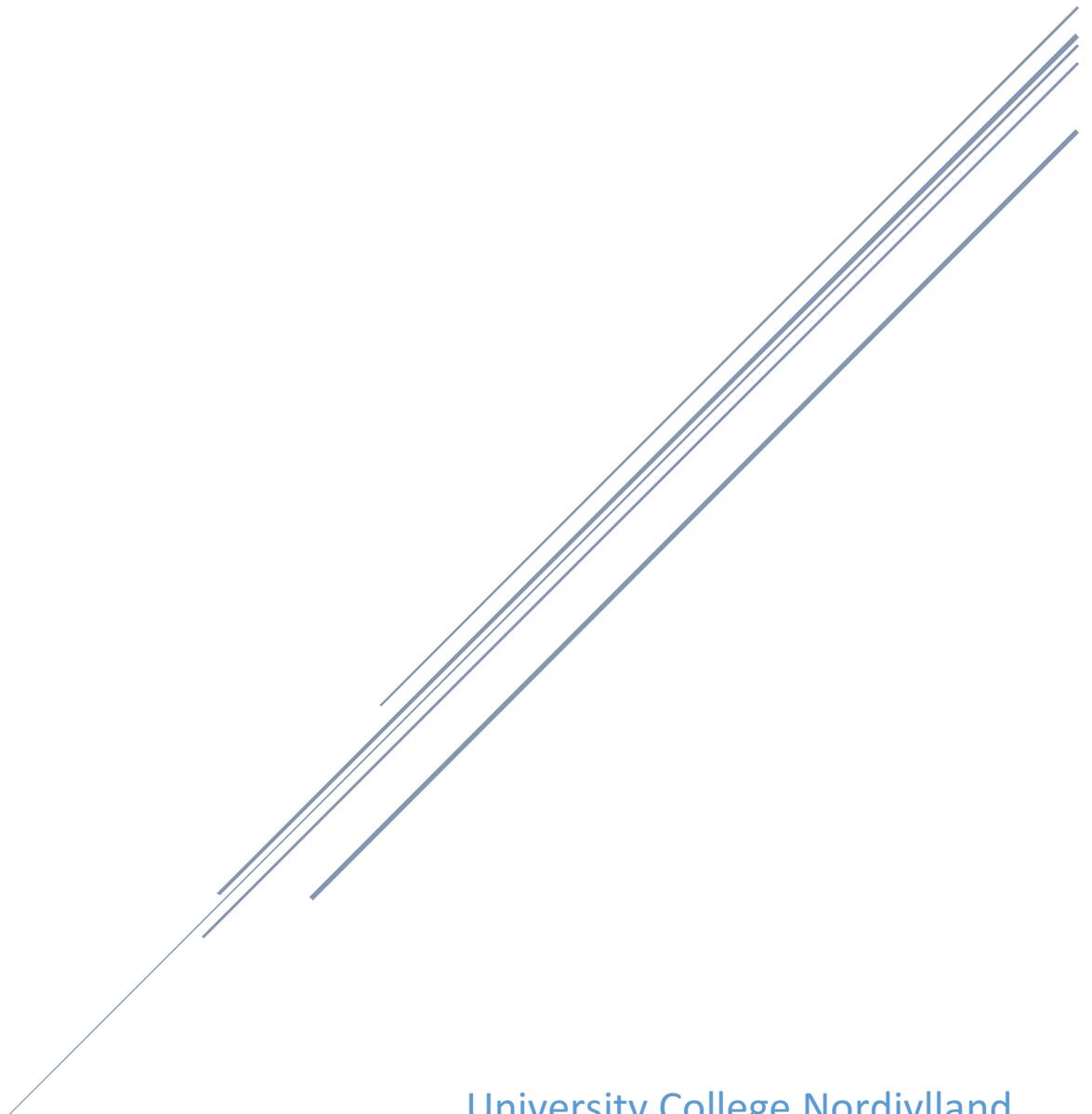


PROJEKT GRUPPERUM

Systemudviklingsrapport



University College Nordjylland
Gruppe 2 - David, Dina, Emil & Mark

Titelblad



Sted: University College Nordjylland, UCN

Uddannelse: Datamatiker 3. semester i Aalborg

Emne: Systemudviklingsrapport

Gruppe: 2

Titel: Projekt Grupperum

Afleveringsdato: 16. december 2015

Vejleder: Rasmus Christiansen Knap

Sider: 22

Normalsider: 14,42

Abstract:

This rapport contains theories about agile programming, like Kanban, Scrum and eXtreme programming. We will explain certain patterns in agile programming, and explain how we used them in our development phase. We will look upon the theories, and use the best of every one of those, to create the best possible development environment for our group.

David Mathias Kolind, 1033937: _____

Dina Sandvad, 1034352: _____

Emil Andersen, 1034514: _____

Mark Hjorth Sørensen, 1033933: _____

Indhold

| | |
|--------------------------------------------------------------------|----|
| Titelblad | 1 |
| Abstract: | 1 |
| Forord | 4 |
| Indledning | 4 |
| Problemformulering som den fremgår af tekniske dokumentation | 4 |
| Problemformulering | 5 |
| Systemudvikling | 5 |
| XP | 6 |
| Communication | 6 |
| Simplicity | 6 |
| Feedback | 6 |
| Courage | 7 |
| Planning game | 7 |
| Small releases | 7 |
| Metaphor | 7 |
| Simple design | 7 |
| Define test first | 7 |
| Refactoring | 7 |
| Pair programming | 7 |
| Collective ownership | 7 |
| Continious integration | 7 |
| 37-hour week | 8 |
| On-site customer | 8 |
| Coding standards | 8 |
| Planlægning | 8 |
| Planning poker | 8 |
| Scrum | 8 |
| Product backlog | 9 |
| Sprint backlog | 10 |
| Sprint | 10 |
| Working increment of the software | 11 |

| | |
|-----------------------------------------------|----|
| Kanban | 12 |
| Vores proces..... | 13 |
| Sprint 0..... | 14 |
| Sprint 1..... | 14 |
| Sprint 2..... | 15 |
| Sprint 3..... | 16 |
| Sprint 4..... | 17 |
| Konklusion..... | 18 |
| Kildehenvisninger | 19 |
| Bilag | 19 |
| Bilag 1: Vandfaldsmodellen | 19 |
| Bilag 2: Noter fra møde med servicedisk | 19 |
| Bilag 3: Personas..... | 20 |

Forord

Denne rapport er skrevet af datamatiker studerende på UCN, David Mathias Kolind, Dina Sandvad, Emil Andersen og Mark Hjorth Sørensen, som et eksamensprojekt på 3. semester.

Projektarbejdet er påbegyndt i efteråret 2015 og afleveret 16. december 2015 kl 12.00.

Denne rapport har til formål at samle teori, fra 3. semester faget "Systemudvikling", med en procesbeskrivelse af den praktiske tilegnelse, som er målet for semesteret.

Rapporten beskriver processens formål og metoder som vil kunne opfylde formålet. Med udgangspunkt i de benyttede metoder, forklares og diskutes teorier, deres fordele og ulemper samt de valg vi har foretaget. Til slut fremlægges konklusionen i forhold til problemformuleringen.

Indledning

Selve projektet bygger på ideen om et system til udlejning af grupperum, som kan forbedre måden, hvorpå det foregår på UCN nu, hvor studerende skal møde op og bede en medarbejder manuelt undersøge mulighederne for at få et grupperum og foretage registreringen.

Problemformulering som den fremgår af tekniske dokumentation

- Hvordan kan vi lave et system, som de studerende selv kan tilgå, oprette grupper og leje lokale igennem, så udlejningen bliver lettere tilgængelig og forhåbentlig mere tilfredsstilende for både studerende og ansatte? Hvordan kan et system give de studerende mulighed for at vælge lokaler ud fra bestemt inventar i lokalerne og også tage hensyn til at en gruppe kan leje et bestemt lokale i f.eks. 5 dage i træk, så de kan indrette rummet efter deres behov?
- Hvordan kan de UCN-ansatte administratorer få et overblik over hvem der har lejet hvilke lokaler og få voret til at slette en gruppe fra et lokale hvis gruppen misligholder aftalen for lokalebrugen?

I forhold til hele opgaven med at nå frem til et brugbart system, kodet i fælleskab af usynkroniserbare mennesker, er der en del kommunikative udfordringer. For at sikre høj kvalitet, både i det som bygges og i samarbejdet, er der efterhånden kommet gode og spændende metoder på banen, som hjælper processen i den rigtige retning.

Problemformulering

Hvordan kan vi ved hjælp af teorier lært i undervisningen, lave et effektivt arbejdsmiljø til udvikling. Hvordan kan vi effektivisere processen og samtidig sikre kvalitet? Hvordan kan vi finpudse Kanban, Scrum og eXtreme programming til at fungere bedst muligt til vores gruppe?

Systemudvikling

Overordnet set – Hvad er det at arbejde agilt og hvorfor giver det særlig mening i softwareudvikling? Når software skal udvikles, er udgangspunktet et behov eller et problem. Der vil ofte gå lang tid, nogle gange år, fra man går i gang med at analysere et behov, beslutter sig for en løsning og får sat udviklingsprojektet i søen, og til det færdige software er klart. Når vi har med IT at gøre, er netop "tid" altafgørende. IT verdenen er i hurtig udvikling og dermed kan forudsætningerne for et projekt være helt væk eller markant forandrede, over den tid der er gået fra den første analyse og til projektet står færdigt.



FIGUR 1 - VISUEL REPRÆSENTATION AF ET IKKE AGILT SYSTEM.¹

Taler man om et motorvejsprojekt kan behov og forudsætninger også ændre sig, men man står ikke med et årelangt motorvejsbyggeri som, når det er færdigt, er ubrugeligt fordi folk er stoppet med at bruge biler. Hvis det skete ville det være katastrofalt og et enormt spild af ressourcer. Ikke desto mindre er det faktisk det, der er sket, i adskillige IT projekter. Samfundet, teknikken og behovet har ændret sig så meget undervejs i et projekts tilblivelse, at det software som skulle løse et problem, er forældet inden det bliver frigivet. Som eksempler kan nævnes rejsekortet, skats EFI system og mange andre. Problemet med disse systemer er dog ikke kun udefrakommende forandringer, men også problemer med at håndtere så kæmpestøre systemer i sig selv. Begge dele, de udefrakommende forandringer og det ekstreme detaljeoverblik, er problemer som agile udviklingsmetoder tager op.

¹ Kilde: http://ekstrabladet.dk/incoming/article4930646.ece/IMAGE_ALTERNATES/p900/Torm%20Gerd



FIGUR 2 - VISUEL REPRÆSENTATION AF AGIL UDVIKLING.²

De agile metoder har til fælles at sigte efter tilpasningsparathed og evnen til at ændre retning og tilpasse målet undervejs. Man kan bruge billedet af et tankskib som ligger stabilt på sin retning i forhold til en lille motorbåd som hurtigt kan navigere og ramme en ny destination. Traditionelt set har meget udviklingsarbejde fungeret som et tankskib. Man har brugt en fast form, som for eksempel vandfaltsmodellen (se Bilag 1), til f.eks. analyse, udvikling og test, som store arbejdsopgaver, gennemarbejdet grundigt én ad gangen. Det kan fungere, men som sagt kan det give alvorlige problemer for det færdige system, at der ikke er givet mulighed for tilpasning og ændringer undervejs.

XP

Xtreme Programming (xp) er en af de agile fremgangsmåder der bruges til programmering. Xp er baseret på fire værdier og 12 praktikker som er essentielle for xp.

Xp's fire værdier er:

Communication

Kommunikation er helt essentiel for at et projekt lykkes. Xp's koncept bygger på at have en god effektiv kommunikation, så man i højst mulige omfang er på "same page". Det gælder både i forhold til productowneren, hvis ønsker har extrem høj prioritet i xp, og det gælder i forhold til teamet imellem.

Simplicity

Simpelhed i koden gør det nemmere at udføre ændringer. Det betragtes som spild at tænke for langt frem og forberede koden til eventuelle tilbygninger. Hellere lave en enkel kode med nøjagtig den funktionalitet der er brug for og sidenhen ændre efter aktuelle behov.

Feedback

Alt bliver bedre med klar og modig feedback. Det gælder både i planlægningsprocessen og i selve programmeringen, at hyppig feedback hjælper med at styre den rigtige vej.

² Kilde: <http://5starhvar.com/wp-content/uploads/2013/04/private-speed-boat-transfers.jpg>

Courage

God programmering kræver mod. Mod til ærlighed for et godt frugtbart samarbejde og mod til at nytænke og ændre på projektet.

Ud over de 4 værdier som må gennemsyre arbejdsprocessen, lægger xp vægt på 12 praktikker. De 12 praktikker er som følgende skrevet med blåt, og kort beskrevet i paranteses.

Planning game

En metode til planlægning som bruges til udforskning, tilslutning og styrefaser er kunden af stor relevans – kunden vælger hvad der er vigtigt at betale for.

Small releases

Ved at målrette projektet mod at få små simple løsninger gjort færdige hen ad vejen, i stedet for at det hele bliver færdigt på én gang, viser man productowneren at der sker noget. Det giver en tryghed for productowner og det giver også mulighed for at productowner kan se systemet og få øje på eventuelle misforståelser inden det bliver katastrofalt.

Metaphor

Fælles forståelse af begreber og sprog er vigtig for kommunikationen mellem udviklere og kunder.

Simple design

Never prepare for tomorrow, work for today. Lav et minimum for at få det aktuelle op at køre. Brug kun diagrammer hvis nødvendigt. Brug ingen duplikeret uoverskuelig kode.

Define test first

Skriv unit testene først – Det giver gode overvejelser om, hvad der kan gå galt og kan give frugtbare input til udviklingen af koden. Red/Green refactor giver også en stor tryghed ved refactoring, fordi testene er en god sikkerhed for kodens duelighed før der committeres.

Refactoring

Refaktorering (oprydning i koden) er vedvarende omskrivning af programmet UDEN at ændre funktionalitet. Man sigter efter at undgå duplikeret kode, for lange metoder, for store klasser, lange parameterlister osv. Kodeordet er stadig simpelhed og overskuelighed

Pair programming

Al kode og unittest skrives parvis – Denne praktik går hånd i hånd med værdierne kommunikation og feedback. At være to om koden forebygger fejl, og løfter det personlige niveau. Desuden kan en mand blive syg uden alt vælter.

Collective ownership

Alle kan ændre i al kode. Der er ikke noget "*min* kode". Alle har ansvar for at højne koden.

Continious integration

Når userstories deles op i tasks, så skal de ikke være større end hvad man kan nå på en dag. For hver arbejdsdag på koden skal den kunne bygges, således man ikke ophober bugs.

37-hour week

Giver effektive nærværende medarbejdere der ikke brænder ud.

On-site customer

Kunden er involveret hele vejen for at sikre fælles vision af programmet.

Coding standards

En aftalt kodestandard er helt nødvendigt, når alle skal kunne refaktuere.

Planlægning

Planlægnings processen starter med planning game.

I planning game laver kunden "user stories", som beskriver vigtige funktionelle krav til produktet. Normalt er disse skrevet i forretnings henseenede, og dermed ikke i tekniske termer. Når disse user stories bliver lavet, kan man sikre user storiernes kvalitet ved at tænke dem ud fra INVEST begreberne.

En gennemført user story skal være uafhængig af andre, **Independent**. Detaljerne i en user story skal kunne diskuteres i dialog med product owner. Den skal altså være **Negotiable**. Alle user stories skal have værdi for product owneren, **Valuable**. Udviklere skal kunne **Estimere**, alle user stories, derfor er det vigtigt at problemstillingerne er konkrete. Ved at gøre user stories korte, **Small**, er de ofte også blevet mere konkrete. Sidste step er at gennemtænke vigtige ikke-funktionelle krav for at gøre dem **Testable**.

Når en alle user stories er blevet accepteret, vil man estimere deres sværhedsgrad. Dette gøres ved hjælp af planning poker

Planning poker

I planning poker er vores mål at estimere alle user stories i forhold til hinanden. Dette gør vi ved at give dem "Story points". Story points er uafhængige af tid. Deres værdi bliver bestemt i forhold til hinanden. User stories bliver tildelt et antal storypoints udfra Planning poker. I planning poker giver man stories points fra 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100. Når man laver estimation, samler man udviklerne, således de kan debattere opgaverne. Alle vælger en værdi for en given opgave, hvis der er stort udsving i værdierne valg, skal udviklerne forklare hvorfor de synes deres givne værdi er korrekt. Derefter reestimeres user storien og den korrekte værdi findes.

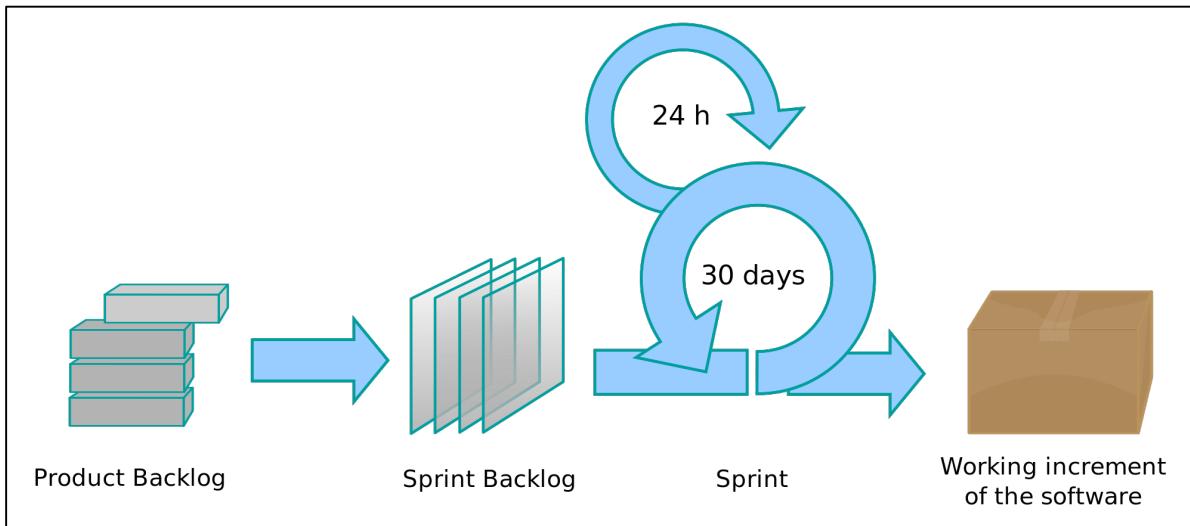
Scrum

SCRUM og xp har flere fællestræk, bl.a. målet om løbende at skabe værdi for kunden, og den agile tilgang med iterationer, som dog i SCRUM kaldes sprint. Produktet opdateres altså efter hvert sprint, så kunden langsomt får et bedre program som kan benyttes.

I Scrum hviler ansvaret også på teamet som bærer ansvaret sammen, men Scrum teams udnævner dog en scrummaster, som er en slags hjælper. Scrummasteren kalder hver dag teamet sammen, til de daglige evalueringer (kaldet standup meetings). Scrummaster lukker herefter

mødet og uddelegerer eventuelle problemer, som er for store til at diskuteres på et standup meeting.

En demonstrering af Scrum kan ses på Figur 3.



FIGUR 3³

Som vist på Figur 3, er Scrum delt op i 4 hovedstadier, nemlig "product backlog", "sprint backlog", "sprint" og "working increment of the software". I de følgende 4 afsnit vil de 4 stadier blive forklaret og hvad der skal til for, at kunne gå videre til det næste stadiet.

Product backlog

I dette stadiet er productownerne til stede og arbejder sammen med udviklerne. Productownerne er ejeren af produktet (kunden) og er den som skal beskrive de opgaver, som det færdige produkt skal kunne løse. Produktbeskrivelserne kaldes for product backlog items. Hvert product backlog item beskriver en del af det færdige program, et eksempel kunne være:

"Som UCN ansat, vil jeg kunne tildele et grupperum til en gruppe, så det fremgår som udlejet."

Product Backlog præciserer hvilken bruger, hvilken funktion og hvilket resultat der er fokus på. Dernæst præciserer udviklerne, i samarbejde med product owner, hvilke krav der skal opfyldes, før dette product backlog item er i mål. Ved at præcisere opgaven og bryde den ned til præcise krav har udviklerne et grundlag til at vurdere opgavens tidsmæssige størrelse. Den vurderes med point fra 0-100. Jo tættere en backlog er på 0, jo lettere er den at lave, og omvendt jo tættere den er på 100, jo sværere er den at lave.

For lettere at kunne vurdere hvor mange point et backlog item skal have, diskuterer udviklerne hvilket backlog item der er det mest komplicerede og bedømmer den først. Når det mest kom-

³ Kilde: https://upload.wikimedia.org/wikipedia/commons/thumb/5/58/Scrum_process.svg/2000px-Scrum_process.svg.png

plicerede backlog item er bedømt bedømmes de følgende backlogs ud fra denne. Det første backlog item der bliver bedømt, fungerer altså som standpunkt.

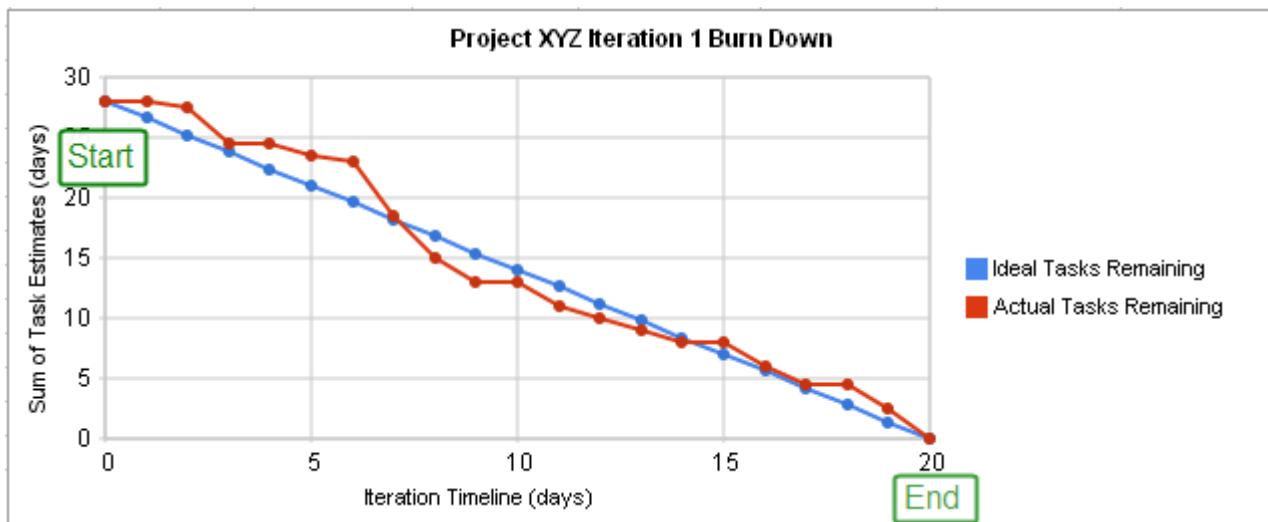
Overgangen fra dette stadie til det næste, foregår ved, at udviklerne bedømmer hvor mange point de forventer at kunne nå på et sprint. Ud fra den vurdering kan product owner få en fornemmelse af hvilke funktioner der er "dyre" i forhold til forretningsværdien og sammen med udviklerne blive enige om hvilke backlog items der skal tages først. Dette møde kaldes Sprint Planning 1.

Sprint backlog

Sprint backlog er oversigten over opgaverne i ét sprint. På et møde kaldet Sprint Planning 2, brydes hvert backlog item op i tasks, som er mindre opgaver der skal til for at fuldføre kravene. Antallet af point som et backlog item har, definerer hvor mange point der skal fordeles på tasks, således vil et backlog item på 10 point have 10 point at fordele på tre tasks med f.eks. 7, 2 og 1 point.

Sprint

Udviklingsteamet er selvorganiserende og fordeler i fællesskab de forskellige tasks imellem sig. Et sprint er en arbejdsperiode på et bestemt antal dage. I dette eksempel, som vist på Figur 3, tager et sprint 30 dage. Et sprint kan tage længere eller kortere tid, men tager altid det samme stykke tid i ét projekt. Hver morgen, i de 30 dage, vil udviklerne tage et stand up-meeting; Alle rejser sig, kommer væk fra skærmene og hen til taskboardet, og hver udvikler beskriver hvad de har lavet og hvor langt de er nået. Hvis en task er estimeret til at koste 5 point, vil udvikleren der er tildelt denne opgave beskrive, i point, hvor lang tid der er tilbage af opgaven, før den er færdig. Standup meeting kan også bruges til at fremlægge problemer. På denne måde kan udviklingsteamet holde et fælles overblik og følge med i hvor mange point der er tilbage før de er færdige. Som en grafisk metode til dette overblik laver man et burn down chart, se eks på Figur 4.

**FIGUR 4⁴**

På Figur 4 vises et burn down chart over et sprint på 20 dage. Som vist på figuren er der tegnet en blå linje der demonstrerer den ideelle nedskrivning af tasks, hvor der er taget højde for tid og personer til rådighed, som gør nedskrivningen realistisk. Den røde linje beskriver den reelle nedskrivning indtil alle tasks i sprintet er kommet i mål.

Burn down chartet, gør det let for udviklerne at se om de er foran, bagefter eller følger planen. I tilfælde af, at udviklerne ikke kan nå alt, hvad de havde sat sig for i indeværende sprint, vil de, i samarbejde med product owner, vælge de backlogs der skal tages af, og overføres til næste sprint. Ligeledes vil product owner kunne inddrages, hvis burndown går så godt at der er overskud til at tage flere backlogs ind.

Når de 30 dage i sprintet er overstået er udviklerne klar til at gå videre til næste stadie.

Working increment of the software

Ved sprintets afslutning fungerer softwaren, som overleveres til productowner ved et Sprint Review møde. Sprint Review er åbent for alle interesserter som har interesse i processen, uden det dog giver taleret til alle. Sprint Review har fokus på forretningsværdien, altså produktet udadtil, som product owner forstår og kan vurdere. Efter dette møde holdes et møde kun for udviklerne. Dette kaldes Retrospective og er udviklernes mulighed for at evaluere sprintet sammen ud fra deres vinkel. Her vurderer udviklerne også hvor mange point de kan tage ind i næste sprint. Når dette stadie er slut, vil udviklerne starte forfra igen, med at tage backlog items ind i et nyt sprint og foretage vurderinger sammen med productowner som tidligere beskrevet.

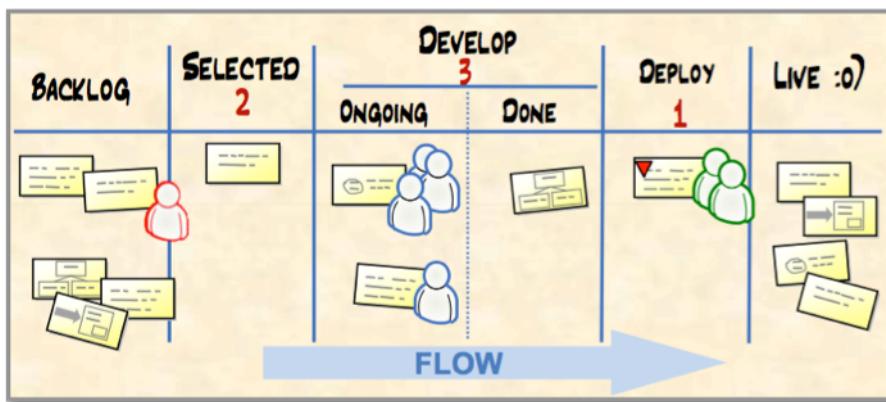
⁴ Kilde: https://upload.wikimedia.org/wikipedia/commons/8/8c/Burn_down_chart.png

Kanban

Kanban er en arbejds metode der fokuserer på antallet af opgaver der er i gang, og deres status, fremfor at fokusere på hvor lang tid der er tilbage til alle opgaverne. Kanban er et helt system der handler om, detailplanlægning og forudseenhed i produktion såvel som salg. Det handler om at gøre behov synlige helt ned i detaljer og foregrive flaskehalse.

I softwareudvikling er især Kanban boardet blevet populært og det er den del af Kanban vi vil fokusere på.

Kanban arbejder under tre regler som sætter rammerne for arbejdet. For det første skal man visualisere sit workflow. Dette gøres med et Kanban Board, hvor alle ens opgaver og de forskellige faser er repræsenterede. På den måde kan man let se hvor hver enkelt opgave er i processen, og man kan følge at alle opgaver flytter sig, så man ikke går i stå. For det andet skal man begrænse sit WIP (Work In Progress). Det betyder at man sætter en fast grænse for hvor mange opgaver der kan være i de forskellige faser. På den måde sørger man for at et udviklingshold ikke tager flere opgaver ind end de kan håndtere at løse. Til sidst skal man måle og forsøge at optimere sin Lead Time, som er den tid det tager en opgave at komme igennem hele Kanban Boarded. Jo lavere Lead Time man har, jo flere opgaver kan man få igennem processen.



FIGUR 5 - KANBAN BOARD.⁵

Et Kanban board består af forskellige "faser", der viser hvor langt de forskellige opgaver er i processen. De forskellige faser er illustreret på Figur 5. Faserne kan kaldes flere forskellige ting som man synes er sigende. Der kan også være flere eller færre faser alt efter behov. Fælles for alle faserne er, at de alle har et tal associeret til sig. Disse tal beskriver hvor mange tasks en fase kan have.

Som det kan ses på Figur 5 ovenfor, kan den første fase f.eks. hedde Backlog. Denne fase indeholder alle tasks der endnu ikke er blevet implementeret. Product owner (som er markeret med rød på Figur 5) kan derfor vælge to tasks ad gangen, som product owner ønsker lavet. Når product owner har valgt en task flyttes denne til "selected" fasen. Som det kan ses på figuren

⁵ Kilde: <http://www.mcpa.biz/wp-content/uploads/2015/08/Kanban-Board.png>

kan "selected" i dette eksempel indeholde to tasks. "selected" fasen fungerer altså som en form for kø, for at fortælle udviklerne hvad product owner ønsker.

Som vist på figuren hedder den næste fase "Develop" hvilket er delt op i to sub-faser; "Ongoing" og "Done". I denne fase kan udviklerne have 3 tasks ad gangen. Når udviklerne er klar til at arbejde på en task tager de den fra "Selected" til "Ongoing". I "Ongoing" fasen arbejder udviklerne på at løse tasken. Når tasken er færdig bliver denne placeret i fasen sub-fasen "Done". Som det kan ses på figuren, må udviklerne i "Develop" ikke tage flere tasks ind, selvom der ligger 3 tasks i "Done". I stedet kan udviklerne gå fra deres post og hjælpe "Deploy" holdet.

I næste fase, "Deploy", kan der holdes én task. Når holdet er klar til at tage en opgave ind, tager de denne fra "Done" til deres fase "Deploy". Holdet sørger, i denne fase, for at sende de løste opgaver videre til product owner så de fungerer med det nuværende system.

En meget vigtig ting i Kanban er at fokusere på at håndtere flaskehalsproblemer. Disse problemer kan opstå hvis de forskellige hold ikke arbejder lige hurtigt eller hvis en opgave giver problemer, og derved bremser processen i en fase. Disse problemer kan løses på flere måder. For det første kan man sætte en højere WIP på de sidste faser i processen, men på den måde risikerer man at skubbe problemet så det kun bliver større når man endelig er nød til at håndtere det. Man kan i stedet for vælge at flytte arbejdskraft rundt i de forskellige faser, så man får flere personer til at arbejde på de faser der tager længst tid så de kan nå mere.

I løbet af et projekt kan der komme opgaver som har høj prioritet. Det kan være der er opstået en alvorlig fejl som skal rettes, eller der kan være en funktion man skal have implementeret for at systemet virker optimalt. I Kanban kan man, i modsætning til Scrum, tage disse prioritets opgaver ind lige så snart der er plads på Kanban Boarded. Det vil sige at en prioritets opgave kan løses relativt hurtigt i Kanban, hvor den i Scrum er nød til at vente til det igangværende sprint er slut og det næste starter.

En ulempe ved Kanban, i forhold til Scrum, er at man tager opgaver ind i processen når man har tid og ikke selv er direkte ansvarlig for om man kan nå det i det fastsatte tidsrum, i modsætning til Scrum hvor teamet selv sætter en ramme for hvor meget man kan nå i et givent tidsrum og derfor vil være mere engagerede i at nå det.

Vores proces

I vores 3. semesters projekt om grupperum ønsker vi at afprøve noget fra de ovenstående teorier i praksis. Det bliver en blanding af værdierne og praktikkerne fra xp sammen med Scrums sprint og rutiner. Burndown charts forventer vi vil blive et fast holdepunkt sammen med et Kanban board til daglige standup meetings.

Der er helt klart nogen udfordringer i at afprøve disse teorier i praksis da vi jo ikke har en reel productowner og dermed de forventninger samt den anderledes forståelsesramme, som en sådan ville have bidraget med. Desuden er det også en udfordring, at vi ikke fra starten af projektet har haft al den viden, som der skal bruges i projektet. Det gør det vanskeligt at estimere opgavers point, når vi endnu ikke har modtaget undervisningen til at kunne løse pågældende opgaver.

Udarbejdning af ide. For at belyse ideen med et grupperumsbookningssystem talte vi med en af de UCN ansatte i servicecenteret. Vi spurgte ind til problemerne, som de oplever dem og spurgte dem hvad de kunne forestille sig af løsninger. Vi satte os desuden ned og beskrev en række personas⁶, som vil være realistiske brugere af et sådan system. Vores personas bestod af ansatte og studerende og dermed altså brugere af en webklient og et distribueret-system. Ud fra det lille interview⁷ var det tydeligt at det system som de umiddelbart forestillede sig, ikke rummede den kompleksitet, som er nødvendigt for at sikre studierelevante udfordringer i projektet.

Sprint 0

Vores proces startede med "Sprint 0" der forløb fra d. 13. oktober til d. 10. november. Dette var et indledende sprint, med fokus på at gøre alt klar til at kunne kode i Sprint 1. Under sprint 0 blev domæne og interaktionsdiagram lavet. Til repository og versionsstyring blev det aftalt at bruge visual studios repository på www.visualsstudio.com, sammen med programmet SourceTree⁸.

Samtlige user stories/backlogs blev estimeret med planning poker, hvorefter de også blev prioritert fra productowners synspunkt med værdierne A-M, med A som højest værdi. Vi gav dem også en sekundær prioritering i forhold til deres vigtighed for vores projekts krav ud fra studiet, især vedrørende algoritme, samtidighed og atomicitet. Dette blev gjort, da disse to prioriteringer ikke stemmer overens.

Vi blev enige om, så vidt muligt at følge product owners prioitering, men hvis dette ikke længere var muligt pga. tid, ville vi tilsidesætte den, og i stedet nå de krav der blev stillet til os i fagene teknologi og programmering.

Sprint 1

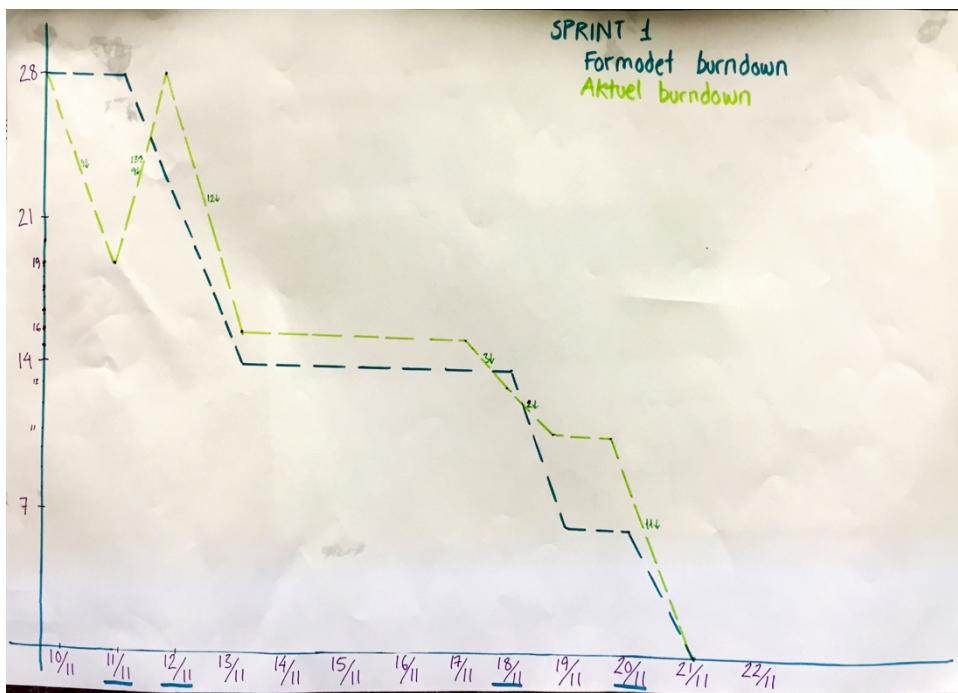
I perioden fra den 11. til d. 20. november var Mark scrummaster. Vi vil følge de teorier vi har læst og lært om i undervisning, dvs. Kanban, Scrum og eXtreme programming. Da gruppen har fire medlemmer, er det to par til par programmering. Vi tog de første to user stories A og B ind, og delte dem op i tasks. Dette vil vi gøre som det første i begyndelsen af hvert sprint. Vi har estimeret at vi skal nå 28 story points. Emil og Mark blev sat til at programmere sammen og de fandt hurtigt ud af, at de havde to vidt forskellige ideer om hvilke klasser der skulle håndtere det result-set som vores database connection hentede op fra databasen, og hvilken klasse der skulle modellere disse. Dette førte til projektets første diskussion om hvilken klasse indenfor database området der gør hvad. Uden par programmering, ville der uden tvivl være blevet brugt to forskellige måder at håndtere modelleringen af klasser på.

Efterfølgende fik vi færdiggjort alle tasks indtil tasks vedrørende webclienten. Vi havde ikke modtaget noget undervisning i dette endnu, og vidste ikke hvordan det fungerede. Det blev besluttet at tage en userstory mere ind på 13 points, som vi kunne arbejde på indtil vi fik undervisning i webudvikling. Dette fremgår også af vores burndown nedenfor.

⁶ Se Bilag 3.

⁷ Se Bilag 2.

⁸ SourceTree: Et revisionsværktøj der fungerer med windows og mac osX.



FIGUR 6 - BURNDOWNCHART FOR SPRINT 1. DATOERNE MED STREG UNDER ER DE REELLE PROJEKTDAGE.

Selvom sprint 1 var officielt på fire arbejdsdage, blev der brugt meget undervisningstid på at spike⁹ og finde ud af hvordan MVC fungerede. Sidste dag på sprint 1 blev brugt på MVC problemstillingen med at få checkboxe til at virke. Alle sad og spikede i flere timer indtil vi endelig fik fundet en løsning. De sidste points blev brændt af 20 minutter før deadline.

Sprint 2

Den 23/11 startede med retrospektive. Når vi kigger tilbage på sprint 1 skal vi finde de ting der er vigtige at arbejde videre på, og holde på de ting der virkede.

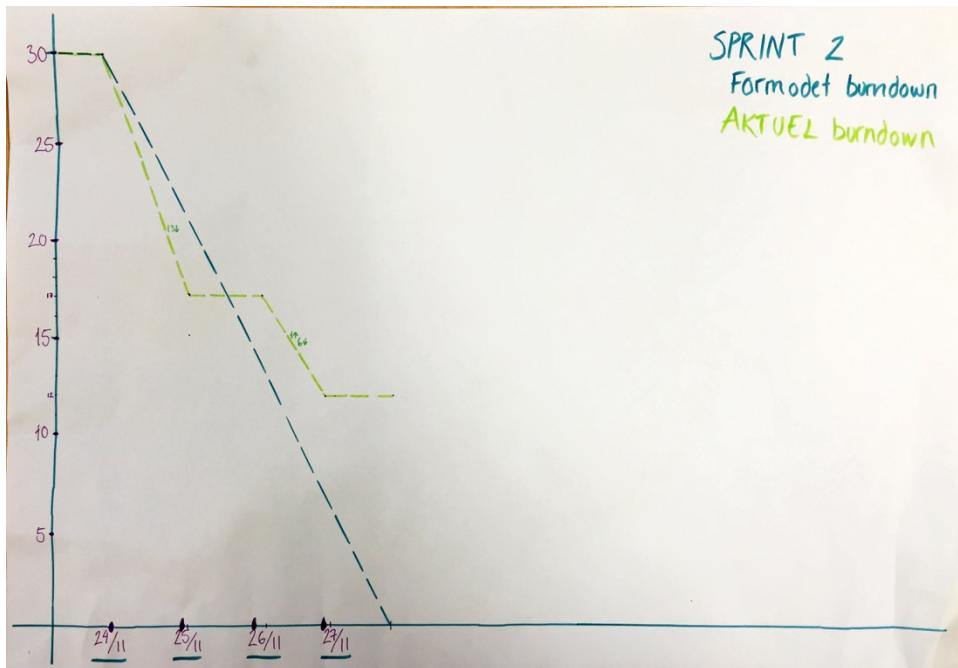
Vi blev enige om at en udvidet kodestandard kunne have været en god guideline at have, sådan man har samme opfattelse af hvordan programmeringen skulle foregå. Dermed giver vi xp ret i fornuften i en af deres praktikker. Desuden var web-delen en kæmpemæssig overraskelse udfordringsmæssigt. Den var underestimeret, heldigvis havde vi tid i undervisningen til at spike, sådan vi ikke brugte hele sprintdage på det. Ifølge eXtreme programming skulle programmeringen være test-first, men da ingen vidste hvordan dette skulle udføres, var dette blevet skubbet til side. Dette er helt klart en ting vi skal have implementeret i vores arbejdsproces i sprint 2.

Mark havde lavet gode ting i løbet af weekenden, der blev nemlig implementeret branches til projektet, sådan at vi nu arbejdede på branches, og mergede først til master når det var blevet

⁹ At 'Spike' betyder at teste forskellige ideer og lave 'proof of concept' kode, som eventuelt kan implementeres i systemet senere.

testet. På denne måde kunne vi have et færdigt build klar til hvis produktowner skulle komme forbi.

Par programmeringen virkede efter hensigten. Der blev sparret med hinanden, og snakket kode igennem. Dette fungerede rigtig godt på baggrund af princippet communication og courage i xp. Når kommunikationen og modet til at være ærlige overfor hinanden og sige ens mening om koden og projektet, bliver problemstillingerne diskuteret igennem.



FIGUR 7 - BURNDOWNCHART FOR SPRINT 2.

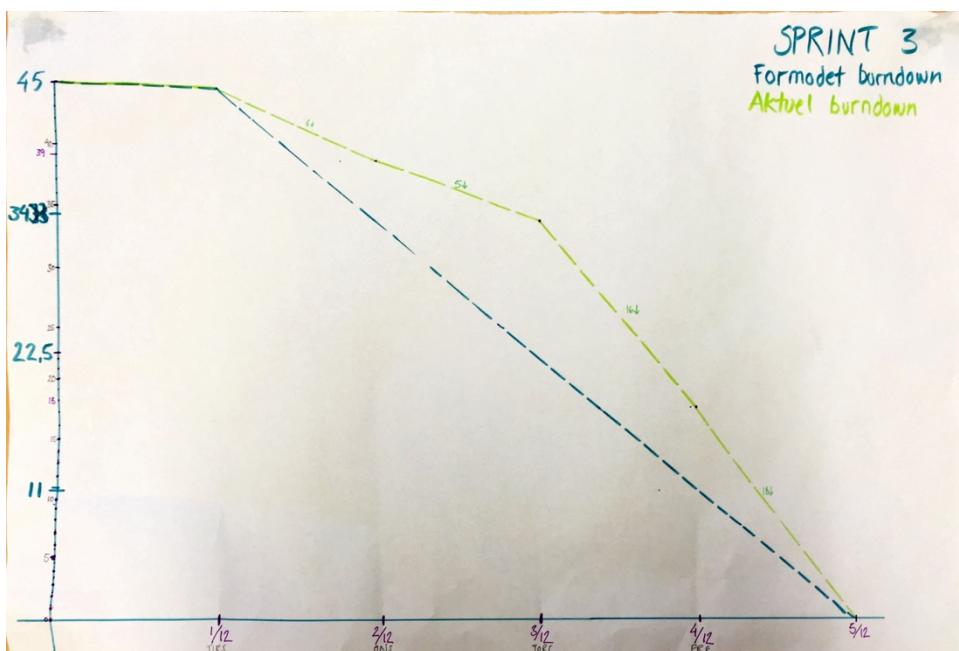
Sprint 3

1/12 - 4/12

Ved opstarten til 3. sprint stod vi i samme problem som tidligere i forhold til at tage userstories ind. Da sprintene kun er på 4 dage og vi helst ikke skal arbejde "oven i hinanden", er det vanskeligt at tage et realistisk antal point ind. De vigtige userstories fylder mange point og kan sagtens fylde et helt sprint alene, men så vil vi netop sidde i flaskehals fra start til slut. Vi ved også at vi bliver nødt til at komme rigtig langt med de komplekse studiemæssige opgaver på dette sprint, for at være sikre på, at vi kan nå i land med dem. Samtidig er vi presset af, at algoritmedelen blev vanskeligere end forventet i sidste sprint og har måttet overføre 12 point til dette sprint. Altså vælger vi at lægge unrealistisk mange point på dette sprint, både for at have forskellige opgaver at arbejde på og for at presse os selv lidt.

Vi bruger også en del af planlægningen på, at diskutere hvordan præcis, vi ønsker at implementere samtidhedshåndteringen. Det er en nødvendig samtale, for at få brudt den pågældende userstory ned, til korrekte tasks der er nemmere at tilgå.

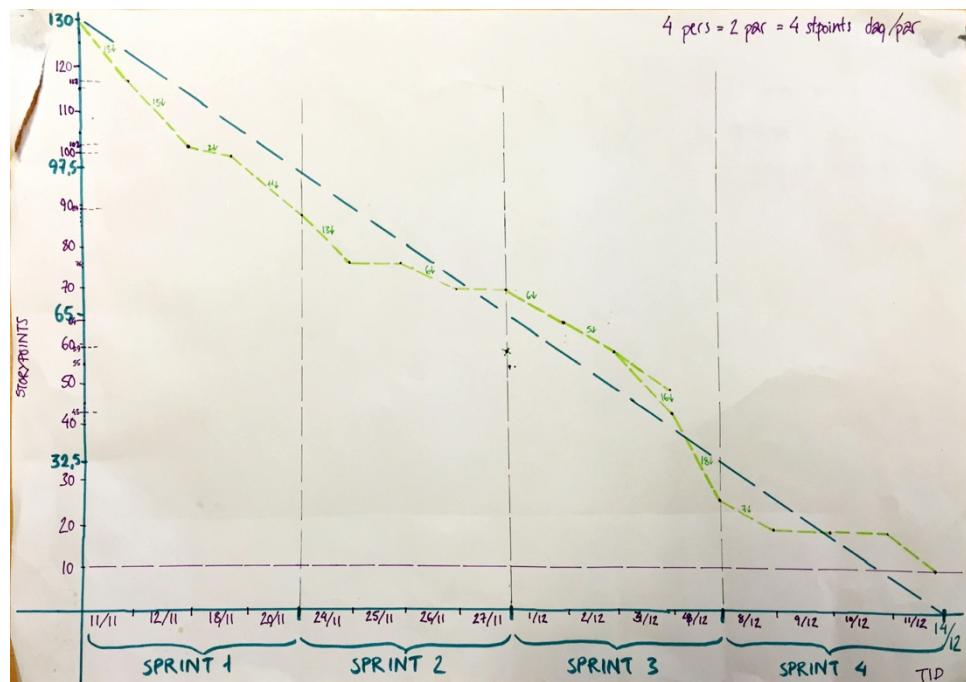
Dag 1 i sprintet arbejder det ene par på gæld og en userstory angående at en gruppe kan leje et grupperum på en bestemt dag. Det tager overaskende lang tid i forhold til de point der er sat af til det, indtil de opdager, på dag 2, at de faktisk er i gang med en userstory, som er en grad vanskeligere, nemlig at en gruppe kan leje et grupperum fra én dato til en anden. At forvente man ved hvad userstoriens omhandler, uden helt præcist at kunne adskille den fra de andre, er en fejl vi også tidligere har begået. Det strider med xp-reglen om at arbejde på small releases og simple design, men det har dog ikke ødelagt noget for os. Dette mere avancerede krav skulle alligevel implementeres i dette sprint. Til gengæld er kanbanboardet noget fyldt med tasks der er i gang, og det bliver lidt uoverskueligt. Som det ses af burndownchart nedenfor kom vi faktisk i mål med de mange point. Grunden til at vi gjorde det, skyldes sandsynligvis også at vi arbejdede en del individuelt for at brænde point af. Uden tvivl mistede vi meget af den feedback og den fælles indsigt som man får af parprogrammering, men presset af tid, var det er valg vi foretog.



FIGUR 8 - BURNDOWNCHART FOR SPRINT 3. 45 POINT BRÆNDT NED.

Sprint 4

Sprint 4 forløb i perioden fra d. 8/12 til den 12/12. I dette sprint undlod vi at tage nye backlogs ind, men valgte i stedet at fokusere på vores allerede fungerende system.



FIGUR 9 - BURNDOWNCHART FOR ALLE 4 SPRINT.

Da vi skulle vurdere, hvad der skulle bruges tid på i sprint 4, følte vi os tvunget til at opprioritere rapportskriving. Vi valgte at hæve bundlinjen, så det skulle passe med de 15 point vi tilegnede diverse tekniske gældsopgaver. Vi nøjedes altså med at beholde de gældsopgaver som var nødvendige for at få de allerede implementerede dele af programmet helt op at køre. Resten af de oprindelige opgaver blev lagt væk til potentielle fremtidige sprints.

Set i et retrospektiv, skulle rapportskriving have fundet sted parallelt med programmeringen, for at undgå et sprint fyldt med rapportskriving og teknisk gæld. I forhold til at benytte retrospektiv til kommende sprints, kan vi ikke benytte sprint 4 i så stor en grad som de forrige sprints. Dog har sprint 4 givet os et indblik i, hvor vigtigt det er at skrive rapport parallelt med programmeringen. Selvom sprint 4 ikke kan gavne os i kommende sprints i dette projekt, kan vi overdrage den viden vi har tilegnet os, til at lave et endnu bedre nyt projekt.

Konklusion

Efter at have arbejdet i en agil proces er det vores konklusion at det er lykkedes os at implementere gode og brugbare elementer fra Scrum såvel som xp og bruge Kanbanboardet og at det tilsammen forbedrede vores proces, højnede kvaliteten og den fælles forståelse. Vi konkluderer at sprint af så kort varighed dog er vanskelige at få fuldt sprintudbytte af.

Kildehenvisninger

www.ekstrabladet.dk

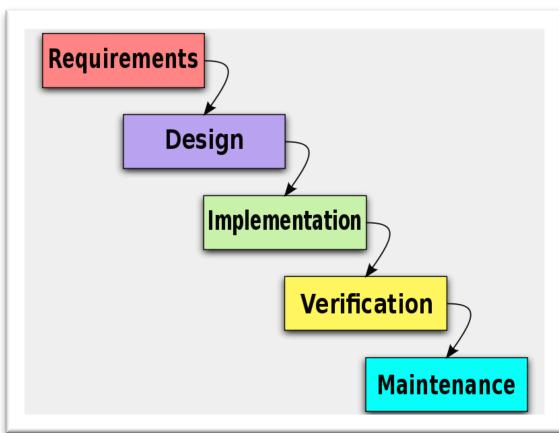
www.5starhvar.com

www.wikimedia.org

www.mcpa.biz

Bilag

Bilag 1: Vandfaldsmodellen



Bilag 2: Noter fra møde med servicedisk

Særligt 2 problemer: 1) Studerende har ikke adgang til at se lokalebookningen. 2) problem at studerende ikke selv kan booke.

Ikke så stort et problem med, om elever ikke er i de lokaler de har booket, men når det hænder er det virkelig generende, fordi der er så stor mangel på lokaler.

Studerende booker alle lokaler om mandagen fra kl. 7 og så har de der kommer lidt senere ikke mulighed for at booke noget i indeværende uge.

Klasselokaler har serviceudlejning ikke noget at gøre med, men kun gruppelokaler. Der er et problem med at studerende roder i lokalerne og ikke sætter stole på plads, derfor er klasselokaler fredet.

Mandag morgen har de 20-25 henvendelser for at få et lokale. Resten af ugen har de ikke rigtigt grupperum at tilbyde. De får ca. 100 henvendelser om grupperum om ugen.

Bilag 3: Personas



Navn: Mads Petersen

Alder: 55

Job: Servicemedarbejder hos UCN

Interesser: Havearbejde

Om: Arbejder i servicelinien hos UCN og er glad for sit arbejde. Han er smilende og social og kan bedst lide de dage, hvor han føler at han gør en forskel for andre. Bliver irritabel over konflikter og klager når han føler det er mangel på anerkendelse af hans indsats.

Scenarie:

Mads oplever problemer med lokalerne, da de bliver lejet ud mandag, og grupper der vil låne resten af ugen kan ikke. Eleverne bliver sure, og Mads synes at dette er træls, og ønsker en løsning. Dog kan han ikke gøre noget ved det selv, andet end at sige det videre.

Med appen kan eleverne selv styrer udlejningen af grupperum.



Navn: Jacob Jensen

Alder: 43

Job: Underviser hos UCN

Interesser: Teknik, nyvindinger indenfor edb.

Om: Jacob har været underviser på UCN i 4 år. Før det arbejdede han for Columbine, men ærgede sig for ofte over dårlig planlægning, som resulterede i stressfyldt arbejdssdag. Han nyder at lærerjobbet giver ham mere overblik over egne opgaver.

Scenarie:

Jacob opfordrer sine elever til at bruge lokalet som grupperum, hvor de kan hænge deres projekt ting op, sådan de får et visuelt billede af hvor langt de er kommet, desuden kan han følge med i processen. Men klasselokalet bliver lånt ud til en eller flere vandreklasser når hans klasse har projekt timer.

Med programmet har UCN færre problemer med vandreklasser, da planlægningen er blevet optimeret.



Navn: Erik Høj

Alder: 25

Job: Studerende hos UCN

Interesser: Laborant.

Om: Har langt i skole og har kun mulighed for at komme til skole med bus. Er glad for dette studie, men er før påbegyndt 2 andre studier som ikke lige var ham.

Noget der generer ham er, at hans uddannelse er baseret på projekter og gruppearbejde, men mulighederne for at få et gruppelokale er alt for begrænset og kun muligt for dem der kan stå på skolen som de første 15 kl 7 mandag morgen. Altså oplever han at de fysiske rammer og mulighederne for at få del i dem er for begrænsede og det sænker kvaliteten af uddannelsen.

Scenarie:

Eriks bus er forsinket og han kan se han vil komme for sent i skole. Han skal bruge et lokale til sin gruppe så de kan arbejde på deres projekt. Han kan nu logge ind med sin telefon og booke et lokale uden at skulle stå i kø når han når frem, og risikere at alle lokaler er taget.



Navn: Rasmus Aggerhøj

Alder: 23

Job: Læser til lærer

Interesser: Pædagogik og friluftsliv.

Om: Rasmus er frivillig spejderleder der nyder at have styr på tingene. Han er glad for børn og glæder sig til at blive folkeskolelærer.

Scenarie:

Rasmus skal ofte bruge et lokale til sin gruppe når de laver projekter, men gruppen er blevet enige om at møde senere og tage senere hjem da de så kan være mere effektive. Det betyder at der ikke er nogen fra gruppen der kommer tidligt nok til at book et gruppe rum i projekt perioderne. Derfor kan de nu bruge deres telefon eller computer hjemmefra til at booke et lokale i god tid.