ProjDat2015 - Bibliotek udlånssystem

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

Instruktor: Markus Wittorf

April 22, 2015

Indholdsfortegnelse

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 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.

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 funktioner[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 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 aktørere har i systemet.

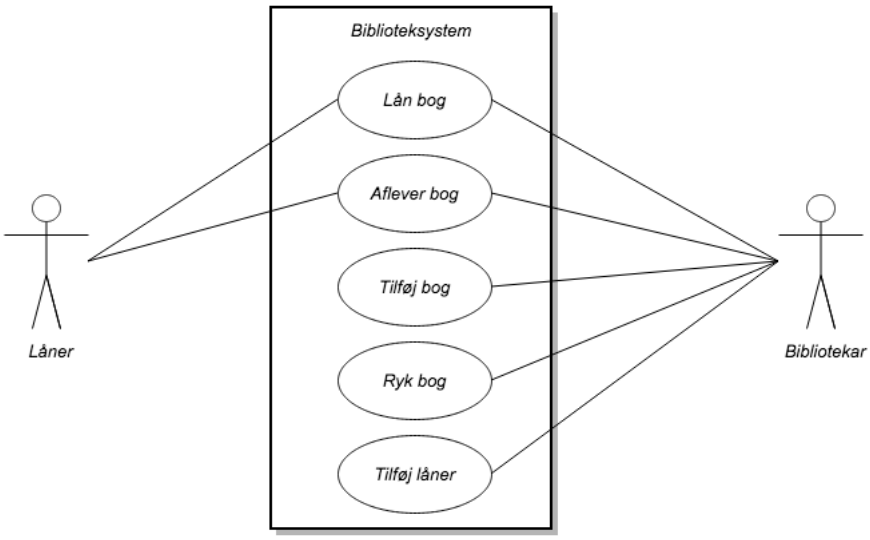

Fig. 3.1: Figuren herover er en illustration af hvilke muligheder, henholdsvis låner og bibliotekar har i systemet. Det skal ikke forstås således at de to aktører interagerer med hinanden, men blot at de har adgang til forskellige funktioner i samme system.

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 både bøger og lånere til databasen, og de har mulighed for at rykke en bog. 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

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

3.3.1 Lån af bog

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. 3.2: Figuren herover viser, 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.

3.3.2 Udsending af rykker

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. 3.3: Figuren herover viser 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.

3.3.3 Tilføjelse af bog

Use case name	TilføjBog
Participating actor instances	alice: Bibliotekar
Flow of events	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

Fig. 3.4: Figuren herover viser 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.

3.4 Problemområde

Systemets problemområde, som består af de bøger, brugere, udlån og rykkere der skal administreres, registreres og lagres i databasen, og som systemet er bygget op omkring.

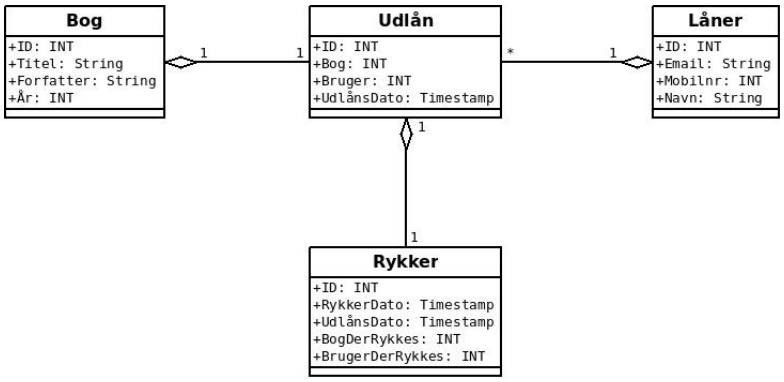

Fig. 3.5: Figuren herover er et klassediagram over de klasser, som vores problemområde, og vores system er bygget op omkring. De givne attributter er ikke fastlagt, men vi forventer, at det bliver noget lignende det, der er anført. Figuren hænger sammen således, at et udlån består af en bog og en låner, 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

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

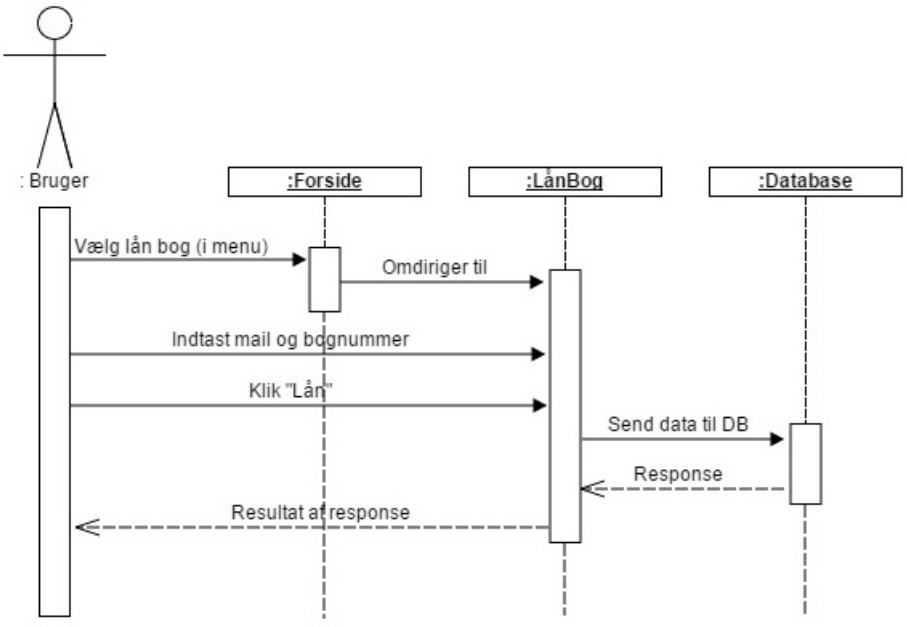

Fig. 3.6: Figuren herover et et sekvensdiagram over use casen der omhandler lån af bog. Processen er delt op i tre instanser, forside, siden der indeholder lån bog funktionen, og databasen. Det skal forstås således, at brugeren vælger lån bog i menuen på forsiden og bliver dirigeret til lån bog siden. Herefter skal brugeren 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

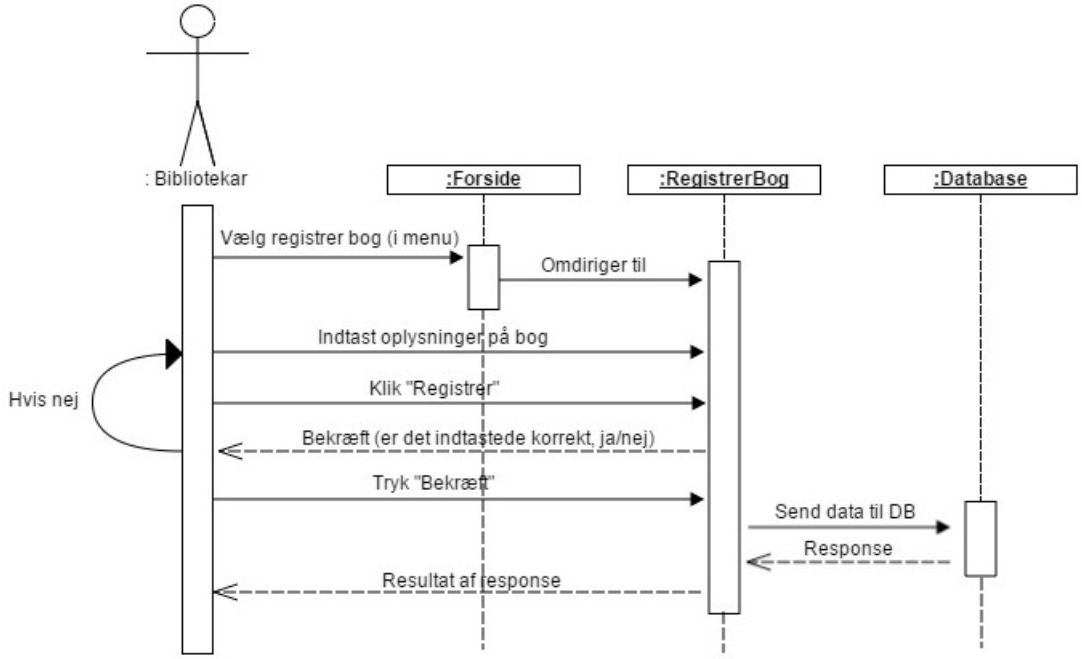

Fig. 3.7: Figuren herover et et sekvensdiagram over use casen der omhandler registrering af bog. Processen er delt op i tre instanser, forside, siden der indeholder registrer bog funktionen, og databasen. Det skal forstås således, at bibliotekaren vælger registrer bog i menuen på forsiden og bliver dirigeret til registrer bog siden. 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

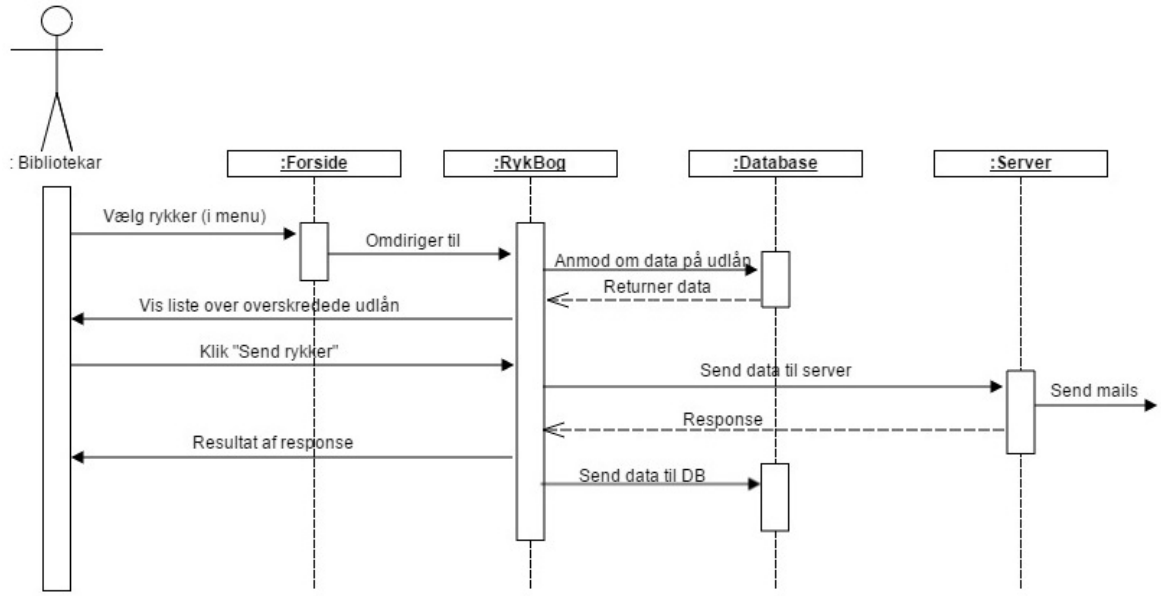

Fig. 3.8: Figuren herover et et sekvensdiagram over use casen der omhandler rykning af bog. Processen er delt op i fire instanser, forside, siden der indeholder ryk bog funktionen, databasen og mail-serveren. Det skal forstås således, at bibliotekaren vælger ryk bog i menuen på forsiden og bliver dirigeret til ryk bog siden. Her vil der være en liste over de udlån, som ikke er blevet returneret i tide. 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

5 Program- og systemtest

6 Brugergrænseflade og interaktionsdesign

7 Projektsamarbejdet

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.

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 fejler, 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. Man bliver ofte nødt til at genoverveje tidligere beslutninger i takt med at man får en bedre forståelse for systemet, hvilket sandsynligvis har været grunden til, at man har udviklet agile arbejdsmetoder som SCRUM.

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.

9.2 Designing for usability: key principles and what designers think

10 Bilag 1: Versionsstyring

Link til github repository:
https://github.com/oliver3660/ProjDat2015.git

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:..."

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.