Bachelor

Bachelor

Bachelor in Software Development,
IT-University of Copenhagen

Jakob Melnyk, jmel@itu.dk
Frederik Lysgaard, frly@itu.dk

May 22st, 2012

Indholdsfortegnelse

1 Indledning

Det har længe været på tale at få lavet et integreret system til IT-Universitetets booking af lokaler for interne og eksterne brugere. Den nuværende metode er besværlig og uoverskuelig, da planlægningen og håndteringen af bookinger foregår i flere forskellige systemer med Facility Management (FM) som centralt bindepunkt[KravSpec, Kap. A].
Det integrerede systems primære formål er at lette arbejdsbyrden for FM samt minimere fejl.

Denne rapport er udarbejdet i forbindelse med et projekt løbende fra 15. februar til 22. maj, 2013. Målet med projektet var at udvikle en løsning til IT-Universitetets ønske om et bookingsystem.

Projektet resulterede i en første udgave, som opfylder nogle af kravene fra en kravspefikation (beskrevet i 1.1). Den første udgave af løsningen består af det grafiske design og implementationen.
Vi evaluerer løsningen ud fra antal problemer i usability tests samt hvor stor en del af kravspecifikationen, løsningen opfylder. Desuden har vi lavet en kort undersøgelse af, hvilke alternativer IT-Universitetet har i forbindelse med anskaffelsen af et booking-system.
Vores konklusion består af en evalueringen af projektet samt en beskrivelse af de erfaringer, vi har gjort igennem processen.

1.1 Kravspecifikationen

Systemet er baseret på en kravspecifikation udarbejdet på kurset "Anskaffelse og kravspecifikation" i efteråret 2012. Kravspecifikationen er udarbejdet af Stig Larsen (stla@itu.dk), Miki Ipsen (mikc@itu.dk), Garwun Jeffrey Lai (gjel@itu.dk) og Merete Larsen (mnol@itu.dk)[1].

1.2 Versionsstyring, database og service info

Vi har anvendt Git (på github.com) til versionsstyring. Derudover har vi brugt en virtuel server opsat af ITU til at hoste vores service og database.

GitHub repository https://github.com/esfdk/itubs

Service URL http://rentit.itu.dk/RentIt12/Services/Service.svc

Database RentIt12 på http://rentit.itu.dk (brugernavn: RentIt12Db – adgangskode: Zaq12wsx)

Test Administrator Brugernavn="Admin@BookIt.dk" Adganskode="zAq12wSx"

Test Bruger Brugernavn="test@bookit.dk" Adgangskode="qwerty123"

[1] Vi anskaffede kravspecifikationen hos Jeffrey Lai.

2 Baggrund For Projektet

2.1 Booking på ITU

IT-Universitet er et stort universitet med mange ressourcer, som de studerende og ansatte kan benytte sig af, der er derfor behov for et system der kan understøtte booking af lokaler, udstyr og forplejning.

I øjeblikket er booking af ressourcer på ITU et uoverskueligt og problemfyldt område. Planlægningen af de forskellige ressourcer sker i dag primært manuelt i mange forskellige systemer hvor Facility Management (FM) fungerer som centralt bindepunkt for alt vedrørende lokale administration. Dette er ikke holdbart, da systemet ikke giver nok gennemskuelighed i forhold til at bruge ressourcerne i systemet bedre. Da meget af arbejdet sker manuelt skal personalet opretter alle bookinger, meget af deres tid bruges derfor på arbejdesopgaver som brugeren skulle udføre selv.

Det udfordrende ved at lave et automatiseret booking system til IT-Universitet er, at det skal understøtte bookinger af de forskellige ressourcer. Det skal derfor understøtte mange arbejdsområder, og der skal tages hensyn til mange forskellige aktører, eksemepelvis studerende, ansatte og kantinemedarbejder.

For at have overblik over, hvad der skulle understøttes og hvad der var vigtigst fulgte vi en kravspecifikation som blev udarbejdet i kurset "Anskaffelse og kravspecifikation" på ITU i efteråret 2012, skrevet af Stig Larsen (stla@itu.dk), Miki Ipsen (mikc@itu.dk), Garwun Jeffrey Lai (gjel@itu.dk) og Merete Larsen (mnol@itu.dk). Kravspecifikationen er lavet med fokus specifikt på ITU og hvad et bookingsystem til ITU kræver.

2.2 Kravspecifikationen

Som nævnt er kravspecifikationen specifikt udviklet med henblik på hvad der kræves af et bookingsystem på IT-Universitet. Den giver derfor et overblik og hvordan bookingen af ressourcer fungerer i øjeblikket. Figur 2.1 viser den nuværende situation.

For at forbedre den nuværende løsning foreslår kravspecifikationen, at der laves et systemet som alle aktører arbejder op imod. Figur 2.2

Til udviklingen af det nye system definerer kravspecifikationen 13 arbejdsopgaver som udgør den funktionalitet der kræves af det nye system. Hvis alle 13 arbejdsopgaver bliver understøtte vil det nye system løse problemerne med gennemskueligheden i forhold til ressourcerne, derudover vil brugeren kunne lave mange af arbejdsopgaverne, og det vil derfor også løse problemet med at de ansatte brugere for meget tid på at oprette bookinger.

2.3 Problemformulering

Vi vil til projektet udvikle et booking system til IT-Universitet som er basseret på kravspecifikationens 13 arbejdsopgaver. Vi har i den sammenhæng fire hovedpunkter.

-
-
-
-

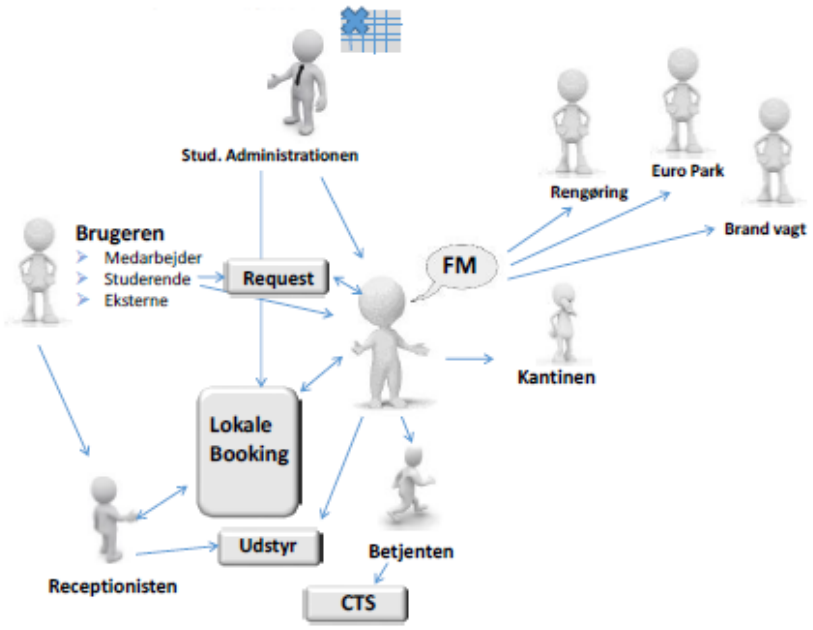

Figure 2.1: Nuværende system flow for booking systemet på ITU

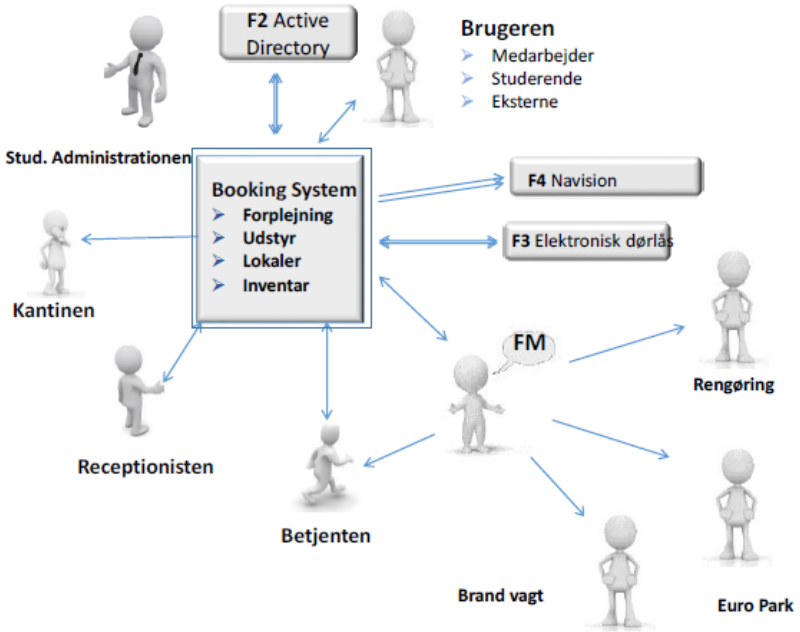

Figure 2.2: Det ønskede system flow for booking systemet på ITU

Vi vil udvikle et booking system til IT-Universitet som er basseret på kravspecifikationens 13 arbejdsopgaver. Vi vil udvikle en webservice og en browser klient som benytter sig af webservicen.

for projektet ligger vores problemformulering. Vi vil udvikle et booking system til IT-universitetet baseret på kravspecifikationen "ITU booking system". Vi vil til det formål udvikle en webservice der undersøtter cross-platform, vi vil samtidig udvikle en klient som benytter sig af webservicen. Vi vil derudover vurdere om hvorvidt vores løsning opfylder kravspecifikationen og i hvilken grad, desuden vi vil undersøge om der allerede findes systemeret på markedet som kan det samme eller eller mere, som vores produkt.

3 Design Af Brugergrænsefladen

Dette kapitel beskriver det generelle design af brugergrænsefladen samt de beslutninger, som ligger bag. Alle vores papermockups og endelige skærmbilleder kan findes i appendix, afsnit B på side 28.

3.1 Generelle mål

Vi har valgt at designe vores brugergrænseflade ud fra reglerne om design af virtuelle vinduer[SL, s. 169] samt Ease Of Use principperne[SL, s. 9]. I forbindelse med dette valg har vi sat følgende mål for designet:

- Strømlinet brugergrænseflade
- Få forskellige skærmbilleder
- Overblik
- Effektivt

3.1.1 Strømlinet brugergrænseflade

Vi har valgt at designe skærmbillederne med samme grundstruktur. Denne lighed bør gøre det intuitivt at gå fra et skærmbillede til et andet i forbindelse med udførsel af opgaver. Desuden følger det reglen om få vindueskabeloner.

3.1.2 Kort vej fra en opgave til en anden

Brugergrænsefladen skal gøre det hurtigt og nemt for brugeren at komme fra en opgave til en anden. Dette skal gøres ved at have få skærmbilleder involvereret i en enkelt task (reglen om få vinduer per opgave).

3.1.3 Overblik

Brugeren skal have mulighed for nemt at danne sig overblik over bookinger, udstyr og forplejning[1]. Derfor skal vi have seperate skærmbilleder, som giver overblik over hver type.

3.1.4 Effektivt

Det skal være effektivt at udføre opgaver for brugere, som anvender systemet ofte.

3.2 Brugergrænsefladens udvikling og udseende

Vores skærmbilleder er opdelt i tre typer: Gitter, Almindelig og Pop-ups.
Gitterskærmbillederne bruger vi til booking af lokale, forplejning og udstyr samt administration af bookinger, lokaleinventar og udstyr.
Almindelige vinduer anvender vi i forbindelse med login eller, hvis man skal ændre noget på et stykke udstyr/inventar.
Pop-up skærmbilleder er generelt advarsler eller fejlbeskeder.

[1] Reglen om den nødvendige oversigt af data.

Vores første mockup af skærmbilledet til booking af lokaler (figur 3.1) havde et gitter, hvor hver række var et lokale og tiderne var kolonner. Man skulle klikke i et felt for at vise, at man ønskede at booke på et bestemt tidspunkt. Når man havde valgt de tider og lokaler, man gerne ville booke, skulle man trykke på en "Book" knap.

Vi valgte at beholde gitterstrukturen. Vi tilføjede dog checkboxe til gitteret for at imødekomme et problem, som vi observerede i forbindelse med vores usability test[2]. Figur 3.2 viser vores endelige version.

Figure 3.1: Første udgave af gitter layoutet

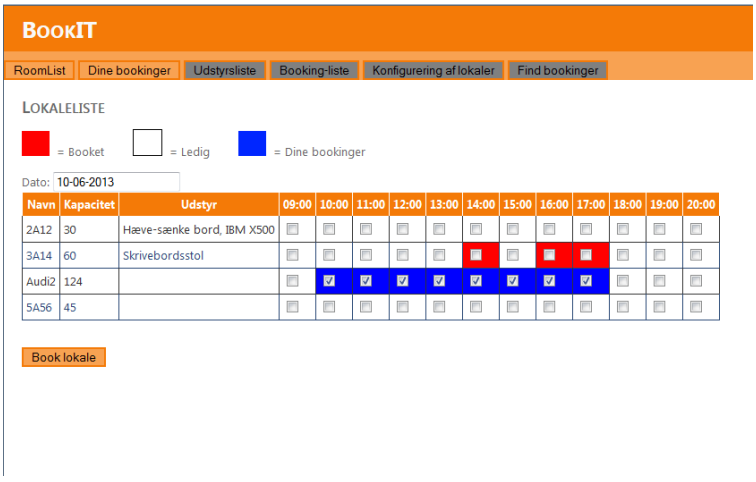

Figure 3.2: Skærmbilledet til booking af lokaler

Når man har valgt lokale og tidspunkt, bliver man sendt videre til en oversigt over ens egne bookinger (se figur 3.3).

[2]Se afsnit 4.2 på side 14

Figure 3.3: Skærmbilledet til visning af bookinger

Figur 3.4 viser det endelige skærmbillede til valg af forplejning. Vi fokuserede på, at skærmbilledet skulle minde om 3.2, så vi holdte os til reglen om få vindueskabeloner. De samme designbeslutninger gælder for skærmbilleder til tilføjelse af udstyr til en booking (se appendix).

Figure 3.4: Skærmbilledet til booking af forplejning

3.2.1 Bookingliste

Figur 3.6 viser vores papermockup til godkendelse af bookinger. I papermockup havde vi godkend/afvis-knapperne inde i selve gitteret. Dette besluttede vi dog kunne blive forvirrende, så den endelige version af skærmbilledet (figur 3.5) har knapperne ved siden af gitteret. Da ingen af de andre skærmbilleder har egentlige knapper inde i gitteret, opfylder i større grad reglen om få vindueskabeloner.

Figure 3.5: Skærmbilledet af recepsionistens liste af bruger bookinger

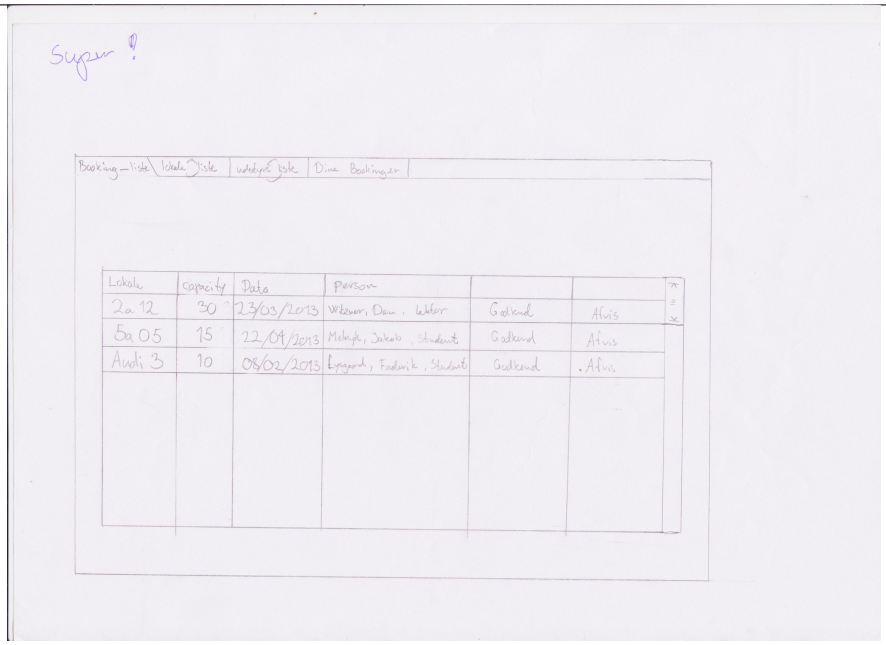

Figure 3.6: Papermockup af recepsionistens liste af bruger bookinger

3.2.2 Udstyrsliste

Figur 3.7 viser det endelige skærmbillede af listen over udstyr. Skærmbilledet indeholder både overblikket over alt det udstyr, som er tilrådighed, og muligheden for at registrere nyt udstyr til systemet.
Vi delte de to funktionaliteter op, så det var tydeligt, at det er seperate funktionaliteter.

Den eneste ændring siden vores papermockup (se appendix) er tilføjelsen af en checkbox, hvor man kan vælge et nyt stykke udstyr skal være til udlån. Dette betyder, at vi sparer et skærmbillede væk. Da udstyr og inventar er stort set ens i systemet, kan vi afgøre, om det nye element er inventar eller udstyr gennem denne checkbox.

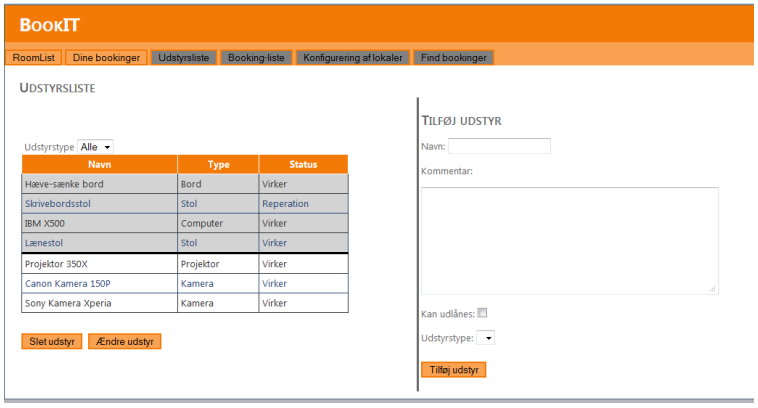

Figure 3.7: Skærmbilledet af recepsionistens liste over udstyr i ITUs system.

3.2.3 Ændring af lokale

Skærmbilledet til ændring af lokale er designet således, at man kan ændre navn og kapacitet på lokalet samtidig med, at man kan få et overblik over udstyret, som er tilføjet til lokalet.

Papermockup af dette skærmbillede (figur 3.9) var delt op i tre dele: et til at skifte navn/kapacitet, et til at tilføje inventar og et til at fjerne inventar.
Dette var meget uoverskueligt og mindede ikke om resten af vores design. Vi valgte derfor at bruge gitterløsningen til udstyret. Den øverste del af gitteret viser det inventar, som er tilføjet lokalet. Resten af gitteret viser det inventar, som ikke er tildelt noget lokale. Teksten på "Tilføj Udstyr"-knappen ændrer sig til "Fjern Udstyr", hvis man trykker på en linje i den øverste del af gitteret.

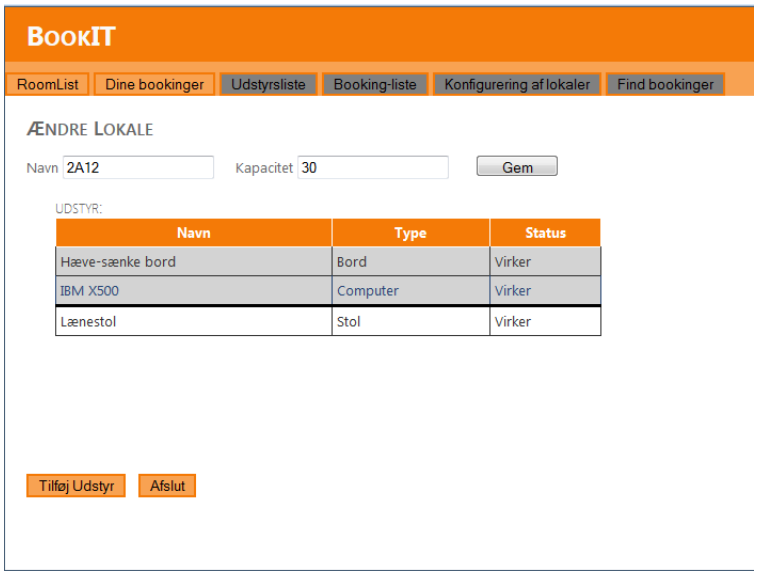

Figure 3.8: Skærmbilledet til ændring af lokale.

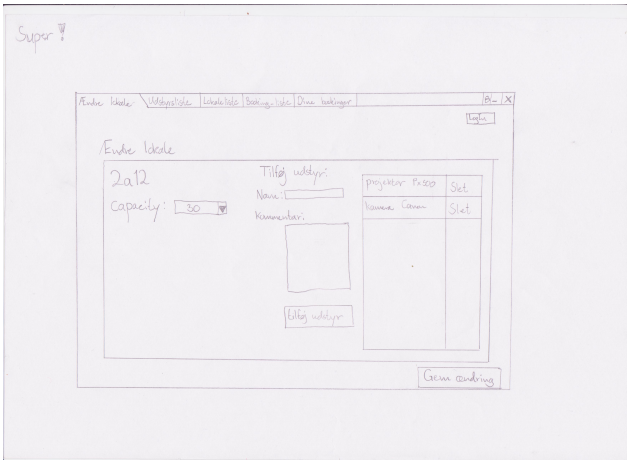

Figure 3.9: Papermockup til ændring af lokale.

3.3 Fremtidig design

Dette afsnit er en liste af vores forslag til ændringer i brugergrænsefladen. Det er ikke en prioteret liste, da alle ændringerne er quality of life forbedringer.

- Brugerens egne bookinger skal være øverst på listen over bookinger
- Sammenlægning af "Booking-Liste" og "Find Bookinger"
- Sammenlægning af "Konfigurering af Lokaler" og "Ændre Lokale"
- Informering om ændringer i status på en brugers booking
- Understøt godkendelse af forplejning og udstyrsvalg
- Mulighed for at skifte mellem sprog (dansk/engelsk)
- Tilføjelse af nye typer forplejning og nye udstyrstyper
- Mulighed for ændre priser på forplejning
- Være i stand til at søge på sal i lokalelisten
- Mulighed for at angive antal deltagere til en booking
- Mulighed for at administrator kan ændre bookinger

4 Usability Testing

Dette kapitel beskriver, hvad vores strategi for usability testing har været samt resultaterne af vores tests.

4.1 Test strategi

Kravspecifikationen beskriver, at det er nødvendigt at lave usability tests tidligt i processen for at bevise proof of concept. Vi lavede papermockups af vores skærmbilleder for at opfylde dette krav.

Vi har valgt at lave to runder af usability tests.
Runde 1 bestod af tests med en papermockup på en enkelt bruger.
Runde 2 består af tests af det endelige produkt på flere brugere.

Vi valgte kun at teste én bruger i runde 1, da flere testpersoner tidligt i udviklingsfasen kan føre til en overvældende liste af problemer[SL, s. 416]. Derudover er en enkelt testperson ofte nok til at afsløre de mest seriøse problemer i brugergrænsefladen.

Vi mener, at det havde været optimalt, hvis vi kunne have udført en usability test midtvejs i processen (mellem runde 1 og runde 2), men vi vurderede, at vi ikke ville få nok ud af testen, da det ville være svært at finde tid til at teste nok personer samt implementere de mulige ændringsforslag.

4.1.1 Test cases

Vi har to brugergrupper i vores usability tests. Der er almindelige brugere (studerende/undervisere) og administration (Facility Management/Receptionister). Til hver af disse brugergrupper har vi defineret syv test cases.

Test cases til almindelige brugere:

Test Case 1: Du har booket et lokale fra 9-10 til et møde med din vejleder og du vil gerne have en diktafon med, så du kan optage mødet.

Test Case 2: Du skal holde et møde på tre timer om et par dage. Du skal i den sammenhæng booke et lokale til formålet.

Test Case 3: Et af dine gruppemedlemmer har meldt tilbage, at han skal aflevere sin datter i børnehaven. Mødet kan derfor først holdes fra klokken 10.

Test Case 4: Da mødet ligger om morgen, tænker du, at det vil være fornuftigt, hvis der var noget kaffe og morgenbrød klar.

Test Case 5: Du skal til dit projekt optage en kort reklamefilm og har derfor brug for et kamera.

Test Case 6: Et af dine gruppemedlemmer har kamera med, så du behøver ikke længere booke et hos ITU.

Test Case 7: Deltagerne til dit møde i 2a12 har desværre aflyst.

Test cases til administrationsbrugere:

Test Case 1: En elev har kontaktet dig og gjort dig opmærksom på, at der ikke længere er en projektor i 2A12.

Test Case 2: ITU har til sin udstyrssamling erhvervet sig to nye kameraer.

Test Case 3: En student henvender sig til dig ved skranken og spørger om hans booking er blevet godkendt.

Test Case 4: Du har et møde i de kommende dage og vil helst gerne holde det på 4. sal.

Test Case 5: Du skal afholde et fordrag for 30 mennesker den 24, tidspunktet er irelevant men der skal være plads til min. 30 mennesker.

Test Case 6: Lokalet 5a12 er ikke længere et privat lokale, men skal istedet registreres som et mødelokale.

Test Case 7: Lokalet 2a12 er ikke længere til rådighed i forbindelse med booking.

Vi har designet vores test cases således, at testpersonen ikke får at vide hvor i systemet, de skal udføre deres arbejdsopgave. Dette har vi gjort for at finde ud af, hvor intuitivt vores system er.
De 14 test dækker til sammen størstedelen af systemets workflows. Vi udfører testene som "think aloud" tests[SL, s. 421]. "Think-aloud" tests består af at læse test casen op for testpersonen og bede dem tænke højt og forklare, hvad de gør under testen.

Efter testen får brugeren mulighed for at give input til, hvad de synes er godt og hvad der kan forbedres. Dette giver os mulighed for at få et indtryk af, hvad vi har overset i vores design og hvilke elementer, vi bør bruge flere steder i systemet.

I første runde af tests havde vi fokus på de store problemer i systemet, hvor vi i anden runde havde fokus på at følge op på de ændringer, vi havde lavet i forhold til resultaterne fra første runde. I anden runde var vi også interesserede i at finde ud af, om vores system er intuitivt og effektivt nok.

4.2 Runde 1: Resultater og konsekvenser

Efter den første runde af tests var der fem problemer, problemerne er klassifikeret i forhold til skalaen fra [SL, s. 439]:

Punkt 1: Medium problem - Usikker på hvordan man vælger booking tidspunkter.

Punkt 2: Cumbersome - Brugeren synes det var besværligt at navigere imellem skærmbillederne.

Punkt 3: Minor problem - Brugeren fandt det ikke naturligt, at man skulle bruge lokalelisten til at ændre sin booking.

Punkt 4: Minor problem - Problem med at tilføje udstyr/forplejning.

Punkt 5: Positiv feedback på den generelle struktur af systemet.

Punkt 1 Test personen havde problemer med at finde ud af, hvordan vores gitter til booking af lokaler fungerede. Personen troede ikke, man kunne klikke på felterne i gitteret, så der skulle lidt hjælp til, før personen forstod princippet.

Konsekvenser og løsning Vi blev enige om, at dette problem kunne løses ved at indsætte en checkbox i felterne. Det vil gøre det mere intuitivt, når man skal trykke på tiderne.

Punkt 2 Testpersonen følte, at der manglede navigeringsmuligheder mellem de forskellige skærmbilleder. Det var ikke intuitivt, at man skulle bruges tabsne til at skifte skærmbillede.

Konsekvenser og løsning Vi løste problemet ved at lave en menubar, som var placeret nærmere midten af billedet, så den blev mere synlig.

Punkt 3 Testpersonen fandt det ikke naturligt, at man skulle bruge lokalelisten til at ændre sin booking.

Konsekvenser og løsning Da vi kun testede på én person og det er et minor problem, valgte vi ikke at gøre noget ved problemet. Hvis problemer opstår i yderligere usability tests, bør en ændring overvejes.

Punkt 4 Testpersonen nævnte, at hun ikke mente, det burde være muligt at tilføje udstyr/forplejning til en booking før, den er blevet godkendt.

Konsekvenser og løsning Vi valgte at imødekomme dette ved at implementere logik, så man ikke kan trykke på knapperne til at tilføje forplejning/udstyr, hvis man ikke har valgt en godkendt booking.

Punkt 5 Testpersonen kunne lide den overordnede struktur af systemet og var positiv overfor booking funktionaliteten (gitteret). Personen skulle dog lære at bruge booking funktionaliteten.

Konsekvenser og løsning Denne feedback havde stor indflydelse på, at vi valgte at beholde vores grundstruktur.

4.3 Usability Test: Runde 2

Vi fik ikke mulighed for at udføre runde 2 af vores usability tests i tide. Da vi planlægger at arbejde videre på systemet efter denne rapport er afleveret, så vil vi følge op på vores strategi ved at lave runde 2 efter at de programmingstekniske fejl er rettet.

Som beskrevet i afsnit 4.1 består runde 2 af test med flere brugere. Der skal laves tests med 2-3 personer fra hver af de to testgrupper. Disse brugere skal udføre "think-aloud" tests som beskrevet i vores usability test strategi. Testcases fra runde 1 skal genanvendes. Desuden vil vi tage en feedback-session efter testen sammen med brugeren på samme måde, som gjort i forbindelse med runde 1.

Målet med runde 2 er, at problemerne fra runde 1 er blevet løst samt at der ikke er nogen major problem eller task failures.

4.3.1 Mulige konsekvenser af usabilitytest runde 2

Konsekvenserne af testresultaterne i runde 2 ville være forslag til ændringer til næste udgave af systemet. Hvis ingen af problemerne fra runde 1 gentager sig, går vi ud fra at det overordnede design/layout er i orden.

I tilfælde af at der kun er cumbersome problemer i forbindelse med brugergrænsefladen, mener vi, at det vil være sandsynligt at systemet er være klar til brug i forhold til usability.

5 Teknisk Design

5.1 Cost-benefit analyse af arkitektur og platform

Da vi skulle implementere vores system var det vigtigt for os, at vi valgte den korrekte implementationsstrategi i forhold til, hvor meget gavn brugeren ville få ud af de forskellige løsninger. Vi overvejede derfor hvilken nytte brugerne ville få hvis vi brugte en implementationsmulighed frem for en anden og hvor meget det ville koste os at implementere det. For at få et overblik over, hvordan disse ting forholdt sig til hinanden valgte vi at lave et skema der viste nytten af en implementation kontra omkostninger.

	Java	.NET	WCF (.NET)	Webservices (Java)	Restful (.NET)	Restful (Java)
	XX	XX				
Thick browser client	O	OO				
	X	X				
Thick application client	O½	O				
			XXX	XXXX	XXX½	XXXX
Webservice (browser client)			OOOO	OOO	OOOO½	OOO½
			XX	XXX	XX½	XXX
Webservice (app. client)			OO½	OOO½	OOO	OOOO

Figure 5.1: Tabel over cost/benefit i forhold til de forskellige implementations muligheder

Figur 5.1 viser vores cost benefit skema over de forskellige implementationsmuligheder. De røde X'er repræsenterer omkostningerne for at implementere den tilsvarende kombination. For at komme frem til omkostningerne af de forskellige implementationsmuligheder tog vi udgangspunkt i hvorvidt vi først skulle tilegne os viden eller om vi allerede havde den fornødende forståelse og således kunne gå i gang med det samme. eksempelvis ville en "Java, Thick browser client" løsning være dyrere end en "Java, Thick application client"løsning da vi ikke ved, hvordan man laver en browser klient i java. Derefter kiggede vi på, hvor meget kode der skulle skrives til de forskellige implementationsmodeller for at kunne få et fungerende system. Eksempelvis er alle Webservice løsningerne dyrere end Thick client løsningerne, da der skal kodes mere, hvis der skal laves en webservice.

De grønne O'er repræsenterer den gavn som brugeren får af den givne implementationsmodel. En af måderne, hvor vi fandt ud af hvor meget gavn de forskellige implementationsmuligheder gav brugeren var at se på, hvor mange platforme der var understøttet.

Eksmepelvis synes vi, at det gav brugeren mere nytte, hvis deres system kunne køre på alle computerere i en browser end hvis man skulle downloade en klient og tjekke om den kunne køre på computeren. Derudover kiggede vi også på, hvor nemt det ville være for brugeren at udvikle videre på systemet, eksempelvis giver implementationsmuligheden "WCF(.NET), Webservice(browser client)" brugeren mulighed for at udvikle en applikation til windows phone da der er en service at kode op imod.

Udfra de måder at vurderer cost benefit på vurderede vi, at den løsning hvor kunden ville få mest gavn var "WCF(.NET), Webservice (browser client)" da det ville give brugeren mulighed for at kunne køre systemet på stort set alle computerere med internet connection og, at de ville være i stand til at lave en mobilapplikation op imod webservicen.

5.2 Databasen

5.3 Servicen

5.4 Klienten

6 Test Af Systemet

6.1 Strategi

6.1.1 Typer af test

6.1.2 Kodedækning

6.2 Resultater

7 Vurdering af vores løsning

7.1 Understøttelse af arbejdsopgaver

Denne sektion beskriver hvilke arbejdsopgaver vi understøtter i første release samt giver en prioritering af, hvilke arbejdsopgaver der bør fokuseres på i følgende releases.

Kravspecifikationen definerer 13 arbejdsopgaver:

C01. Administrer booking

C02. Forespørg på booking

C03. Modtag forespørgsel fra Ekstern

C04. Klargør lokaler

C05. Udlevér nøgle

C06. Håndtér forplejning

C07. Fakturér forplejning

C08. Opdatér menukort

C09. Opdater listen for ekstra udstyr

C10. Opdatér lokaler

C11. Administrér bruger

C12. Håndter statistik

C13. Behandl faktura

Ud af de 13 arbejdesopgaver understøtter vi **C01**, **C09** og **C10**.
Hver af arbejdsopgaverne har en række underpunkter, som beskriver delopgaver og varianter af disse delopgaver.

C01: Administrer booking
Brugere: Ansatte og studerende på ITU.
Hyppighed: Flere gange om dagen i alt for både studerende og ansatte.
Start: En person går ind i booking systemet for at booke, ændre eller få overblik over bookinger.
Slut: Når bookingen eller lændringen er foretaget.

1. Søg efter ledigt tidspunkt for ny booking

1a. Søg efter eksisterende booking

2. Vælg ledigt lokale, forplejning og ekstra udstyr

2a. Vis eksisterende booking(er)

3. Opdater booking

4. Annullér booking

5. Opret booking

C09: Opdater listen for ekstra udstyr
Brugere: Facility Management (FM)
Hyppighed: Når der sker ændring i udvalget af ekstra udstyr (en gang om måneden)
Start: Når der er behov for at udstyrslisten skal opdateres
Slut: Når listen er opdateret

1. Rediger udstyrslisten

C10: Opdatér lokaler
Brugere: Facility Management (FM)
Hyppighed: Når der sker ændring i antallet af lokaler til bookingen (en gang om måneden)
Start: Når der er behov for at lokalelisten skal opdateres
Slut: Når listen er opdateret

1. Rediger lokalelisten
2. Rediger lokale inventar

7.1.1 Prioritering af arbejdsopgaver

Vi valgte til den første release af systemet at priotere

Til den første release af systemet valgte vi at fokusere på at understøtte systemadministration og booking af lokaler og udstyr, hvilket tydeligt kan ses i afsnit **??** da alle arbejdsopgaverne i arbejdsområderne **C1**,**C9** og **C10** er understøttet.

I kravspecifikationen er de arbejdsområder også blandt dem som er vægtet højest og det er samtidig de arbejdsområder som indeholder kernefunktionerne i systemet. Det som ikke blev prioriteret højt nok til at komme med i første release var de arbejdsopgaver som fokuserede på integrationen med kantinen, samt mulighed for fakturing og statestik , arbejdsopgaverne **C6**, **C7**, **C12** og **C13** er eksempler på sådanne opgaver.
Grunden til, at vi ikke tog de arbejdsopgaver med var, at vi vurderede, at de ikke var nødvendige for at have et system der understøttede basis funktioner, derudover lagde arbejdsopgaverne op til, at der skulle implementeres et interface specifikt til kantinen hvilket vi vurderede ville tage lang tid at implementere og vi ville også have tre brugergrupper at tage hensyn til ift. usability og testing.

Fokus for næste release vil derfor være at få udarbejdet et interface som kantinen kan bruge til at integrere med resten af systemet og få finpudset de allerede eksisterende funktioner i kravspecifikationen. Behandling af faktura har også en høj vægtning, så det vil der også blive lagt fokus på.

8 Alternative Systemer

Udover at udvikle vores udgave af et booking-system til IT-Universitetet, vil vi i dette kapitel give en kort beskrivelse af to alternative systemer og sammenligne dem med vores løsning.

8.1 Room Booking System

"Room Booking System"[1] er et web-baseret booking system designet til at opfylde både skoler og virksomheders behov.

Interfacet er designet som en kalender, der viser eksisterende bookinger i farvekoder. Desuden er login til systemet seperat og dermed er der ikke mulighed for at integrere det med hverken Active Directory (AD) eller Where Are You From (WAYF). Et abonnement, der understøtter 200 lokaler/udstyr og 500 brugere, koster 434 kr. om måneden.

En fordel ved "Room Booking System" er, at det giver mulighed for at lave statistikker. Desuden kan det ses som en fordel, at det er hostet eksternt.

Ulempen ved "Room Booking System" i forhold til vores løsning er, at der er en grænse for hvor mange lokaler og udstyrselementer, som kan bookes i systemet. Desuden er det ikke muligt at integrere med et eksisterende ITU login system. Da der kun er mulighed for 500 brugere i systemet, vil nogle brugere skulle slettes for, at der kan oprettes nye i forbindelse med nye studerende/ansatte.

8.2 School Booking

"School Booking"[2] er et andet web-baseret system, som primært er designet til brug af skoler.

Systemet giver mulighed for at booke både lokaler og udstyr[3]. Man kan betale ekstra for at tillade eksterne bookere. Statetisk er også muligt i dette system. Der er dog ikke mulighed for at integrere det med AD eller WAYF.

Et system, der understøtter 1000 brugere, 300 lokaler/resurser, eksterne bookere og statestik koster årligt 11.500 kr.

Fordelen ved dette system er det samme som ved "Room Booking System": muligheden for at lave statestik.

Ulempen ved dette system er, at der er en begrænset mængde af brugere, som kan anvende systemet, samt at det ikke er muligt at integrere det med AD eller lignende, fordi det er hostet eksternt.

Systemet er det, som opfylder kravspecifikationen i højest grad, men det er samtidig det dyreste af alternativerne.

[1] http://www.roombookingsystem.co.uk/
[2] http://www.schoolbooking.com/
[3] Det bookede udstyr kan være forplejning, hvis man sætter rigtigt op.

9 Konklusion

9.1 Lessons learned

I dette afsnit vil vi beskrive de tign, som vi har lært af projektforløbet. Det er specielt ting, som vi i retrospekt ville have gjort anderledes eller bedre.

9.1.1 Kravspecifikationen

I begyndelsen af projektet så vi kravspecifikationen som en række af fejlfri krav, som ikke kunne diskuteres eller bøjes. Vi forsøgte at følge kravspecifikationen i dets fulde omfang i stedet for at opfylde de krav, som vi mente var de vigtigste.

Da vi havde nærlæst kravspecifikationen, fandt vi ud af, at der var dele af den, som ikke var godt designet. Dette førte til, at vi i samråd med Søren Lauesen lavede nogle ændringer til data modellen[1].

Som nævnt forsøgte vi at følge kravspecifikationen i dets fulde omfang, da vi så den som en række krav til en første release. På grund af dette forsøgte vi at designe et system, som kunne håndterer alle de forskellige krav fra kravspecifikationen.
Dette var dog ikke nødvendigt i forbindelse med dette projekt, da det er langt mere interessant, hvordan systemet håndterer de vigtige arbejdsopgaver.
Vores tidsplan skred på grund af dette, og dermed endte vi op med at skære ned på mængden af features alligevel.

Vi kunne have anvendt en iterativ projektstyringsmodel. Dette havde betydet, at vi kunne lave en første udgave tidligere i processen, som vi senere udvidede. I forbindelse med en sådan iterativ projektstyringsmodel ville vi også have haft bedre mulighed for at lave usability tests og funktionelle tests.

Lektien for os har været, at udvikleren skal være opmærksom på, hvilke krav kunden stiller, og være klar til at diskutere kravene med kunden, så man er sikker på, at det bedste produkt bliver lavet. Desuden bør man diskutere en prioriteringsliste med kunden, således at kunden er sikker på, at udvikleren laver det vigtigste først.

9.1.2 Projekter af samme natur er ikke nødvendigvis ens

Da vi begge havde været med til at udvikle et system til filmudlejning, endte vi med at undervurdere, hvor meget arbejde der lå i implementationen af vores design af booking-systemet.

Systemet var udviklet i C# med en WCF-service hooket op til en Microsoft SQL-database med en tilhørende klient. Da dette system delte mange af de samme elementer som systemet, vi har udviklet i forbindelse med dette projekt, så vurderede vi, at det ikke ville være et stort problem at overføre meget af funktionaliteten til dette projekt.

Vi glemte at tage med i vores vurdering, at vi var fem personer på det forrige projekt, og derfor havde vi specialiseret os i forskellige dele af systemet. Dette betød, at de fleste af tingene ikke bare kunne tages med direkte over til vores nye projekt, og derfor blev vores vurdering forkert.

Lektien i denne forbindelse har været, at man skal passe på med at fejlvurdere, hvor meget erfaring fra tidligere projekter kan gavne en i et andet projekt, selvom projekterne er meget ens.

[1] Ændringerne er beskrevet i kapitel 2.

9.1.3 En service er ikke altid den rigtige løsning

I forbindelse med vores valg af implementationsstrategi, lavede vi en vurdering af, hvor meget det ville gavne kunden (ITU), hvis vi udviklede en service som en del af systemet. Vi vurderede, at det ville være en stor fordel for kunden, hvis det var nemt at udvide systemet.

Den første release, som vi udviklede i forbindelse med dette projekt, havde dog slet ikke brug for en service. Vi burde i stedet have koblet vores klient direkte til databasen.
Arkitekturen, vi anvender i klienten, gør det relativt smertefrit at koble klienten op til en service i stedet for direkte til databasen, hvis systemet senere skulle udvides.

Vi tænkte ikke over, at projektperioden var begrænset, så vi vurderede kun omkostningerne i vores cost/benefit-analyse i forhold til hinanden. Dette betød, at vi ikke havde lavet en egentlig vurdering af, om det var muligt for os at nå udvikle både service og klient.

I fremtidige projekter vil vi sørge for at vurdere, om der overhovedet er brug for avancerede features i forhold til kundes behov.

9.1.4 Projekt styring

Vores strying af projektet har været meget løs og parallel.

Det havde været en fordel for vores projekt, hvis vi havde haft delmilestones eller en iterativ styringsmodel. Vi ville have haft bedre styr på, hvor meget vi manglede på et givent tidspunkt i processen. Desuden ville vi haft mere materiale at få feedback på fra vores vejleder.

Da vi har arbejdet meget parallelt, har vi ikke været gode nok til at holde hiinanden opdateret på, hvor man var i forbindelse med det, man arbejdede på. Vi burde have lavet en form for stand-up meeting, så vi fik holdt hinanden opdateret.

Dette har tydeliggjort vigtigheden i, at man dokumenterer, hvad man laver og beslutter. Hvis vi havde haft flere delmilestones, ville vi også have været i stand til at se, om vi var på samme bølgelængde i forbindelse med projektet.

10 Litteraturliste

Bibliography

[SL] Lauesen, Søren. User Interface Design, A Software Engineering Perspective. Great Britain: Pearson Educated Limited, 2005. Print.

[KravSpec] Miki Ipsen, Garwun Jeffrey Lai, Merete Larsen, Stig Larsen. Kravspecifikation til ITU Booking System. December, 2012.

Appendices

A System Diagrammer

B GUI-Images

B.1 Paper mockup

B.2 Endelige skærmbilleder

C Udvalgte Dele Af Koden

C.1 SQL-kode til oprettelse af database

C.2 BookItContext.cs

C.3 Configuration.cs

C.4 Booking.cs

C.5 BookingManagement.cs

C.6 BookingModel.cs

C.7 RoomListViewModel.cs