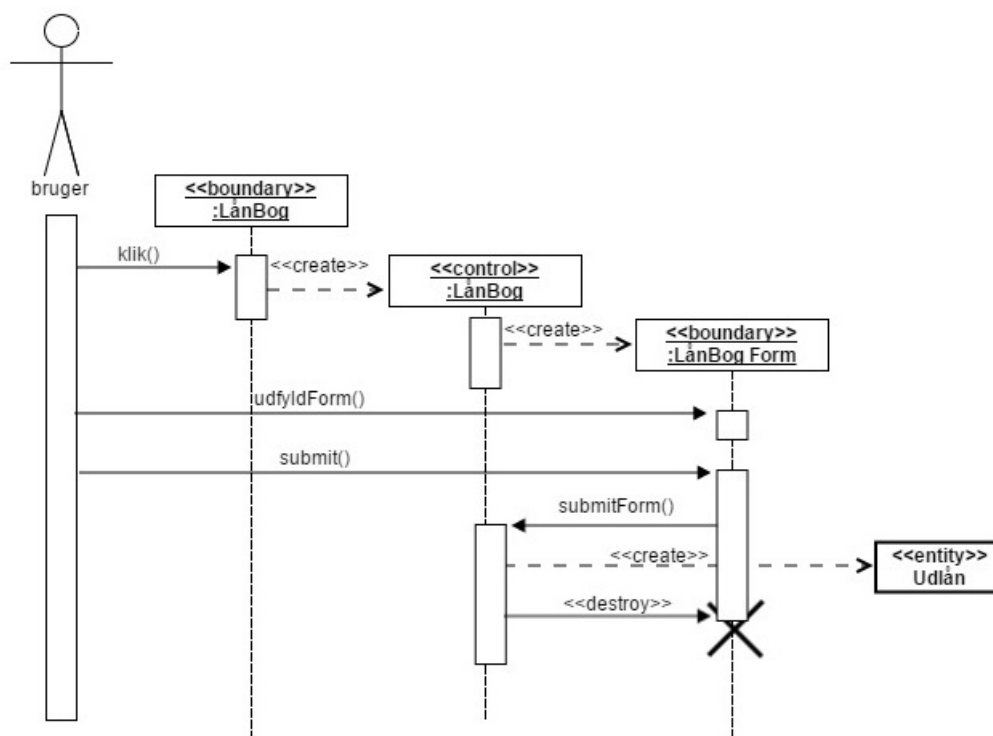


Fig. 3.6: Figuren herover et et sekvensdiagram over use casen der omhandler lån af bog.

I figur 3.6 herover, er sekvens processen delt op i tre instanser, menupunktet lån bog, en controller, og form der skal udfyldes. Brugeren skal indtaste sin mail(brugeren skal være registreret for at kunne låne!) og nummer på bogen han vil låne. Herefter klikkes på en knap for at fuldende lånet. Inputtet sendes her til databasen, og den returnerer om hvor vidt lånet blev registreret(hvis mail eller bog ikke findes, vil den fejle).

3.6.2 Tilføjelse af bog

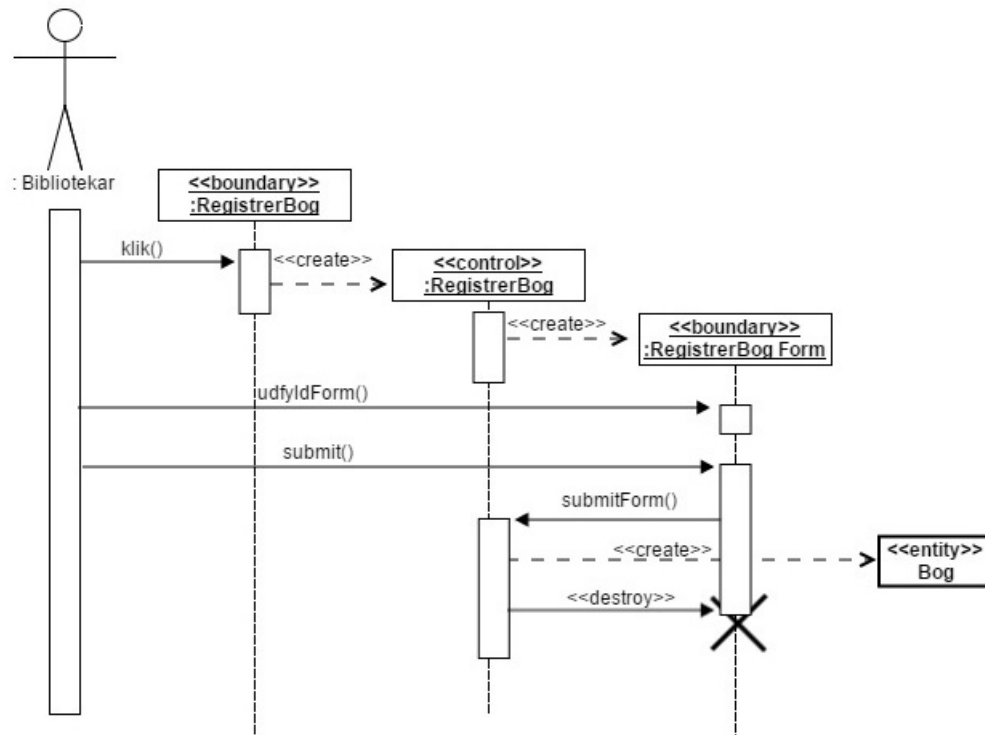


Fig. 3.7: Figuren herover et et sekvensdiagram over use casen der omhandler registrering af bog.

I denne figur 3.7, er sekvens processen ligeledes delt op i tre instanser, menupunktet registrer bog, en controller, og formen der skal udfyldes. Herefter skal bibliotekaren indtaste data på bogen(hvad dette indeholder er ikke fastlagt, men det vil være ting som forfatter, år, titel osv.). Herefter bliver han bedt om at bekræfte at han har indtastet de rigtige informationer. Klikkes der her på nej, skal han starte forfra/redigere i det indtastede. Klikkes der bekræft, sendes inputtet til databasen, og den returnerer om hvor vidt bogen blev registreret.

3.6.3 Udsending af rykker

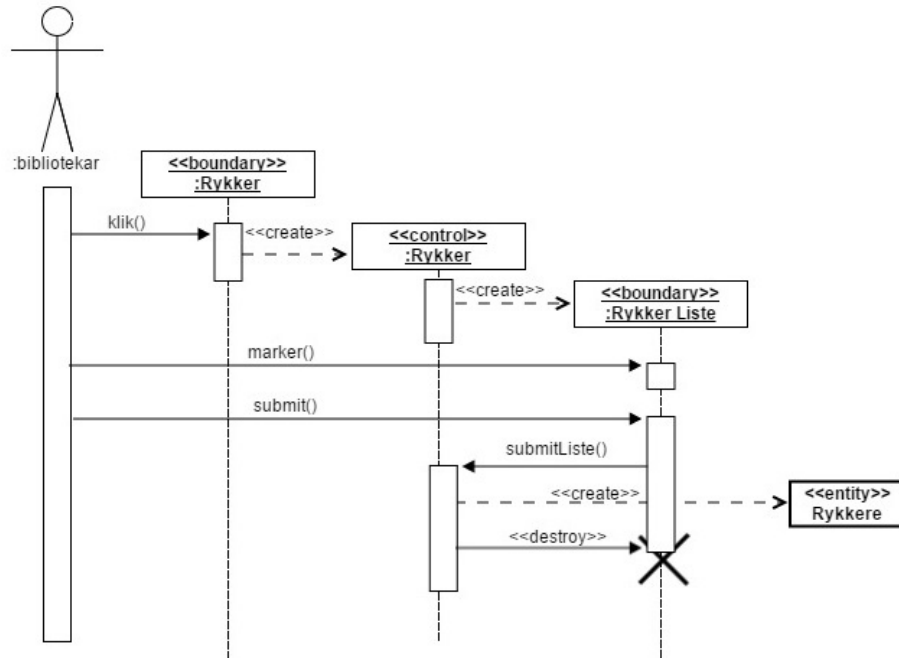


Fig. 3.8: Figuren herover et et sekvensdiagram over use casen der omhandler rykning af bog.

I figur 3.8 herover, er sekvens processen ligeledes delt op i tre instanser, menupunktet rykker, en controller, og en liste der indeholder de udlån der har overskredet afleveringsdatoen. Denne data er hentet fra databasen. Her klikker bibliotekaren på knappen send rykkere, og dataen sendes til mail-serveren. Her sendes mails afsted, og den returnerer om hvor vidt de blev sendt eller ej. Til slut registreres de udsendte rykkere i databasen.

4 Systemdesign sammenfatning

Systemet skal fungere som en hjemmeside, der er programmeret i HTML, CSS og PHP, samt en mySQL database hvori alt data skal lagres. Hjemmesiden er kun tilgængelig fra deres eget netværk, således at man formindsker risikoen ved at nogle misbruger det. Dette gøres ved at kontrollere IP-adressen.

Systemet skal desuden være konstrueret sådan at bibliotekets ansatte har et fælles login, således at de har flere muligheder end de ordinære brugere af systemet, såsom at tilføje eller fjerne brugere og bøger fra systemet. Da dette login kræves, vil siden altså også kunne fungere som selvbetjening for den enkelte låner, da de kun skal have mulighed for at låne og aflevere bøger.

5 Program- og systemtest

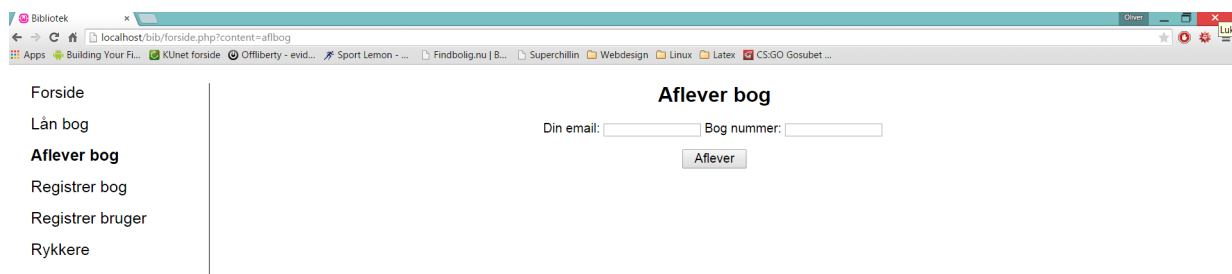
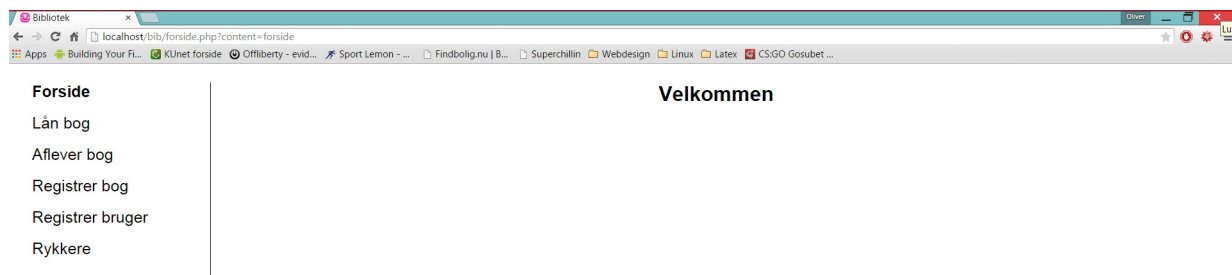
Vi har endnu ikke haft mulighed for at lave store hverken program- eller systemtest. Men det som skal testes når tiden kommer, er i bund og grund at vores funktioner de virker. Altså hvis man går ind og låner en bog i systemet, skal vi teste om den rent faktisk bliver registreret i databasen. Det gælder for alle funktioner der interagerer med databasen. Derudover skal vi sørge for at hvis en bog allerede er udlånt, skal den ikke kunne udlånes igen før den er blevet returneret. Disse database tests har vi allerede lavet nogle af, i takt med udviklingen af systemet.

En anden ting der også bør testes grundigt, er at hvis en bruger har overskredet returneringsdatoen for en bog, skal vedkommende ikke spammes med et overflødigt rykkere. Det skal kunne begrænses til at de modtager en hver 14. dag eller hver måned.

Til sidst vil vi teste brugervenligheden af systemet ved at lave tænke højt forsøg med de egentlige brugere af biblioteket.

6 Brugergrænseflade og interaktionsdesign

Herunder er et meget tidligt stadie af vores brugergrænseflade og vores generelle design. Vi lægger stor vægt på at systemet skal være så simpelt som muligt, og intuitivt hvorledes systemet fungerer. Der vil sandsynligvis senere blive skrevet nogle flere vejledende tekster til de forskellige funktioner, for at gøre det lettere for brugeren. Vi har også valgt at highlighte hvilken side på systemet, man er inde på, igen for at gøre systemet mere overskueligt og navigerbart. Som det ses på billederne er 'Forside' skrevet med tyk skrift når man er på forsiden, og 'Aflever bog' når man er der.



7 Projektsamarbejdet

7.1 I gruppen

Arbejdet internt i gruppen fungerer ganske fint, selvom vi alle har været tidspresset udover dette projektarbejde. Derfor har vi valgt ikke at holde fysiske møder, og i stedet benyttet os af kommunikations- og skærmdelingsværktøjet Skype. Som udgangspunkt arbejder vi i gruppen sammen om tingene, men hvis der udføres individuelle opgaver, vendes resultatet med alle i gruppen, og diskuterer eventuelle rettelser.

7.2 Kunden

Der er desværre opstået problemer i forhold til vores samarbejde med kunden, da de ikke længere svarer på hverken emails eller facebook beskeder. Derfor kan det ikke undgås at store dele af vores system vil blive bygget på antagelser, på hvad kunden måtte forlange at systemet skal kunne. Derudover har vores instruktør Markus, tilbudt at han kan agere kunde i situationer hvor vi har særligt behov for det, og eventuelt som tester af systemet.

8 Bibliografi

1. Bruegge, Bernd og Dutoit, Allen H. 2014. "Object-Oriented Software Engineering Using UML, Patterns, and Java third edition"
2. Mathiassen L., Munk-Madsen A., Nielsen P. A., Stage J. 2000. "Object-Oriented Analysis and Design" (Uddraget fra ugeseddel 6)
3. Gould, J. D. og Lewis, C. 1985. "Designing for usability: key principles and what designers think". s. 300-311
4. Parnas, D. L. og Clements, P. C. 1986. "A rational design process: How and why to fake it" s. 251-257
5. Sutherland, Jeff. 2010. "SCRUM Handbook"

9 Litteratur Review

Herunder er udført to litteraturreviews af to artikler. Artiklerne fremgår også af bibliografien.

9.1 A rational design process: How and why to fake it

Denne artikel af David Lorge Parnas og Paul C. Clements omhandler udviklingen af et it-system, og problematikken i at lave en god dokumentation af systemet. De lægger vægt på, at det stort set er en umulighed at planlægge et it-system fra bunden af, og derefter programmere det, således at det virker, som det skal første gang. Dette er også noget OOSE[kilde 1] kommer ind på, i deres beskrivelse af software udvikling, de beskriver nemlig, at: "*Assumptions that developers make about a system change constantly.*"[kilde 1, s. 7]. Det bekræfter, at det er stort set umuligt at udvikle et system på denne måde. I artiklen, beskriver de meget præcist hvorfor, at man aldrig vil kunne opnå denne rationelle design process, hvoraf kompleksitet og menneskelige fejl er nogle af grundene til det.

Det centrale som denne artikel omhandler, er at man bør "*fake a rational design process*"[kilde 4, s. 2]. De forklarer, at det gøres ved at gå tilbage til dokumentationen og ændre i den, hver gang man finder ud af en ny løsning, således at når man læser dokumentationen, virker det som om, at systemet er udviklet perfekt fra bunden af. De forklarer at selv matematikere benytter sig af denne metode, for at gøre deres beviser pænere og simplere: "*Mathematicians diligently polish their proofs, usually presenting a proof very different from the first one that they discovered.*"[kilde 4, s. 6]

Det er meget svært, hvis ikke umuligt at designe et program helt fra bunden og opbygge det hele uden noget fejl, eller skal løses på en anden måde. De sammenligner med måden andre videnskaber arbejder på: "*Ideally, we would like to derive our programs from a statement of requirements in the same sense that theorems are derived from axioms in a published proof.*"[kilde 4, s. 1]. Forskellen på software engineering og andre videnskaber, er dog bare at et it-system, ofte er så stort og komplekst, at man ikke kan planlægge det hele perfekt fra starten.

Artiklen er i bund og grund et godt værktøj til at lave denne 'falske' dokumentation, da den meget præcist og uddybet, fortæller hvorfor, det er nødvendigt og hvordan man skal gøre det i punktform.

Vores umiddelbare tanke efter at have læst denne artikel er, at det er en let løsning at rette i dokumentationen, hvis forløbet ikke lige går som planlagt. Men når man selv har forsøgt at sidde med et projekt, eller endda bare et simpelt program, så er man klar over hvor svært, det er at planlægge det til punkt og prikke i første omgang. Det er meget mere komplekst end man nødvendigvis lige forestiller sig, og man bliver derfor nødt til jævnligt at overveje, om det man har udrettet bør gøres på en anden måde eller lignende. Derfor opstår der også tit uventede forhindringer og problemer. Man bliver derfor ofte nødt til at genoverveje tidligere beslutninger i takt med, at man får en bedre forståelse for det system man udvikler, hvilket sandsynligvis har været grunden til, at man har udviklet agile arbejdsmetoder som SCRUM [kilde 5].

SCRUM er netop baseret på, at man har en sprint (tidsperiode) på mellem 1-4 uger, hvor udviklerteamet efter hver sprint, mødes og drøfter hvad der er lavet siden sidst, hvad der bør laves til næste gang, og hvad der måtte være opstået af problemer.[s. 10, kilde 5] Et møde efter en sådan sprint, vil også være et oplagt tidspunkt til at få samlet op på, hvad der er nødvendigt at få rettet i dokumentationen. På den måde vil udarbejdelsen af dokumentationen også foregå iterativt, således undgår man uoverskueligheden ved at skulle rette dokumentationen til sidst i projektet.

9.2 Designing for usability: key principles and what designers think

Artiklen "Designing for Usability: Key Principles and What Designers Think" af John D. Gould og Clayton Lewis omhandler det de mener er de tre essentielle dele af en god og succesfuld tilgang til brugerflade design. De nævner de tre dele, eller skridt, tidligt fokus på bruger og opgave, emperiske målinger og iterativt design. Disse tre dele af designprocessen mener forfatterne også er skridt som folk, når de bliver præsenteret med ideen, nikker anerkendende til men ikke nødvendigvis gør brug af eller forstår til fulde, "*Often the reaction is that they are obvious. Nevertheless, they are not usually employed in system design...*" [kilde 3, side 1]. En undersøgelse hvori de gav folk point afhængig af hvorvidt de kunne nævne og forklare de 3 principper fungerer som en indikator både overfor forfatterne såvel som læserne, og danner grobund for en videre forklaring af principperne.

Det første af principperne, det tidlige fokus på brugerne og opgaverne involveret, forklares som en helt essentiel del af designprocessen. Der bliver lagt stort fokus på forståelse af opgaven, men i særdeleshed også et generelt billede af de formodede brugere af systemet. Der bliver også lagt fokus på at være i direkte kontakt med brugeren og ikke igennem en mellemmand eller et inaktivt medie, såsom at læse sig frem til data om brugergrupper.

Det andet princip, emperiske målinger, omhandler indsamling af data fra brugeres oplevelser med prototyper af systemet. Målingerne og tests skal altså fungerer som en test af systemet fremfor en selvbekræftende spiral som kun fungerer som et klap på skulderen til udvikleren, "*What is required is a usability test, not a selling job...*" [kilde 3, side 3]. Disse test og data indsamlet kan også give en indsigt som udvikleren med sine "udvikler-skyklapper" ikke kan overskue.

Den iterative process er kernen af designfilosofien. Evnen til at kunne komme tilbage med den nye data indsamlet fra de imperiske målinger og granske ens system fra et friskt synspunkt er helt centralt. I artiklen bliver der også gjort op med ideen om at et system kan være rigtigt designet fra starten af, som også nævnt i artiklen "A rational design process: How and why to fake it". Denne iterative proces åbner op for muligheden om, og gør det helt naturligt at ens udgangspunkt med designet ikke nødvendigvis er det rigtige. Den iterative proces kan dog kun ske hvis ens system er fleksibelt nok til at tillade nem redigering og videreudvikling, "*This means that the flexibility of the implementation approach has to extend as far into the system as possible...*" [kilde 3, side 9].

De nævner også mulige svagheder ved deres designproces, såsom øget omkostninger og tid, men retfærdiggør processen ved at argumentere for at andre, mere "direkte" processer er nødsaget til at gøre det samme for at opnå acceptable og brugbare systemer, "*User testing will happen anyway: If it is not done in the developer's lab, it will be done in the customer's office...*" [kilde 3, side 7]. Tests, indsamling af data og videreudvikling af prototyper og systemer ligger altså så centralt til den 3-delte designproces at tids- og ressourcspildet bliver minimeret.

Artiklen ridser en agil designproces op, muligheden for fleksibilitet og tid til at sætte spørgsmålstegn ved allerede eksisterende systemkrav ligger centralt i tankegangen bag. Iterationsdelen af processen må altså forstås som det der driver udvikling af bedre og bedre systemer.





































10 Bilag 1: Versionsstyring




























Link til github repository:

<https://github.com/oliver3660/ProjDat2015.git>

Vi har herunder indkluderet vores nuværende commit-log. Vi har dog ikke inkluderet commits helt fra starten af, da det ikke har haft indflydelse på projektet. Det var blot nogle tests til at lære github at kende.

Der har kun været et commit der har haft noget med selve programkoden at gøre. Dette commit blev foretaget d. 20 april, og indeholder html, css og en smule php. Det indeholde til sammen en fungerende menu, hvori man kan dirigere mellem de forskellige sider.

Commits on Apr 20, 2015
<div> kode1 oliver3660 authored 2 hours ago</div> <div> e93b9f8 </div>
Commits on Apr 19, 2015
<div> Rapport 2 - Opdatering oliver3660 authored 20 hours ago</div> <div> 80f95d1 </div>
Commits on Apr 15, 2015
<div> opdateret rapport 2 oliver3660 authored 5 days ago</div> <div> 24ac76d </div>
<div> tilføjer kodemappe oliver3660 authored 5 days ago</div> <div> 43ae3c5 </div>
<div> flytter readme oliver3660 authored 5 days ago</div> <div> e86fa9b </div>
<div> flytter Readme oliver3660 authored 5 days ago</div> <div> 8505682 </div>
<div> tilføjer mappe oliver3660 authored 5 days ago</div> <div> 5844c2f </div>
<div> Sletter alt for at rydde op oliver3660 authored 5 days ago</div> <div> 8df0455 </div>
<div> Slettet fil kopi oliver3660 authored 5 days ago</div> <div> be31901 </div>
Commits on Apr 14, 2015
<div> Rapport 2, start oliver3660 authored 6 days ago</div> <div> a70090c </div>
Commits on Mar 30, 2015
<div> Merge branch 'master' of https://github.com/oliver3660/ProjDat2015 oliver3660 authored 21 days ago</div> <div> a8494eb </div>
<div> done rapport oliver3660 authored 21 days ago</div> <div> a433e09 </div>

Commits on Mar 26, 2015	<div>  krav til delrapport 2 kiksekage authored 25 days ago </div> <div>  6f01597 </div> <div>  </div>
	<div>  jeg addede lige rettelser kiksekage authored 25 days ago </div> <div>  eeda4b2 </div> <div>  </div>
Commits on Mar 24, 2015	<div>  review1 oliver3660 authored 27 days ago </div> <div>  983d0df </div> <div>  </div>
Commits on Mar 22, 2015	<div>  rapport oliver3660 authored 29 days ago </div> <div>  db2743f </div> <div>  </div>
Commits on Mar 19, 2015	<div>  rapport oliver3660 authored on 19 Mar </div> <div>  b603ae6 </div> <div>  </div>
Commits on Mar 16, 2015	<div>  rapport oliver3660 authored on 16 Mar </div> <div>  8b19022 </div> <div>  </div>
Commits on Mar 13, 2015	<div>  problemformulering oliver3660 authored on 13 Mar </div> <div>  60bb100 </div> <div>  </div>
Commits on Mar 12, 2015	<div>  nyt billede kiksekage authored on 12 Mar </div> <div>  d850f85 </div> <div>  </div>
Commits on Mar 10, 2015	<div>  Systembeskrivelse oliver3660 authored on 10 Mar </div> <div>  9df1d3d </div> <div>  </div>

11 Bilag 2: Changelog for projektrapporten

14-04-2015 OSK Rapport påbegyndt
 14-04-2015 OSK Punkt 2 og 3 påbegyndt
 15-04-2015 OSK Review af "A rational design process:..." færdiggjort
 15-04-2015 Alle Punkt 7 påbegyndt
 15-04-2015 Alle Punkt 1 påbegyndt
 19-04-2015 Alle Punkt 3 fortsat
 19-04-2015 TNS Review af "Designing for usability:..."
 20-04-2015 Alle Punkt 1, 2, 3, 7 færdiggjort
 04-05-2015 Alle Rapport rettet til genaflevering

12 Bilag 3: Timeline

Tidslinjen herunder daterer og beskriver milepæle og/eller centrale igangsætninger og afslutninger af processer.

09-03-2015:

Møde med product owner Simon Shine, for at få de overordnede krav, og ønsker til IT-systemet.

13-03-2015:

Arbejde med første delrapport begyndes.

23-03-2015:

Første delrapport færdig.

14-04-2015:

Arbejde med anden delrapport begyndes.

20-04-2015:

Kodning af systemet begyndes.

04-05-2015:

Prototype 1 færdig.