Rejsekortet 2.0

DMAA0212-GRP2, 4. SEMESTERS PROJEKT

Chris Tindbæk, Mads Nielsen, Morten Klim Sørensen

UNIVERSITY COLLEGE NORDJYLLAND | SOFIENDALSVEJ 60, 9200 AALBORG SV

SYNOPSIS

Titel	Rejsekortet 2.0
Projektperiode	2. december – 20. december 2013
Synopsis	Rapporten er en dokumentation for projektforløbet under 4. semester på datamatiker AK uddannelsen, University College Nordjylland. Projektet omhandler en fiktiv case for trafikselskaberne; Rejsekort 2.0 med forbedret brugervenlighed. Der er blevet lagt vægt på selve processen og denne rapport udgør derfor en væsentlig del af projektet.
Kildekode	https://github.com/iammadsnielsen/ucn-4semproject-dm79-group2/
Vejleder	Gunhild Marie Andersen Mogens Holm Iversen
Deltagere	Chris Tindbæk _____ Mads Nielsen _____ Morten Klim Sørensen _____

INDHOLDSFORTEGNELSE

INDLEDNING

I dette projekt har vi arbejdet med systemudvikling. Et af kerneelementerne i forløbet har været læren om den praktiske tilpasning af udviklingsprocesser. Igennem semesteret har vi lært om forskellige procesmetoder, Scrum, XP og Kanban. I projektet har vi valgt at fokusere på brugen af Scrum med inddragelse af udvalgte praktikker fra XP. I projektet har vi arbejdet agilt, for at gøre op med brugen af den plandrevne udviklingsmetode. Den agile metode gør op med de lange designaspekter fra plandrevne udviklingsprocesser, så som UP. Fordelen er, at man hurtigere begynder udviklingen og derved kan kunden se en reel fremgang.

Projektet har været bygget op omkring en case, inspireret af Rejsekortet. Vi ønskede at lave et lignende system, der tillader brugerne nemt at tjekke ind og ud af busserne. Vi ønskede at lave en overskuelig prisudregning i forhold til det nuværende system, samt en masse andre features.

Velkommen til Rejsekortet 2.0!

FORMÅL

Formålet med dette projekt har været at lære at arbejde agilt og følge de agile praktikker. Udover dette har vi forsøgt at opfylde de stillede læringsmål, som kan læses i paragraffen nedenfor.
Gruppen vil i øvrigt forsøge at integrere elementer fra undervisningen på semestret bedst muligt og hvor de passer naturligt ind i projektet.

LÆRINGSMÅL

- At alle i gruppen får højnet deres faglige i niveau i projektfasen. Det er et mål at styrke vores evner til at arbejde i gruppesammenhæng omkring løsning af komplekse IT-opgaver.
- At alle i gruppen får erfaring med at arbejde med agile procesmetoder.
- At alle i gruppen kan udvælge og bruge praktikker fra XP hvor det er nødvendigt for at løse en konkret problemstilling.
- At alle i gruppen har mod til at drøfte tekniske problemer med de andre gruppemedlemmer og at alle har respekt for hinanden som studerende.

KONKLUSION

Igennem projektet har vi arbejdet med de agile udviklingsmetoder Scrum og XP. Det var været lærerigt og vi føler os godt beredt på at bruge lignende metoder i fremtidige projekter. Vi nåede det vi havde sat som mål og vi er glade for resultatet. Vi er positive omkring processen og det endelige produkt.

PRODUKTVISION

Vi har herunder beskrevet hvordan vi bar os ad, med at starte projektet. Herunder idé generering, kortlægning af målgrupper, samt en produktvision, der sammenfatter produktets formål.

IDÉ GENERERING

Som det første skridt i ide generering skal ideen formes. Man starter med at tænke på problemer i den virkelige verden som man vil løse. Det kan ikke være ligetil at komme på en god ide med det samme, derfor kan man hjælpe det på vej med forskellige stimuli. Billeder eller ord som kan skabe inspiration eller få hjerne til at tænke i kreative baner. Det er vigtigt at man på dette stadie ikke sætter sig tungt om en bestemt idé, men blot skriver alle indfald ned på et stykke papir.

Næste fase handler om at sortere i de ideer man har fået. De mest lovende ideer bliver udvalgt og man prøver at bygge videre på dem, med hjælp fra såkaldte ”Hvad sker der så” scenarier. Hvor man kommer på alle de features som systemet kunne indeholde. Igen er opgaven ikke at begrænse systemet, men blot komme med ideer, også selvom de ikke ender med at være i det endelige produkt. Det er en prøve i den kreative proces.

Når man så har en færdig ide, bør man teste hvad andre personer synes om den. Ideen kan være nok så god, men hvis der ikke er nogen kundeinteresse, vil ideen aldrig blive profitabel.

MÅLGRUPPE

Rejsekortet 2.0 er tiltænkt at erstatte det eksisterende rejsekort. Kortet skal derfor ramme de samme målgrupper og befolkningssegmenter som det eksisterende rejsekort. Den overordnede målgruppe er pendlere. Pendlere indfatter alle rejsende som er afhængige af offentlig transport for at nå deres mål. Mindre segmenter er indeholdt heri blandt andre; studerende, ældre og erhvervspendlere. Hver af disse segmenter har generelt forskellige interesser i systemet. De ældre vil vurdere brugervenlighed meget højt i det nye system. De studerende vil bruge det på grund af systemet billige brugspris. Mens erhvervspendlere søger systemet pålidelighed.

PRODUKTVISION

Rejsekortet 2.0 er til folk, der pendler hver dag og ønsker en nemmere og mere pålidelig måde at betale for dine rejser. Med Rejsekortet 2.0 slipper du for manuelt at tjekke ind og ud af bussen, det er blot nødvendigt at du har kortet på dig, i modsætning det nuværende kort. Vores produkt er stabilt, billigt, personligt og forsøger at sikre den optimale komfort og rejsesikkerhed.

I modsætning til Rejsekortet, kan du ikke glemme at tjekke ud, hvorved du slipper for dumme bøder.

SAMMENLIGNING AF VIRKSOMHEDERS AGILE TILGANG (NYKREDIT VS. COMBINE)

I forbindelse med undervisningen, var vi ude i 2 virksomheder og interviewe dem om, hvad og hvordan de brugte agil udvikling. Vi har herunder beskrevet og sammenlignet udviklings metoder i Nykredit og Combine.

METODEVALG OG PROCESFORBEDRINGER

Nykredit foretager en risikovurdering ved opstart af alle deres projekter, for at vurdere hvor de skal være særligt opmærksomme i udviklingen, men alle deres projekter bliver kørt ud fra deres "use case-component based development". Dette er Nykredits egen fortolkning af UP, hvor de har inddraget agile aspekter i udviklingen.

Combine laver derimod ingen risikovurdering inden et projekts start, men foretager derimod en vurdering af projektet ud fra dets størrelse/varighed, kundens forhåndsviden (kunden ved præcis hvad de vil have lavet) og projekttypen.

Ved Combine bliver der ud fra en vurdering af projektet, besluttet hvilken udviklingsmetode der vælges. Der bliver brugt Scrum på større projekter og plandreven på dem hvor kunden ønsker det. På mindre projekter af få dages varighed bliver der hverken brugt en agil eller plandreven udviklingsmetode.

Vi mener forskellighederne i virksomhedernes fremgangsmåde, bunder ud i den type af opgaver de 2 virksomheder løser. Nykredit løser opgaver for hele koncernen og fejl i disse er meget dyre. I Combine er det typisk opgaver der er mindre kritiske og fejl ikke berøre så mange kunder eller kan føre til økonomiske tab.

Reviews bliver brugt i hver af virksomhederne. Ved Nykredit bliver det gjort ved slutningen af hvert sprint, hvorimod ved Combine er det mere løst hvornår der bliver lavet reviews. Ved Combine kan det ske ved start af et sprint, men også midt i. De har personer med fra flere afdelinger, når der laves review med kunden og kunden kan også selv foretage test af systemet undervejs i et sprint.

Ved Nykredits reviews er der også personer med fra flere afdelinger, dette er for at sikre der bliver lavet kvalitetssikring af et review og dermed også se om der er noget der kan forbedres der til næste gang.

PLANLÆGNING OG KVALITET

For at sikre en god kvalitet i projekterne sammensætter Nykredit deres teams på tværs af organisationen, hvilket er muligt pga. at de arbejder i en matrix-organisation. De bruger planning poker til at estimere de forskellige opgaver. De har erfarne udviklere som gør dette, for alle holdene. Ved Combine benytter de ikke nogen værktøjer til estimering, men snakker i stedet om opgaverne og regner i timer i stedet for story points. Hold bliver sammensat efter personlighed og ikke så meget ud fra kvalifikationer. Det er typisk de samme teams, der arbejder sammen og de kan godt være på flere opgaver afgangen.

I forhold til deres kvalitet, vil Nykredit hellere justere på tiden og prisen af projektet end at justere på feature listen. De har dog erkendt over for os at de gerne vil lære også at justere på features.

Nykredit går meget op i kvaliteten af deres produkter og dette bliver både testet internt og med eksterne repræsentanter. De afholder også reviews på store projekter som udelukkende har kvalitetssikring som prioritet. Nykredit forsøger også at måle på de ikke-funktionelle krav. De har en person i afdelingerne som skal forsøge at vurdere værdien af et nyt stykke software, ud fra hvilken indflydelse han ser det har i afdelingen.

Combine har en anden tilgang til at sikre kvaliteten, ved at hver delopgave/story har en sparringspartner, som de drøfter alt med omkring hvordan opgaven skal løses og hvis der skulle opstår problemer undervejs. Der bliver ikke brugt par-programmering eller test-first, men der bliver lavet test under hvert sprint. Det gør der derimod i Nykredit, dog er par-programmering mest brugt i forbindelse med optræning af nye ansatte.

DEPLOYMENT OG KONFIGURATIONSSTYRING

Nykredit arbejder i et .NET miljø og har derfor valgt at bruge Microsofts versionsstyring TFS (Team Foundation Server).

Hos Combine, bruger de 3 forskellige, afhængigt af udviklingsmiljøet. De bruger både SVN, Git og TFS.

Combine har deres eget staging miljø, som IT-afdeling står for. Der bliver sendt en “pakke” afsted til IT-afd. som så står for at teste og udgive det nye release.

KRAV- OG ÆNDRINGSSTYRING

Nykredit bruger use cases til at udspecificere deres krav til et stykke funktionalitet. Deres krav fastlægges før udviklingen starter.

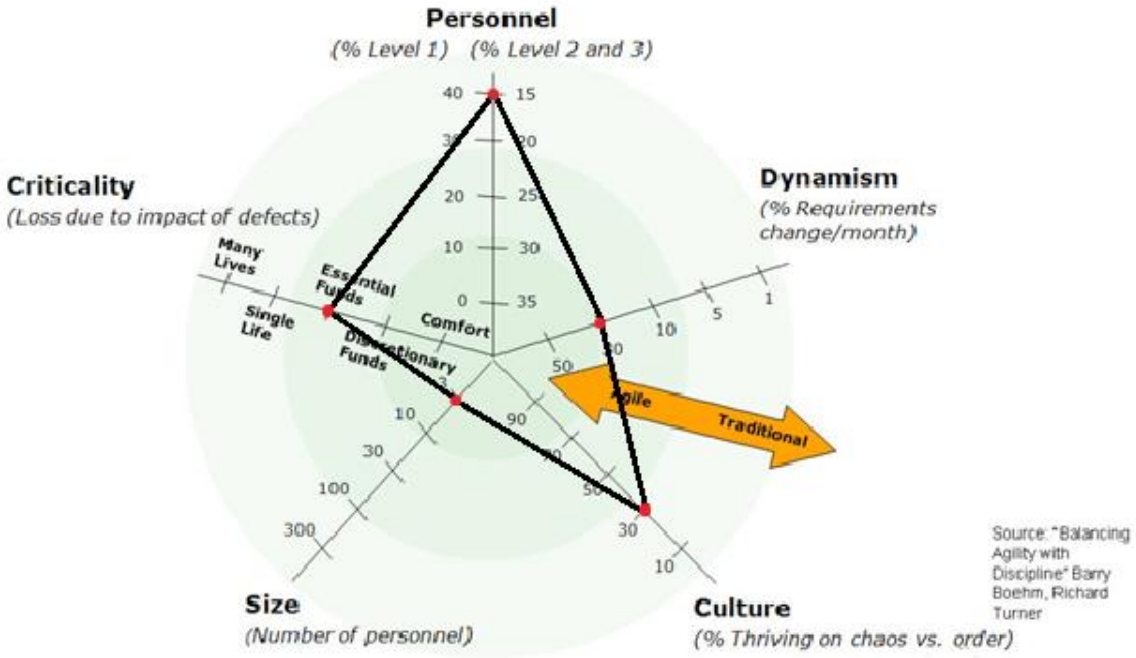
Combine laver ikke altid kravspecifikation, men gør det efter behov. En sælger samarbejder sammen med en projektleder eller en teknisk person, omkring at specificere kravene ude ved kunden.

METODEVALG OG RISIKOSTYRING

I dette afsnit vil vi se nærmere på overvejelser omkring metodevalg og risikostyring i forbindelse med projektet. I dette projekt er det allerede bestemt, at vi skal bruge Scrum. Derfor vil metodevalget, fokusere mere på, hvilke områder gruppen skal være ekstra opmærksomme på i projektet.

PROJEKTKORT

Ud fra projektkortet kan man se at især gruppens kultur og personale-erfaring er kritiske punkter, som man bør prøve at af hjælpe ved at tilpasse arbejdsformen derefter.



KULTUR

Kulturen i gruppen er vurderet til 30%. Gruppen foretrækker ordnede forhold. Vi laver glædeligt dokumentation hvis det opfattes som afklarende eller nødvendigt. Før opgaver startes fortrækker hvert medlem at vide hvad funktionaliteten indeholder. Hvis der er for meget kaos i gruppen kan der opstå uenighed omkring opgavens indhold. Dette er én af de kritiske faktorer i gruppen i forhold til at arbejde agilt, da præferencerne ligger mod det plandrevne. Det er derfor vigtigt at vi i gruppen bruger tid på at afklare opgavernes funktionalitet og indhold inden påbegyndelsen af opgaven. Kulturen i gruppen gør også, at der er en smule modvilje imod at følge princippet omkring refactoring. Da der bliver lagt op til at refactoring bruges til at tilpasse funktionaliteten undervejs. Gruppen foretrækker i store træk at færdiggøre opgaver, i det omfang det er muligt, før de går videre til en ny.

DYNAMIK

Dynamik er blevet vurderet til 30. Der kommer en del ændringer i løbet af projektperioden. Dette skyldes til dels, at ideen er ny og vi arbejder løbende med at udvikle konceptet. Der vil desuden komme en række problemer, hvor vi vil blive nødt til at ændre vores oprindelige planer. Dette er et godt argument for at bruge agil udvikling. Det ville være meget svært og tidskrævende, hvis man skulle have op imod 30 ændringer i løbet af et projekt med en plandreven tilgang.

PERSONALE

Personalets erfaring ligger generelt på et lavt niveau – eftersom vi stadig er studerende. Det er derfor ikke underligt at Boehm's projektkort giver anledning til at vurdere plandreven udvikling som det smarteste alternativ. Der skal hertil siges, at der er tale om et projekt under uddannelsen, hvor ved formålet er at højne dette niveau.

KONSEKVENSER AF FEJL

Konsekvenser af fejl er vurderet til: essential funds. Systemet skal være den drivende faktor til den offentlige transport sektors indkomst, hvor systemet skal være pålideligt nok til at det ikke går ned. Der er ikke tale om menneskeliv, men man skal stadig overveje hvilke dele af systemet der skal have en streng dokumentation for at undgå fejl.

VURDERING AF PROJEKTKORT

Igennem dette forløb skal der arbejdes agilt. Projektkortet har vist, at dette sagtens er muligt. Der skal dog lægges vægt på at kulturen ikke bliver for kaotisk. I løbet af planlægningen af backloggen og opgaverne, skal der være en sund diskussion omkring opgavernes indhold. Sådan at man ikke står med en opgave som skal laves om 3 gange. Tages der hånd om dette bør der ikke være noget der står i vejen for et vellykket forløb.

STØRRELSE

Der er 3 medlemmer af udviklingsgruppen. Størrelsen gør agil udvikling ideel, da den interne koordination, kan ordnes indbyrdes. Mængden af planlægning der skal bruges til at holde styr på medlemmernes arbejde er så lille at de frie rammer inden for agil udvikling er optimal.

RISIKOANALYSE

Der er blevet lavet en risikoanalyse i forbindelse med opstarten af projektet som kan findes i bilagene.
Ud fra denne analyse har vi opstillet en række af de problemer som vi kan risikere at støde på. Hvert af disse problemer er blevet vurderet i forhold til vigtighed og sandsynlighed.

ID	Dato	Beskrivelse	Sandsynlighed	Effekt	Ranking
1	25/11-2013	Kommunikationen med MessagePack fungerer ikke som tiltænkt.	5	10	50
2	25/11-2013	C kan volde problemer, da størstedelen af gruppen ikke er kompetente deri.	4	10	40
3	25/11-2013	RFID skaber problemer ved integrationen, grundet det er et ukendt element for gruppen.	2	10	20
4	25/11-2013	Sygdom eller uventet fravær skaber problemer for den fælles kodeforståelse.	1	5	5
5	25/11-2013	Programmets kompleksitet gør, at programmørerne arbejder langsommere end forventet.	9	6	54
6	25/11-2013	Designet er for simpelt til at alle har samme forståelse.	3	3	9
7	25/11-2013	Gruppens manglende erfaring med de agile udviklingsmetoder, gør processen langsom.	2	3	6
8	25/11-2013	Nye ideer trækker udviklingen i langdrag.	10	1	10
9	25/11-2013	Brede udviklings præferencer sænker release hastigheden.	5	5	25
10	25/11-2013	Uklare krav skal løses hen af vejen.	8	3	24
11	25/11-2013	Mangel på specialister, gør udviklingen langsommelig og besværlig.	1	1	1
12	25/11-2013	Tidsplanen er for optimistisk.	10	2	20
13	29/11-2013	Problemer med underleverandør.	10	4	40

LØSNINGSFORSLAG

Ud fra risikoanalysen, skulle vi prøve at opstille retningslinjer, eller løsningsforslag til problemerne således, at vi kunne mindske sandsynligheden for, at de skete. De risici med den største ranking blev valgt, da disse var vurderet mest kritiske.

- **ID 5** Der skal være klare design og programmering retningslinjer. Der skal desuden ligges kræfter i at undersøge særligt problematiske områder før projektstart i en eller flere spikes.
- **ID 1** Der skal undersøges hvordan kommunikationen mellem C og C# fungere. MessagePack er det valgte klasse-bibliotek og det skal undersøges i en spike.
- **ID 2** Der kan laves små programmer i C, med parprogrammering for at udbrede viden derom til de andre medlemmer af gruppen.
- **ID 9** Stories skal estimeres til at tage højde for en brede udviklings præferencer under planning poker.
- **ID 10** Før projekt start skal der bruges kræfter på at klargøre programkravene i en spike.
- **ID 13** Da der var spikes tilknyttet de komponenter som underleverandøren skulle have leveret, disse indgår nu i sprint 1. Dette betyder, at vi skal revurdere vores estimater og tidsplan. Vi udelader nu spikes omkring RFID læseren helt, indtil vi får besked omkring leveringsdatoen.

KVALITETSKRITERIER OG ARKITEKTUR

Vi har udvalgt kvalitetskriterier som vi har vurderet som værende meget vigtige for vores systems succes. Disse er udspecificeret nedenfor:

- Reliability

  Systemet håndterer finansielle midler hos kunderne – det er derfor altafgørende at brugeren har en oplevelse af et stabilt og sikkert system.
- Usability

  Systemet skal være brugervenligt og nemt at lære at bruge. Dette er en drivende faktor i systemets succes, i forhold til det nuværende rejsekort.
- Security

  Sikkerhed er vigtigt, da kunderne skal føle sig sikre. Deres data må ikke falde i de forkerte hænder.
- Efficiency

  Systemet skal være hurtigt og effektivt, da det ikke er acceptabelt hvis systemet bruger 5 minutter på at tjekke en person ind.
- Testability

  Systemet skal kunne testes efter fejl, så det endelige produkt er så fejlfrit som muligt og eventuelle fejl efter release kan spottes og løses hurtigt.

KVALITETSCENARIER

Vi har ud fra vores udvalgte kvalitetskriterier lavet kvalitetsscenarier, som skal hjælpe os med at opstille opgaver i backloggen som kan sikre at disse scenarier bliver overholdt. Dette gøres i forsøget på at garantere det endelige systems kvalitet i forhold til hvad der er blevet vurderet som vigtigt.

Factor	Measures and Quality scenarios	Variability	Impact of factor on stakeholders and architecture	Priority for success	Difficulty or risk
Reliability					
Pålidelig data.	Når en bruger vil se de registrerede rejser, skal disse være korrekte.	Såfremt man ser sin oversigt lige efter endt rejse, kan der gå tid inden oversigten opdateres.	Det er vigtigt for brugerne, at deres data er korrekt og pålidelig.	H	M
Usability					
Brugervenligt design.	Det skal være nemt for en bruger at navigere i systemet og tilgå oplysninger.	N/A	Det vil påvirke populariteten af programmet hvis det er nemt og intuitivt.	H	L

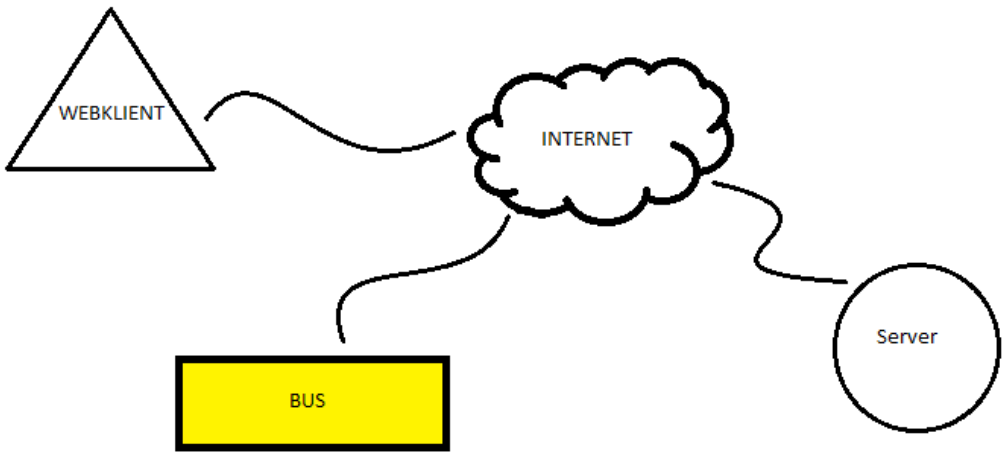
Security					
Kryptering og autentificering.	Brugernes oplysninger bliver krypteret så de er sikret imod sikkerhedsbrud.	Simple uvigtige attributter vil ikke blive krypteret.	Brugerne føler sig trykke, selv hvis der er sikkerhedsbrud.	M	M
Efficiency					
Hurtig tjek-ind og tjek-ud.	Systemet skal udvikles til at muliggøre nem rejse uden besvær.	Systemet skal signalere hvis tjek-ud ikke virker.	Brugerne skal ikke være i tvivl om de er tjekket korrekt ind/ud.	M	L
Testability					
Systemet skal testes.	Systemet skal udvikles til nemt at udføre test, så de overstående scenarier sikres.	N/A	Arkitekturen skal velegnet til test, både på grund af debugging og vedligeholdelse.	L	L

ARKITEKTUR

Vi har valgt at udstyre de offentlige transportmidler med nye RFID læsere, da de nuværende ikke ville kunne opfylde de krav vi har opstillet for brugervenligheden i systemet.

Kommunikationen mellem de offentlige transportmidler, har vi valgt skal foregå i et privat netværk koblet til internettet, over en krypteret forbindelse. Det er vigtigt at kommunikationen sker real-time, så brugerne altid har adgang til de nyeste oplysninger på webklienten.

Dette er valgt, da der med det nuværende system nemt kan skabes inkonsistens mellem de data der vises for brugeren og for hvad selskabet bag rejsekortet kan se i deres system.



AGILE UDVIKLINGSMETODER

Agile udvikling, er et alternativ til den traditionelle projektledelse – hvis navn i sig selv, er en samlet betegnelse for en række iterative og trinvise metoder. I sammenligning med traditionelle metoder, er de agile metoder rettet mod komplekse projekter hvor stabile planer samt forudsigelser ofte er svære at determinere i de tidlige faser.

Der findes en lang liste af metoder og frameworks, som betragtes at være agile, dog vil dette afsnit kun kort koncentrerer sig om kerneværdierne, praktikker ol. på følgende: Extreme Programming (XP), Scrum og Kanban.

EXTREME PROGRAMMING (XP)

XP er en udviklingsmetodik, der kan spores tilbage til 1996. Metodikken er beregnet til at kunne forbedre software kvaliteten og reagere positivt på ændringer i kundes behov. XP bygger på hyppige udgivelser i korte udviklingsforløb, hvilket er til for at forbedre produktiviteten samt indføre "check-points", hvor nye kundekrav kan indføres.

XP har dog fået flere kritiske bemærkninger, hovedsageligt på områder omkring manglende krav samt design specifikationer og generel dokumentation.

DE 5 KERNEVÆRDIER

- Kommunikation
  Softwareudvikling kræver at systemkrav bliver formidlet ud til udviklerne. I en formel metodik, opnås dette via dokumentation.
- Simpelhed
  Der opfordres til at begynde med de mest enkelte løsninger, ekstra funktionalitet kan tilføjes senere hen. Dette begrundes med at holde fokus på de nuværende behov og ikke fremtidige.
- Tilbagemelding
  Der findes 3 typer af tilbagemeldinger i XP; fra systemet (unittest), fra kunden (funktionstest) og fra holdet (ved f.eks. introduktionen af nye krav fra kunden).
- Mod
  Giver udviklere mulighed, for at føle sig godt tilpas ved f.eks. refactoring af kode når nødvendigt, fjernelse af unødvendig eller forældet kode.
- Respekt
  Der skal aldrig tilføjes nyt eller ændret kode, som bryder kompileringen, da dette vil resultere i en forsinkelse hos de andre medlemmer af holdet.

DE 12 PRAKTIKKER

- Tilbagemelding
    - Par-programmering

      Er en teknik, hvori to udviklere arbejder fysisk sammen. En skriver koden, den anden observerer samt gennemgår koden linje for linje.
    - Planlægnings spil

      Planlægningen er opdelte i 2 dele, begge indeholdende: udforsknings, engagement samt en styrings fase.
        - *Udgivelses planlægning, har fokus på hvilke krav der skal inkluderes i de specifikke udgivelser samt hvornår disse skal leveres.*
        - Iterations planlægning, er tilegnet udviklerne – og indeholder hvilke aktiviteter og stories for det enkelte medlem.
    - Test-drevet udvikling

      Går ud på at udvikle en test, som definerer den ønskede funktionalitet – for derefter at udvikle denne med de færreste antal linjer af kode, der skal til for at bestå testen.
    - Hele holdet

      Består ikke kun af udviklingsholdet bag, men også kunden. Dette skal forstås som, at kunden skal stå til rådighed gennem hele processen, til at svare på eventuelle spørgsmål der skulle opstå.
- Kontinuerlig proces
    - Kontinuerlig integration

      Er praksissen med, at integrerer nyt eller ændret funktionalitet med kodens eksisterende kildekode repository.
    - Refactoring

      Er en teknik, til omstrukturering eller generelle ændringer af eksisterende kode, uden at ændre dennes adfærd.
    - Små udgivelser

      Leveringen af projektet, sker i hyppige udgivelser. Disse små udgivelser, hjælper kunden med at få tillid til udviklingen af projektet.
- Fælles forståelse
    - Kodningsstandarder

      Er et aftalt regelsæt, som udviklingsholdet er enig om at holde sig til – gennem hele projektet.
    - Kollektiv kode ejerskab

      Har den betydning, at alle er ansvarlig for al kode – hvilket betyder at alle ret til at ændre hvilken som helst del af koden.

    - Simpelt design

      Udviklere, burde tage den simpleste tilgang til software design. Det samme gør sig gældende ved refactoring af komplekskode.

    - System metafor

      Går ud på, at alle medlemmer af projektet; kunden, udviklere, scrum-master og anden ledelse, kan fortælle hvordan projektet/softwaren fungerer.

- Programmør velfærd
    - Bæredygtigt tempo

      Er et koncept, hvori at udviklerne ikke arbejder mere end de standard 37 timer. Hvis overarbejde er en faktor i den pågældende uge, bør den efterfølgende uge ikke have dette.

SCRUM

Scrum frameworket blev skabt i starten af 90'erne og kan som udgangspunkt betragtes som et system hvori, man har kendskab til hvad der laves – men et detaljeret kendskab til systemet som en helhed, er en ukendt faktor.

Scrum i sig selv, er ikke et fuldendt værktøj, når det gælder produktudvikling – da denne ikke er en del af frameworket. Ved produktudvikling, kan f.eks. ScrumBot benyttes – da dette giver mulighed for at tilpasse metodikken, til at passe til det pågældende behov. Foruden kerneværdier og praktikker, består Scrum også af 3 roller og 3 artefakter.

DE 5 KERNEVÆRDIER

- Fokus

  Ved at bevare fokus på kun enkelte stories, vil der være en bedre arbejdsgang på holdet – der vil resultere i en optimering af holdet.

- Mod

  Da der arbejdes i hold, vil der være en følelse af støtte bland medlemmerne samt flere ressourcer til rådighed – hvilket vil give den enkelte større mod, til at påtage sig mere komplekse udfordringer.

- Åbenhed

  Ved at samarbejde i hold, praktisere medlemmerne at udtrykke hvad og hvordan de gør deres arbejde – dette vil resultere i, at det enkelte medlem føler at de kan udtrykke eventuelle bekymringer og disse tages op.

- Engagement

  Jo mere kontrol der overgives til et hold, jo mere engageret bliver medlemmerne.

- Respekt

  Ved at arbejde i hold, deles succeser og fiaskoer – hvilket vil skabe en fælles respekt blandt medlemmerne.

DE 5 KERNEPRAKTIKKER

- Backlog; Produkt, udgivelse samt sprint

  Alle ønsker/krav af funktionalitet for det enkelte produkt, udgivelse og sprints. Produktejer og/eller leder står for udvikling og vedligeholdelse af backlogs.

- Iterativ udvikling

  Hele projektet blev delt ind i sprints, med en fast varighed. Dette er til for at, hjælpe udviklingsholdet bag, med at holde fokus på et leverbart produkt for enden af hvert sprint.

- Scrum møder

  Daglige møder, hvor hvert medlem på holdet svare på følgende: Hvad lavede du i går, eventuelle problemstillinger der skal tages op samt hvilke planer har du for i dag.

- Burndown diagrammer

  Giver en indikation, det enkelte sprints fremskridt. Dette giver også mulighed for at eventuel revurdere holdets velocity til fremtidige sprints.

- Sprint review møde

  Giver en mulighed, for at inspicere fremskridtet efter hvert sprint. Hvis mål er at forbedre udviklingsprocessen ved at indføre nye metoder, ændre eksisterende praksis osv.

DE 3 ROLLER OG 3 ARTEFAKTER

- Roller
  - Produkt ejer

    Repræsenterer aktørerne og er stemmen for kunden. Det er også produkt ejerens ansvar, at tilføje og prioriterer stories til produkt backloggen.
  - Scrum-master

    Er ansvarlig for at fjerne eventuelle forhindringer for holdet, så dette kan levere fastlagte mål for enden af hvert sprint.
  - Udviklingshold

    Holdet er ansvarlig, for levering af et potentielt leverbar produkt, for enden af hvert sprint. Det enkelte medlem er tværfunktionelle og ikke låst fast på et område.
- Artefakter
  - Produkt backlog

    Er en overordnet liste over krav, tiltænkt det enkelte produkt.
  - Sprint backlog

    Er en liste over stories, som udviklingsholdet skal tage sig af i løbet af det næste sprint.
  - Burndown diagram

    Giver en indikation, det enkelte sprints fremskridt og opdateres på et daglig basis.

KANBAN

Kanbans oprindelse stammer tilbage fra 1940'erne, da Toyota ønskede at optimere produktions planlægningen og basere den på efterspørgsel. Kanban metoden, gældende agil software udvikling, blev dog først formuleret af David J. Anderson i 2003. Hvor Kanban metoden skiller sig ud, i forhold til f.eks. Extreme Programming, er ved at ikke være en proces eller en projektledelses metodik for software udvikling – men derimod et tillæg til eksisterende metoder og frameworks. Kanban i sig selv, består af 4 basisprincipper og 6 kernepraktikker.

DE 4 BASISPRINCIPPER

- Start med hvad du ved

  Der startes med de roller og processor som findes i forvejen – hvorefter der er mulighed for at udvide disse, hvis ønsket.
- Enighed om at forsætte trinvis

  Holdet skal acceptere, at trinvise forandringer er vejen til at forbedre systemet.
- Respekter de nuværende processor, roller, ansvarsområder og titler

  Holdet har sandsynligt en opsætning som fungerer og som er værd at bevare.
- Lederskab på alle niveauer

  Der opfordres til lederskab, fra den enkelte medarbejder til den øverste ledelse.

DE 6 KERNEPRAKTIKKER

- Visualiser

  Visualisering af arbejdsgangen og dennes synlighed, er nøglen til at forstå hvordan arbejdet skrider frem. Uden denne forståelse, vil det være svære at tage de rigtige beslutninger.
- Begrænset antal igangværende stories (WIP)

  Ved at begrænse antallet af igangværende stories, undgås unødvendig overbelastning af holdet – frem for at holde en 100% arbejdsbyrde.
- Administrer arbejdsgangen

  Ved en aktiv administration af arbejdsgangen, kan der lettere vurderes om introduktionen af nye ændringer, har en positiv eller negativ virkning på systemet.
- Eksplicitte fremgangsmåder

  Uden en eksplicit forståelse af hvordan en proces virker og hvordan arbejdet faktisk udføres, vil en diskussion ofte være følelsesmæssig, anekdotisk og subjektiv.
- Implementere tilbagemeldinger

  Manglende tilbagemeldinger vil ofte føre til, at optimeringen af arbejdsgangen og dennes byrde er ikke eksisterende for holdet.
- Forbedre fællesskabet

  Et hold med en fælles forståelse for arbejdsgangen, processen og risici, er bedre i stand til at opbygge en fælles forståelse for et problem og forslå rettelser som kan ændre på dette.

BRUG AF UDVIKLINGSMETODE

I dette forløb har vi beskæftiget os med Scrum – med inddragelse af forskellige praktiker fra XP. I det følgende afsnit vil det blive gennemgået hvordan vi har forholdt os til – og brugt de forskellige praktiker igennem de 3 sprints.

PAR-PROGRAMMERING

I stedet for at bruge par-programmering konsekvent, besluttede vi os for at det ville være mere effektivt, at udvælge specifikke områder hvor par-programmering var nødvendig. Vi var enige om det var ikke gav mening af bruge 2 personers dag på at lave én opgave, hvis opgaven kompleksitet ikke var større end man kunne sætte sig ind i den skrevne kode på egen hånd. Vi valgte par-programmering ved kritiske opgaver, hvor kompleksiteten enten var meget høj, eller opgaver som skulle løses hurtigt for at kunne færdiggøre andre opgaver. Dette gjorde vi for at minimere fejl i den komplekse kode og være sikker på alle scenarier blev tænkt igennem.

Par-programmering blev også brugt i tilfælde, hvor vi løb ind i fejl. Her valgte vi at sætte os 2 sammen for at debugge problemet. Dette mener vi var godt givet ud i sidste ende, frem for én blot prøvede at sidde og finde frem til hvad der skyldtes den exception blev smidt.

REFACTORING

Refactoring er blevet brugt flittigt, i løbet af udviklings processen. Det skal dog understreges at vi aktivt gjorde en indsats for udvikle systemet bredt nok til at minimere nødvendigheden af refactoring. Som det blev fastslået i vores risikoanalyse arbejder gruppen bedst under ordnede forhold. Så selvom at story-based-development arbejder med at stories, skal kunne laves uafhængige af hinanden. Har vi i vores arbejde tænkt bredere, i forhold til senere funktionalitet og opgaver. Vores holdning er at refactoring er et smart værktøj, hvis man bruger det rigtigt. Et eksempel på hvor vi har brugt refactoring, er i sprint 1 med graf-servicen. I starten skulle den kun kunne sige hvor mange zoner der er mellem 2 vilkårlige zoner. Hvilket ikke indeholdt den prisberegning, som skulle skrives ind senere. Refactoring er der ud over også blevet brugt til at opfylde kravet om simpelt design.

KOLLEKTIV KODE EJERSKAB

Hvis en person har fundet en fejl, eller synes der manglede et stykke funktionalitet i en del af programmet. Måtte personen gerne selv tilføje / ændre koden. Såfremt det var en fejl man havde fundet, skulle ham der oprindeligt har skrevet koden, hvor fejlen opstod i blot informeres. Hvis det var ny funktionalitet skulle personen der ønskede ændringen, blot sikre at andre i gruppen ikke allerede havde tilføjet dette i et lokal commit.

Det blev også udøvet i databasen, hvor vi alle ændrede kolonner og datatyper. Et andet eksempel er web-klienten, hvor vi alle arbejdede med layout filerne og de enkelte views, i takt med at vores stories krævede integration med web-klienten.

TEST-FIRST

Der er ikke blevet udviklet ud fra test-first princippet i løbet af vores sprints. I stedet for blev der oprettet test-klasser parallelt med en story. Når man havde lavet en metode, der relaterede til storiens funktionalitet, skulle man skrive en test, som sikrede at metodens udformning og logik var korrekt. Igen var grunden til dette at vi synes at man uundgåeligt ville komme til at lave redundante tests hvis man skulle teste alt, for hvert skridt man tog i udviklingsprocessen. Vi ønskede at holde det simpelt og gennemteste større blokke kode af gangen. Udover dette brød vi os ikke om fremgangsmåden med at man skrev en test(s), før man havde logikken man ønskede testen skulle køre på. Dette ville i nogle tilfælde gøre at den test man har skrevet, skal skrives om efter logikken er lavet.

Test-first er egentlig beregnet til at stimulere programmøren til at tænke over hvilke krav der er til hver metode. Vi mente dog godt at kunne håndtere dette uden.

SIMPEL-DESIGN

Vi har forsøgt at følge simpel-design meget hen af vejen. En af de ting vi gjorde var at dele vores struktur over mange lag, så funktionaliteten i det enkelte lag var meget udspecificeret. Web-klienten havde en MVC opbygning. Systemet bagved havde business-logic, som tog sig af alt forretningslogik, beregninger og manipulation af data inden det blev sendt videre og til sidst repository laget, hvor alt database kommunikation lå i.

Et andet tiltag vi gjorde var at bruge refactoring til at skrifte kodestykker om, hvis et medlem af gruppen synes, der var en lettere/simplere måde at lave sammen funktionalitet, i takt med at systemet fik flere features.

1. SPRINT (UGE 49)

Det første sprint strakte sig fra mandag den 2. december til mandag den 9. december 2013. Hovedformålet med dette sprint var at etablere systemstrukturen fra databasen helt op til den grafiske brugergrænseflade, der i vores tilfælde var en hjemmeside. Det var vigtigt for os, at programmets struktur hurtigt blev fastlagt, for at undgå alt for meget refactoring senere.

STORIES

Ugen startede med at vi skulle identificere hvilke stories der skulle laves i sprintet. Opgaverne i sprint 1 var udvalgt til at danne grundlag for den senere funktionalitet i kommende sprint. Herudover ville disse opgaver også forhindre for meget dummy-data i koden. "Opret bruger", "Overføre penge til konto" og "Bruger og administrator roller". Herefter valgte vi 2 stories, der indeholdte kernefunktionaliteten for systemet. Enten i form af sikkerhed i systemet: "Kryptering af følsomme data" eller metoden som skulle stå for at regne ud hvor mange zoner man havde gennemrejst: "Graf-service". Alle stories blev udfyldt på de udleverede ark, som skulle sikre at værdien af en enkelt story var klar for brugeren.

Story Name:			ID:
Opret bruger			1
As a	Bruger.		
I want to	Opret mig på hjemmesiden.		
So that	Jeg kan bestille rejsekort.		
Estimate:	Actual:	Priority:	
3	- ikke angivet -	2	

I denne story ønskede vi at brugeren skulle være i stand til at oprette sig på sig på vores hjemmeside. Systemet skulle have den understøttende funktionalitet til at gemme oplysningerne i databasen. Der skulle være validering af inputfelterne, således at de indtastede oplysninger overholdt de formalia, vi havde stillet (personnumre skal være 10 cifre lange, telefonnumre skal være 8 cifre, mm.). Vi ønskede derudover også automatisk at hente brugerens by ud, når man indtastede sit postnummer. For at gøre dette skulle alle postnumre og byer gemmes i databasen og hente det tilsvarende bynavn ud, når man indtastede postnummeret. Bruger objektet skrækker sig over flere tabeller i databasen. Som en del af denne story skulle der derfor også aftales design af, samt sammenhængen mellem disse tabeller.

Det var oprindeligt ikke tiltænkt, at der skulle par-programmeres på denne story. Men pga. problemer med at få vores ORM til at virke, blev det nødvendigt. Vi kom til et punkt, hvor at det var nødvendigt at få det lavet, for ikke at falde bagud. Vi valgte derfor at sætte ekstra ressourcer på opgaven, for at få løst problemet.

Story Name:			ID:
Kryptering af følsom data			3
As a	Bruger.		
I want to	Vide at mine oplysninger bliver beskyttet.		
So that	Jeg føler mig tryg ved at bruge systemet.		
Estimate:	Actual:	Priority:	
5	- ikke angivet -	3	

Vi ønskede med denne story, at repræsentere sikkerhed i systemet ved at kryptere de mest kritiske personoplysninger i systemet: Personnummeret og password. Der skulle implementeres en klasse, der kunne kryptere brugerens personnummer ved brug af AES-256 krypteringen, samt en klasse der ved hjælp af den dertilhørende nøgle kunne dekryptere koden. Her ud over skulle der laves et ”salted hash” af password.

Story Name:			ID:
Overfør penge til konto			6
As a	Bruger.		
I want to	Indbetale penge på kortet.		
So that	Jeg kan forsætte med at rejse.		
Estimate:	Actual:	Priority:	
2	- ikke angivet -	5	

I denne story ville vi gøre det muligt for brugeren at indbetale penge til rejsekortet. Vi lavede et view på web-klienten, som skulle tage sig af dette. Der skulle være validering på kortnummeret, der bliver brugt (vi gemmer dog ikke kortnummeret). I databasen skulle der oprettes en tabel som holdte styr på alle indbetalingerne. Som en del af det view, der var på web-klienten skulle man også kunne se ens saldo blive live-opdateret. Det vil sige at, der skulle laves en metode, som kunne summere alle indbetalinger og hævninger på en bestemt brugers konto, i forbindelse med en ny indbetaling. Dette blev et eksempel på refactoring, da vi i starten af denne story ikke havde en måde at hæve penge på brugerens konto. Først da vi havde funktionaliteten til at hæve fra en konto blev tilføjet, skrev vi koden om.

Story Name:			ID:
Bruger og administrator roller			9
As a	Administrator.		
I want to	Have andre rettigheder og muligheder i systemet end brugerne.		
So that	Så man kan styre systemet som administrator.		
Estimate:	Actual:	Priority:	
13	- ikke angivet -	1	

Formålet med denne story var at adskille, brugernes login med administratorernes. Der skulle oprettes database tabeller til at håndtere roller i systemet. Når man loggede ind i systemet, skulle systemet undersøge om der var tale om en bruger eller administrator og derefter vise den tilhørende startside.

Story Name:			ID:
Graf-service			14
As a	Administrator.		
I want to	Se hvilke stoppesteder, der høre under hvilken zone.		
So that	Så man kan beregne rejsen korrekt.		
Estimate:	Actual:	Priority:	
40	- ikke angivet -	4	

Dette var den mest komplekse story i sprint 1. Idéen var at vi ville lave en repræsentation at alle zonerne i NTs system ved hjælp af en graf. Ud fra denne graf skulle vi så kunne regne ud hvor mange zoner, der er mellem 2 stoppesteder (tjek-ind og tjek-ud punktet). Hertil skulle vi bruge en masse data omkring alle stoppesteder, zoner og nabozoner. Digitaliseringsstyrelsens API havde disse oplysninger. Vi besluttede derfor at lave en metode til at konvertere JSON outputtet til data som kunne lægges ind i databasen. Da dette var gjort skulle vi have lavet metoden som kunne bestemme afstanden mellem 2 stoppesteder, vi besluttede os for at bruge en version af Dijkstras algoritme uden vægte på kanterne. Vi havde besluttet at bruge par-programmering for at implementere algoritmen, da vi regnede den for at være særdeles kritisk i forhold til vores system. Vi ville hermed både sikre, at der var mere end 1 person med viden omkring den specifikke story, men vi regnede også med at vi ville kunne drage nytte af, at 2 personer arbejdede på opgaven, i forhold til kompleksiteten. Graf-servicen opgaven indeholdte også en del refactoring. I starten skulle metoden til zone-beregningne kun kunne sige hvor mange zoner der er mellem 2 vilkårlige zoner. Hvilket ikke indeholdt den prisberegning som skulle skrives ind senere.

Estimeringen af denne story viste sig at have været pessimistisk. Med 2 sæt øjne på opgaven, lavede vi algoritmen og sikrede os den virkede på omkring 10 timer, hvilket var en del under det forventede.

RESUME

Tirsdag d. 3 december: Opstartsdagen var plaget med problemer. Da vi først startede gik det op for os at vi ikke havde aftalt en konkret mappe struktur til systemet. Det blev overset i sprint 0, hvor vi ellers lige have indlagt en spike til at diskutere dette. Et andet problem som opstod var at vi ikke kunde få driveren til PostgreSQL databasen til at arbejde sammen med vores ORM – entity framework. Det var svært at forudsige, at der ville blive driver problemer, men det ville have været ideelt at teste igennem i et spike. Men det blev ikke vurderet som et sandsynligt problem hvilket er grunden til at vi ikke spikede på det. Dette forårsagede at forbindelsen mellem systemet og databasen ikke kunne oprettes på første dagen, hvilket gjorde at ingen af de i gang satte stories kunne færdiggøres. Vi arbejdede rundt om dette med at en person stod for at designe og bygge databasen, mens de 2 andre lavede business-logic, repository og web til de aktive stories. Brugernes data lå i flere forskellige tabels på databasen, for at undgå null værdier mm. Det blev derfor nødvendigt at have en diskussion om hvordan repositoryklasserne/userlogic skulle samles for at trække de nødvendige oplysninger ud omkring brugeren på en fornuftig måde. Løsningen blev at sammensætte user objektet af flere andre objekter fra de forskellige tabels. Således kunne vi undgå at hente mere data ud end nødvendigt. Der blev desuden oprettet unit-test klasser for stories. I slutningen af dagen var der under tasks, der var færdige, men ikke stories, da forbindelsen til databasen stadig ikke virkede.

Onsdag d. 4 december: Dagen fortsatte med database problemer Morten og Mads satte sig sammen for at prøve at løse problemet, da dette nu sinkede os. I mellemtiden blev der lavet views, model-klasser og mapping klasser klar til databasen. For ikke at falde bagud tog vi nye opgaver ind og prøvede at lave så meget som muligt uden database forbindelse, der med overholdt vi ikke princippet med at man laver en opgave færdig, før man starter på en ny. Men i den situation vi var i virkede det som det bedste alternativ. I slutningen af dagen virkede ORMen stadig ikke – så vi besluttede os for at skifte teknologien ud, så vi kunne komme over problemerne. Vi var dog ikke håbløst bagefter, da vi havde været meget effektive med at trække nye stories ind. Var der mange subtasks, som var færdige. Vi besluttede at lave en vurdering af hvor mange points, der var manglede på de enkelte stories. Vi nedskrev derefter vores burndown ud fra denne vurdering. Selvom det egentligt ikke er meningen, at man skal nedskrive burndown, før man er helt færdig med en opgave, var vi enige om at vores beslutning ville give et mere retvisende billede af vores fremskidt. Vi fik desuden at vide i dag at leverandørerne af vores RFID-chip som vi skulle bruge i projektet, sandsynligvis ikke ville komme. Hvilket gav anledning til at overveje hvordan vi så skulle simulere tjek ind / ud i senere opgaver.

Torsdag d. 5 december: Vi startede dagen med den gode nyhed at der var forbindelse til databasen, hvilket betød at vi kunne færdiggøre flere stories. Morten og Chris satte sig sammen for at pair-programme graf-servicen. Dette viste at være et virkeligt godt valg, da opgaven blev løst meget hurtigere end forventet, med 2 sæt øjne på opgaven. Vi brugte en del af dagen på at hente informationer omkring stoppesteder og zoner fra digitaliseringsstyrelsen, samt konvertere dem fra JSON til brugbar CSV, som kan ligge i databasen.

Fredag d. 6. december: Dagen var præget af internetproblemer efter gårsdagens storm. Vi havde svært ved at udføre de sidste tests som manglede i vores validering af de sidste stories, da vi ikke kunne forbindes til serveren. Dette resulterede i at vi havde et lille hængeparti, som vi ville prøve at få lavet mandag.

Mandag d. 9 december: Dagen gik med sprint-retrospektive, samt estimering og prioritering af det nye sprints stories. Vi valgte, at der hver uge skulle være en ny scrum-master så vi alle prøvede det. I slutningen af dagen havde vi halvanden time til at indhente hængepartiet fra sidste uge. Hvilket blev nået. Så vi kunne gå ind til sprint 2 med en ren tavle.

ESTIMERING

Vi estimerede vores stories ved at gøre brug af planning poker. Efter vi alle havde smidt vores kort havde vi en diskussion, for at argumentere for vores estimater. Herefter prøvede vi at spille vores kort igen. Ofte kom vi til enighed allerede her. Det var sjældent at vores bud adskilte til fra hinanden med mere end et kort, hvilket gjorde at vi forud for diskussionen allerede kunne fornemme at vi havde den samme opfattelse af storien. Vi valgte at prioritere vores stories ud fra sort-by-value princippet og stories som havde stor vigtighed for systemets salgbarhed samt forretningsværdi blev prioriteret over tekniske svære stories.

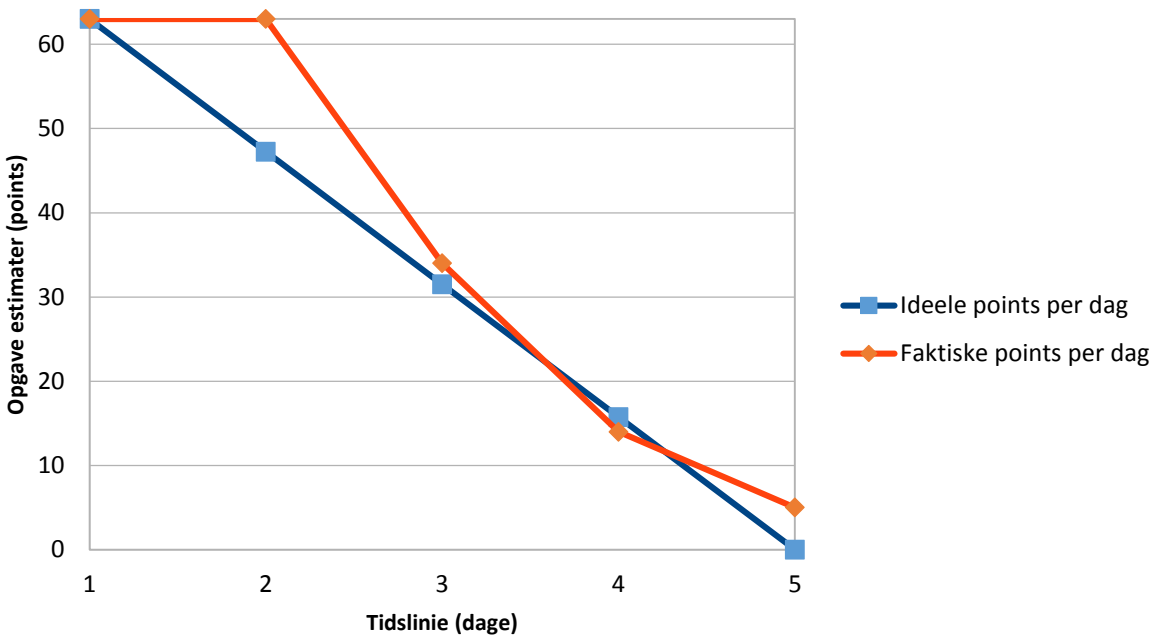
I slutningen af dagen havde vi planlagt hele sprintet: 63 story points, fordelt på 5 stories. Vores daglige velocity blev beregnet til 15 points/dag. Vi mente alle at dette var realistisk, givet vores estimater.

MORGENMØDER

Hver morgen klokken 9:00 holdte gruppen morgenmøde. Formålet med dette var at sikre de andre i gruppen kendte til de andres opgaver, samt hvilke problemer, der havde været i forbindelsen med den konkrete opgave. Herefter skulle den enkelte person redegøre for hvad man havde tænkt sig at lave den efterfølgende dag. Det var på morgenmøderne hvor der internt blev aftalt hvilke opgaver, hvor der var behov for par-programmering og hvordan vi ville teste opgaverne. Vi holdte alle møderne i sprint 1 som stand-up møder, for at sikre at mødet varighed, ikke trak ud, længere end det var nødvendigt.

I slutningen af mødet nedskrev vi vores burndown. Normalt nedskriver man først sin burndown når en opgave er helt færdig. Vi kunne dog ikke lide denne metode, da den ville resultere i en misvisende graf, hvor det ville se ud som om vi altid var bagud. Vi valgte derfor at vurdere hvor langt vi var nået med en story, i slutningen af hver dag. Hvorefter vi nedskrev burndown med et tilsvarende antal points. Vi var alle enige om at denne metode skabte en bedre repræsentation af arbejdet på burndown.

BURNDOWN DIAGRAM



VIDENSDELING

Vidensdeling var i fokus igennem hele forløbet. Vi sikrede at en fornuftig vidensdeling ved at implementere flere forskellige tiltag. Det første punkt var morgenmøderne, hvor man frit kunne spørge ind til en persons arbejde dagen før, hvis man følte at emnet godt måtte blive uddybet for at skabe en bedrer forståelse på tværs af gruppen.

Vi brugte også par-programmering til at skabe en sund vidensdeling. I særlige systemkritiske opgaver besluttede vi, at det var fornuftigt hele tiden at have 2 sæt øjne på opgaven af gangen. Herved opnåede vi at der ikke var en enkelt person, som sad med kritisk viden omkring systemet.

Hvorved arbejdet kunne fortsætte i personens fravær. Det blev i sær brugt på de opgaver som blev betegnet som de mest komplekse og de stories som stod for kernefunktionaliteten f.eks. graf-servicen.

En fordel ved at sidde i en lille gruppen på tre mand og udvikle er, at man ikke holder sig tilbage med at stille spørgsmål. Om det så er tekniske spørgsmål eller afklaring af indholdet af en bestemt opgave. Når man tilretteligger arbejdsmiljøet således, kommer der naturligt også en fælles forståelse af hvad de andre personer i gruppen laver. Dette sætter dog krav til medlemmerne i gruppen. De skal kunne håndtere at have flere bolde i luften af gangen, uden at det distrahere deres arbejde i større grad. Det er en balancegang, som vi dog i sidste ende var enige om fungerede godt.

RETROSPEKTIV

I slutningen af sprintet skulle vi lave et retrospektiv. Her skulle vi kategorisere hvordan vi synes at vi havde brugt de forskellige praktiker fra XP, samt hvordan vi synes arbejdet var gået generelt. Herunder skulle vi også komme med forslag til hvordan processen kunne forbedres til næste sprint.

De gode ting:

- Vi var glade for vores estimering og velocity, da de hang godt sammen. Det lykkedes os at tage en passende mængde stories ind i sprintet, så vi ikke var håbløst bagefter. Vi skulle omvendt heller ikke tage flere opgaver ind i sprintet, så vi har godt tilfredse, selvom vi måske ville have været foran hvis det ikke havde været for opstartsproblemer i starten af sprintet.
- Vores brug af par-programmering var vi også glade for. I stedet for at bruge det hele tiden, havde vi fra starten af sagt at par-programmering skulle bruge selektivt, når der var brug for det. Da det ellers ville sinke vores tempo, i forhold til hvad vi kunne få ud af det.

Ting der kunne være bedre:

- Planlægning af spikes kan forbedres. Vi stødte ind i en masse tidskrævende problemer i sprintet, fordi vi ikke havde været opmærksomme nok på eventuelle tekniske udfordringer.
- Morgenmøderne blev kritiseret for at være for løse, uden nogen fast agenda. Det var derfor ikke alle, der følte, at møderne gav ret meget til forståelsen af de andre gruppemedlemmers arbejde.
- Fælles kodeforståelse.
- Proces dokumentation.

Forbedringsforslag:

- Fast agenda ved morgenmøderne, for at skabe orden og overblik.
- Procesdokumentation skal forbedres, da den mest har bestået i en enkelt logbog.
- Blive bedre til at forud se problemer, som kan løses med spikes.

SPIKES

Vi løb ind i en del problemer løbet af 1. sprint. En del af disse blev foranlediget af at vi ikke have været tilstrækkelige grundige i vores planlægning op til dette sprint. Vi havde en del tekniske problemer med at få vores ORM til at arbejde sammen med den planlagte PostgrSQL database. Dette var et problem som kostede os meget tid, fordi vi ikke havde sat tid af til at afprøve alternativer. Det endte dog med at vi blev nødt til at gå over til en tidligere kendt teknologi, da vi blev presset på tid.

Et andet problem som kunne have været undgået med en spike var vores planlægning af systemets mappe struktur. Da vi skulle til at i gang med at kode, blev spørgsmålet lige pludselig meget relevant, da vi gerne ville starte ud med en fornuftig og gennemtænkt struktur, som kunne bruges igennem alle 3 sprint. Dette kostede også en smule tid.

2. SPRINT (UGE 50)

Følgende resume, for den pågældende uge hvori 2. sprint forløb (fra den 10. til den 13. december 2013), henviser til stories som står beskrevet under 2. Sprint/Stories på side 30 samt 31.

TIRSDAG DEN 10. DECEMBER

Der blev ved dagens stand-up møde introduceret en ny scrum-master samt blev der kigget på de stories som var blevet tildelt dette sprint. Storiesne blev oprindelig estimeret under sprint 0, men da det var gået op for os sprintet før, at disse estimater var forholdsvis pessimistiske – blev disse diskuteret i gruppen. En enkelt storie, tjek ind og ud på rejse, som havde et estimat på 100 – blev delt op i stories 2.1 samt 2.2 samt fik et lavere estimat.

For at undgå, at gruppen løb ind i de samme problemer som der var i det forrige sprint – blev der tilføjet 2 spikes, som gik ud på at kunne oprette netværksforbindelse fra en Arduino Uno til vores server (2.4) samt at kunne opretholde klient/server forbindelse mellem enhederne (2.5). Begge af disse spikes, blev påbegyndt denne dag.

Dagen afsluttede med at spike 2.4 var færdiggjort, trods en estimering på 13 – samt spike 2.5 var omkring 70%, 9/13 points, afsluttet. Samlet antal points, for denne dag, endte på 22 – hvilket er noget højere end de 15 points gruppens velocity lå på.

ONSDAG DEN 11. DECEMBER

Efter dagens stand-up møde, forsatte arbejdet på story 2.5 samt 2.2 og 2.3 blev påbegyndt efterfølgende.

De resterende 4 points i spike 2.5, blev hurtigt afsluttet og testet igennem – og da succeskriteriet, at kunne sende data til serveren og modtage en bekræftelse for denne, matchede den i story 2.1, kunne denne implementeres direkte senere hen i sprintet.

Da story 2.2 blev påbegyndt, gik det hurtigt op for gruppen – at bekymringerne omkring endianness, var ubegrundet. Da begge arkitektur benytte sig af little endian, eller det vil sige at ARMv6k benytter denne som standard og kan benytte big endian, var der umiddelbart ingen grund til at sende data via MessagePack. Refactoring af den oprindelige kode fra spike 2.5, så denne kunne håndtere deserialisering af JSON data sendt fra tyndklienten/terminal blev påbegyndt denne dag.

Story 2.3 blev påbegyndt af en enkelt mand, men der gik ikke længe før kompleksiteten af denne gjorde, at der blev afsat to af gruppens medlemmer til par-programmering på denne.

Dagen afsluttede med at spike 2.5 var færdiggjort samt refactoring af dennes kode med funktionaliteten fra story 2.2 og 2.3 var påbegyndt, med en afslutningsprocent på de omkring 30%, 2.5/8 points på story 2.2 samt 6/20 på 2.3, på begge stories. Samlet antal points, for denne dag, endte på 13 – hvilket er lavere end den forgående dag, dog tættere på gruppens afsatte velocity.

TORSDAG DEN 12. DECEMBER

Der blev ved dagens stand-up møde, vedtaget at introducere en ekstra story – da de forrige stories var for pessimistisk estimeret og gruppen ville løbe tør for disse, inden sprintet var afsluttet. Story 4, som blev valgt var visning af rejseoversigt for brugere, grundet dens lave estimat og kompleksitet.

Den oprindelige kode fra spike 2.5 blev implementeret som story 2.1 i projektet og der blev forsat arbejdet videre med story 2.2.

Story 2.3, med resterende 14 points, blev afsluttet denne dag og testet igennem – så der var sikkerhed omkring, at graf-servicen kunne håndtere både enkelte og sammensatte rejser. Gruppen var dog overrasket over, på trods af dennes kompleksitet, at denne kunne færdiggøres så langt under estimatet på 20 points.

Story 4, som blev tilføjet dette sprint som ekstra story ved dagens stand-up møde, blev påbegyndt samme dag.

Dagen afsluttede med at story 2.1 samt 2.3 var færdiggjort, samt story 2.2 og 4 havde en afslutningsprocent på de omkring 50% og 67%, henholdsvis 6.5/8 points og 2/3. Samlet antal points, for denne dag, endte på 25 – hvilket er sprintets højeste og 40% over velocity.

FREDAG DEN. 13 DECEMBER

Var sidste arbejdsdag på det pågældende sprint og efter dagens stand-up møde, blev der arbejdet videre på story 2.2 samt 4.

Refactoring af koden i story 2.1 forsatte og gruppen kunne inden længe producere persistent data gennem den tyndklienten/terminal via serveren. Det samme gør sig gældende for story 4, hvor brugeren har mulighed for at se en rejseoversigt over ikke sammensatte rejser.

Dagen afsluttede med at story 2.2 samt 4 var færdiggjort og det samlet antal points for denne dag, endte på 2 – hvilket er sprintets laveste.

MANDAG DEN. 16 DECEMBER

Dagen gik med et tilbageblik på sprint 2 samt en mindre præsentation af denne foruden en offentlig fremvisning af produktets daværende tilstand.

Desværre var der sneget sig en fejl med, som vidste sig under fremvisningen, hvilket resulterede i at det ikke var muligt for en bruger at få adgang til systemet. Dette er naturligvis yderst uheldigt og situationen kunne være havde været undgået, med test og en gennemgang af koden før fremvisningen.

For yderligere information vedrørende det retrospektiv segment af præsentationen, henvises der til afsnittet 2. Sprint/Retrospektiv på side 33 og 34.

STORIES

Følgende er beskrivelsen af de stories, som indgik i dette sprint.

Story Name:			ID:
Kommunikation fra tyndklient/terminal til server			2.1
As a	Arduino Uno.		
I want to	Kunne sende data til systemets API.		
So that	Jeg kan sende data omkring tjek ind/ud.		
Estimate:	Actual:	Priority:	
5	- ikke angivet -	11	

Denne story gik i sin enkelthed ud på at kunne oprette en forbindelse mellem eksterne enheder og serveren. Da der var færdiggjort et spike (ID 2.5), hvori der blev eksperimenteret med sokkel kommunikation – hvor succeskriteriet var at kunne sende data til serveren og modtage en bekræftelse for denne – var det blot at implementere disse i projektet.

Story Name:			ID:
API: Kommunikation			2.2
As a	API.		
I want to	Modtage data fra Arduino Uno via strømsokkel og MessagePack.		
So that	Tjeck ind/ud kan registreres korrekt.		
Estimate:	Actual:	Priority:	
8	- ikke angivet -	10	

For at kunne håndtere indkommende transaktion forespørgsler, blev der kigget på et binær serialisering format (MessagePack) grundet bekymring med endianness – da den tyndklienten/terminalen der tilgår systemets API og server er af to forskellige arkitektur; x86_64 og ARM. Da begge af disse arkitekture benytte sig af little endian, viste det sig, at den tidligere bekymring var begrundelsesløs og at JSON ville, uden yderlige problemer, kunne benyttes til den ønskede funktionalitet. Der kan argumenteres for, om denne story skulle have været et spike, grundet dens eksperimental lignende formål eller en under task til story 2.1 – dog var det gruppens ønske, at beholde kommunikations formatet som en selvstændig story.

Story Name:			ID:
API: Graf-service			2.3
As a	API.		
I want to	Bruge graf-servicen til at beregne den korrekte billetpris for en rejse.		
So that	Ejerne kan trække det korrekte beløb fra kunden.		
Estimate:	Actual:	Priority:	
20	- ikke angivet -	12	

Denne story omhandlede behandling af rejsetransaktioner – til det formål at udregne den korrekte billetpris. Dette gøres ved at beregne hvilke zoner der rejses i og igennem, ved at benytte følgende parameter: start samt slut id (holdeplads nr.). Hvilke zoner der er rejst fra, igennem og til, er ikke det interessante – men derimod antallet af zoner. Som udgangspunkt, kun gældende i Region Nordjylland, kan billetprisen udregnes som følgende: 1 zone = 20DKK samt >= 2 zoner = 10DKK per zone.
Der blev benyttet par-programmering under denne story, hovedsagelig grundet kompleksiteten. Selve estimeringen af den enkelte story var pessimistisk, da denne uden yderlige problemer, kunne have været på det halve.

Story Name:			ID:
SPIKE: Oprettelse af netværksforbindelse på Arduino Uno			2.4
As a	Arduino Uno.		
I want to	Undersøge hvordan man for net forbindelse.		
So that	Denne kan kommunikere med backend.		
Estimate:	Actual:	Priority:	
13	- ikke angivet -	1	

Under undersøgelsen af første spike, blev det valgt at benytte en Raspberry Pi. Dette blev begrundet med, at det var muligt at tilslutte denne via traditionel ethernet forbindelse, frem for en mere ustabil USB netværks tunnel. Erfaringen med denne type af "rapid prototyping board" samt Linux i gruppen, gjorde at den oprindelige estimat på 13 kan opfattes som særdeles pessimistisk.

Story Name:			ID:
SPIKE: Opretholde klient/server forbindelse mellem enheder			2.5
As a	Klient/server.		
I want to	Undersøge modtagelsen af data og status.		
So that	Der er en kommunikations linje mellem disse.		
Estimate:	Actual:	Priority:	
13	- ikke angivet -	1	

I det andet spike, blev der tilføjet 3 under tasks; synkron sokkel, asynkron sokkel samt kommunikation på kryds af arkitektur. De 2 første under tasks, synkron og asynkron sokkel kommunikation, blev udviklet og testet uafhængigt af ekstern enhed – da disse var til, for at finde den mest optimale måde at modtage og behandle data på. På den 3. under task, blev der udviklet en klient til afvikling på den eksterne enhed. Der blev benyttet par-programmering, til et hvis punkt, i disse 3 under tasks – hovedsagelig grundet introduktionen af et nyt sprog (C) samt brugen af strømsokkel var ukendt for den største del af gruppen

Story Name:			ID:
Visning af rejseoversigt for brugere			4
As a	Bruger/kunde.		
I want to	Se en liste af mine tidligere, ikke sammensatte, rejser.		
So that	Jeg kan afstemme disse, med mine afsatte udgifter.		
Estimate:	Actual:	Priority:	
3	- ikke angivet -	13	

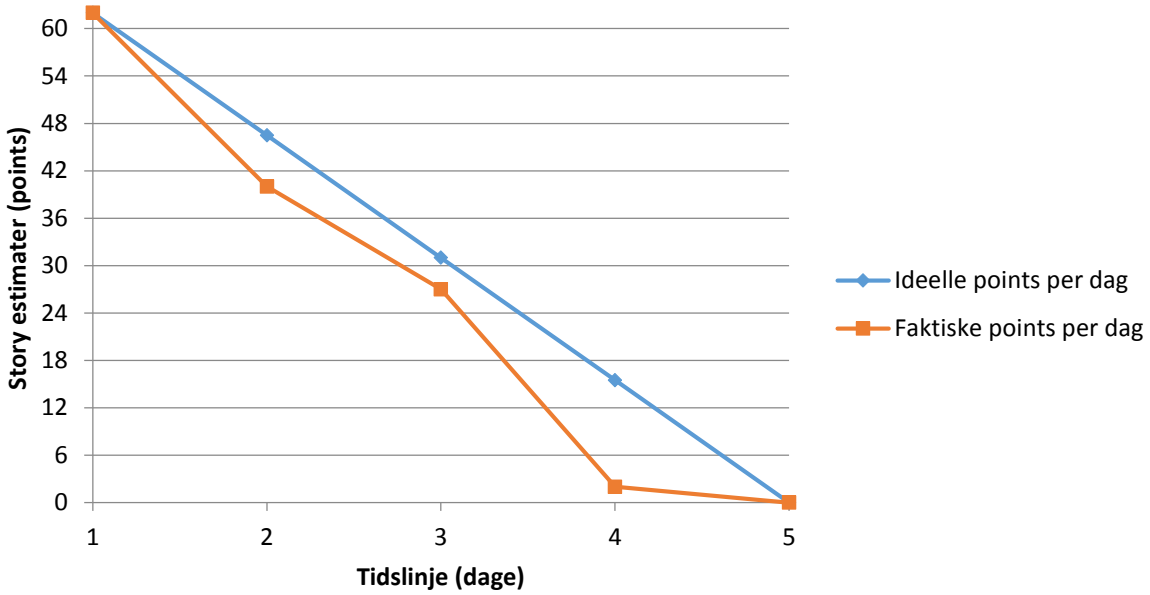
Denne ekstra story der blev introduceret torsdag den. 12/12/13, var af en ukompliceret størrelse og hvis eneste formål, var at give brugeren adgang til at se forrige, ikke sammensatte, rejser. Grunden til at storien kun indeholde visning af ikke sammensatte rejser, var at denne feature ikke var implementeret på det givne tidspunkt – ellers ville der være argument for at dele storien op i 2 under tasks som indeholder begge.

ESTIMERING

Som i det forgående sprint, blev planning poker benyttet til estimeringen af stories. Gruppen var ofte enig i estimaterne og hvis nogen skilte sig ud – blev der lyttet til deres argument og efterfølgende re-estimeret.

Der blev i alt til 62 points, fordelt over 6 stories og sprintet forløb, på trods af story 2.3 samt spike 2.4 var noget pessimistiske og introduktionen af en ekstra story, forholdsvis acceptable med en variation på den estimerede og faktiske velocity på +0.5 points.

BURNDOWN DIAGRAM



STAND-UP MØDER

Som i sprint 1, afholdte gruppen stand-up møde hver formiddag klokken 09:00. Disse var fordelt således:

- Forgående/afsluttede stories
  Den enkelte stories beskrivelse og formål, status, eventuelle problemer der skulle have været.
- Nye stories
  Stories som skal påbegyndes, deres beskrivelse og formål samt kompleksitet.
- Vidensdeling
  Q&A hvor alle gruppens medlemmer, kan stille spørgsmål til andre stories og dens ejer.
- Sprintets forløb
  Kort gennemgang af sprintets forløb, antal rest stories, nedskrivning af burndown diagram.

På mødet, blev der ligeledes diskuteret behovet for par-programmering ved enkelte stories og dennes nødvendighed.

Der kan dog argumenteres for disse møders nødvendighed, grundet gruppens størrelse samt indgående kendskab til hinandens tilgang til stories og udviklingsmetoder.

VIDENSDELING

Dagens stand-up møde, satte gruppen i stand til at kunne forstå problemstillingerne som det enkelte story havde, samt en åbenhed gennem dagen som gjorde at alle i gruppen kunne stille eventuelle spørgsmål omkring et kode segment eller story.

Såfremt der skulle opstå problemer, en ny teknologi skulle tages i brug eller en story havde en kompleksitet af en hvis størrelse – var par-programmering en naturlig løsning for gruppen, hvilket også fremmede den interne vidensdeling.

RETROSPEKTIV

Som i forrige sprint, blev retrospektiv for dette forløb udarbejdet og præsenteret. Under dette sprint, blev der forbedret end del fra det forrige, dog var der enkelte områder som stadig faldt uden for ønsker til forløbet.

GODE

- Par-programmering (selektivt)

  Gruppens ønske, om at par-programmering skal forgå selektivt, har fungeret godt. Hvis det ikke var fastlagt forud for påbegyndelsen af en story – og der var ønske om dette fra dennes ejer, blev der hurtigt fundet en løsning på dette. Førnævnte har også minimeret eventuel spildtid, ved at flere arbejder på en storie, hvis kompleksitet var lav.
- Planlægning af spikes

  Introduktionen af de 2 spikes, benyttet i dette sprint, har uden tvivl været en stor hjælp under forløbet. Udarbejdelsen af disse spikes, har dog ikke været problemfrit – da en enkelt spike havde den samme succeskriteriet som en storie, hvilket gjorde at disse 2. estimater faldt uden for det forventede.
- Stand-up møder

  Der blev fastlagt en agenda, som kan ses i afsnittet "Stand-up møder", hvilket resulterede i at gruppen fik en del mere ud af de daglige møder og også fremmede den interne vidensdeling.
- Afslutning af påbegyndte stories

  Introduktionen af spikes i dette sprint, har været en betydelig forbedring fra forrige – når det gælder afslutningen af komplekse stories. Dog hang enkelte stories som "in-progress" i længere tid end ønsket, grundet at funktionaliteten ikke var mulig at implementere på det givende tidspunkt.

KUNNE HAVE VÆRET BEDRE

- Estimater

  På trods af at alle stories blev færdiggjort, var disse stories generelt pessimistisk estimeret, hvilket resulterede i at gruppens velocity, til et punkt, kan betragtes som værdiløs. Dog skal det bemærkes, at arbejdsmængden passede, forholdsvis, til sprintet.
- Proces dokumentation

  Var desværre ikke tilstede under forløbet, på trods af den gavn det kunne have givet gruppen. Den manglende dokumentation, kan efterfølgende betragtes som et irritationsmoment for gruppen og der skulle have været mere opmærksomhed på dette.
- Kvalitetssikring

  En enkelt fejl, havde sneget sig med til sprintets bedømmelse om mandagen den. 16, som gjorde at brugergodkendelse processen ikke var funktionsdygtig under præsentationen for kunden. Fejlens marginale størrelse, kodemæssigt, ændrer dog ikke at dette er uacceptabelt – og at denne var til stor ærgrelse for gruppens medlemmer.

FORBEDRINGER

- Proces dokumentation, logbog

  En fælles logbog for gruppen, kunne være acceptabel, hvis den ikke blot var skrevet af en enkelt medlem. At dette er sket, har resulteret i en mangelfuld fortælling om sprintets gang, da alle stories ikke er nævnt eller beskrevet i denne.

- Proces dokumentation, scrum-practises

  Der er blevet diskuteret i gruppen, om indførelse af en praksis i gruppen agile metodik, som omhandler indførelsen af en kort dokumentation af den afsluttede storie. Dette ville have været til gavn, for alle gruppens medlemmer, i udarbejdelsen af iterationens proces dokumentation.

3. SPRINT (UGE 51)

Det tredje og sidste sprint i forløbet foregik i perioden 16. december 2013 til 20. december 2013, hvor den 20. var afsat til reviews og retrospektiv. I dette sprint skulle meget af funktionaliteten omkring rejser udbygges til blandt andet forsat rejse, havde brugeren nok penge på kortet og brugeren skulle kunne følge med i sine rejser via hjemmesiden.

STORIES

Herunder er der en beskrivelse af hver story, der indgik i dette sprint.

Story Name:			ID:
Forsat rejse			11
As a	Bruger.		
I want to	Betale det rigtige beløb ved en sammensat rejse.		
So that	Så jeg ikke betaler for meget eller for lidt.		
Estimate:	Actual:	Priority:	
40	- ikke angivet -	7	

Denne story var den mest komplekse i dette sprint og der var i starten, én hvis usikkerhed omkring hvordan den skulle løses.

Vi ønskede i vores system at implementere princippet med forsat rejse fra hvor på måden det fungerer på med kontant billet. Derfor skulle alle rejser der foregik inden for én time efter den første billet var købt, være gratis, angivet at det var i samme zone man rejste.

Som prioriteten viser var dette den første story vi startede med i sprintet. Vi var 2 gruppemedlemmer der gik sammen om hele denne story og lavede par-programmering på den. Dette valgte vi fordi opgaven var så kompleks som den var og det var meget vigtigt det blev korrekt regnet ud og vi ingen fejl havde omkring dette.

Story Name:			ID:
For lidt penge på kort			5
As a	Bruger.		
I want to	Vide hvornår jeg skal indbetale penge på kortet.		
So that	Jeg ikke kommer til at løbe tør		
Estimate:	Actual:	Priority:	
13	- ikke angivet -	9	

I denne story, ønskede vi at når brugeren tjekkede ind ved hjælp af rejsekortet, skulle standeren (i vores tilfælde Raspberry Pi'en) giver en besked om det var godkendt tjek-ind eller ej. I første omgang havde vi 2 scenarier, godkendt tjek-ind og for lidt penge på kort. Dette skulle foregå ved at vores klient sendte en besked til serveren om denne bruger ønskede at tjekke ind. Serveren skulle herefter sende besked tilbage til klienten om hvor vidt brugerens tjek-ind blev godkendt eller der var for lidt penge på kortet. Der skulle så afspilles en passende lyd herefter.

Vi brugte par-programmering i forbindelse med denne story, da ét gruppemedlem stod for at udvide socket serveren til at sende besked tilbage og ét andet gruppemedlem stod for klienten.

Her var der behov for tæt kommunikation mellem de 2 gruppemedlemmer for at få dette til at snakke sammen hurtigst muligt og være sikker på vi arbejdede i samme retning.

I og med dette var vores første projekt, hvor vi har arbejdet med socket kommunikation, så var der usikkerhed omkring, hvordan denne opgave kunne løses. Derfor ramte vi også en del ved siden af i forhold til estimatet på de 13 points/timer – hvilket i aktuelle timer blev til 4 points fordelt på 2 mand.

MORGENMØDER

Efter de 2 første sprint, var møderne i dette sprint blevet meget bedre. Vi fulgte vores plan for morgenmødet og dermed var vi sikre på vi fik gennemgået de vigtige ting hver dag. Vi fik hørt hvert gruppemedlem om hvad personen lavede i går, om der var nogle problemer og om hvad de forventede at lave i dag. Også fik vi drøftede eventuelle problemer der havde været i forbindelse med en story samt hvis der var usikkerhed omkring story som skulle igangsættes.

Det virkede dog stadig "kunstigt" ved hvert møde, som højst sandsynligt har noget at gøre med vi kun var 3 personer i gruppen. Ved hvert møde vidste vi stort set alt der blev nævnt på mødet forinden, i og med det havde været en del af dagen før. Derfor kan det vurderes om det var optimalt at holde de møder, når vi ikke var flere gruppe medlemmer.

VIDENSDELING

Vi har i gruppen gjort brug af vidensdeling på den måde, at inden vi gik i gang med en story, blev de drøftet med alle gruppemedlemmer. Der blev gået igennem hvordan den skulle løses, hvilke problem stillinger der var osv. Såfremt der opstod problemer undervejs, var det hermed nemmere for de andre gruppemedlemmer at sætte sig ind i hvorfor det var sådan og på den måde havde vi også lettere ved at komme frem til en løsning. Til de komplekse opgaver, hvor vi valgte at bruge par-programmering, havde vi selvfølgelig også undervejs i udviklingsprocessen af den opgave, en hvis sparring mellem de 2 parter der par-programmeret.

ESTIMERING

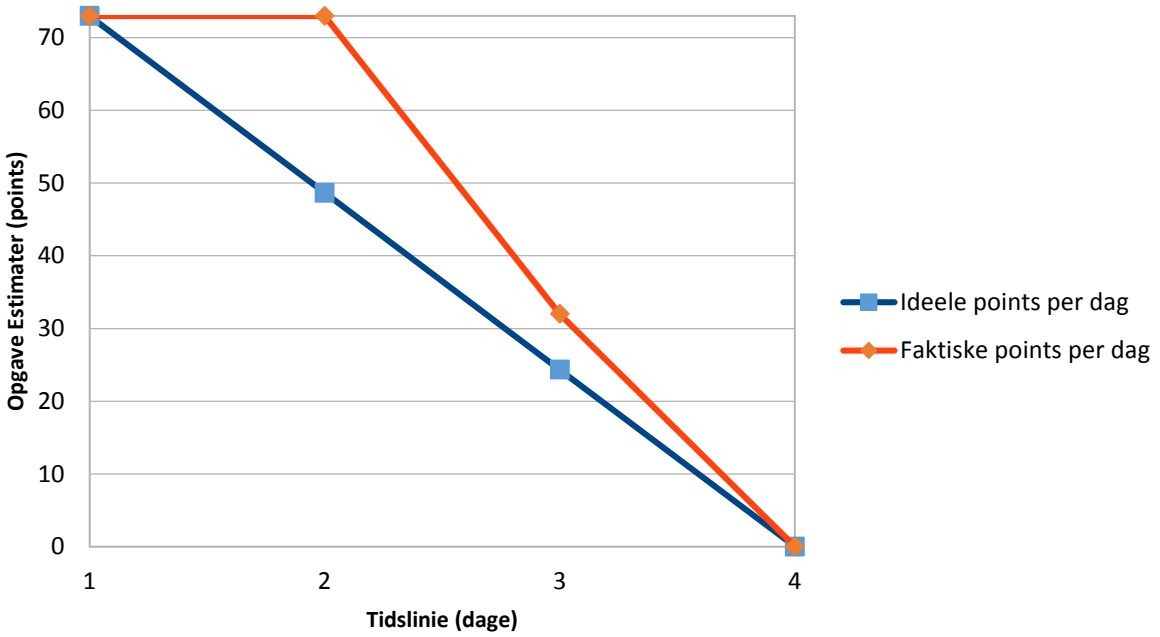
De 3 stories var ved sprintets start allerede estimeret fra 1. sprint og vi snakkede om i gruppen om der var behov for en re-estimering af de stories. Vores vurdering var at hver estimat passede og der var derfor ingen grund til at re-estimere. Når man ser tilbage på dette, var det ikke det bedste valg vi traf der og vi burde have re-estimeret vores stories inden sprintets start.

Vores estimater på stories for dette sprint, viste sig at være meget pessimistiske og skudt noget over mål. Det vi mener har haft stor indflydelse på vores estimater har ramt ved siden af, er den usikkerhed der har været omkring nogle af opgaverne og deres teknologi.

Som nævnt før, var socket kommunikation ikke noget vi havde arbejdet med før. Dermed viste vi ikke hvordan opgaven hvor dette indgik i helt skulle løses og der skulle noget research til inden.

Vi havde som man kan se på vores burndown nedenfor estimeret de 3 stories til 73 storypoints i alt. Med vores velocity på 15 points per dag ville vi, hvis vores estimater holdte stik, ikke kunne nå alle vores stories. Det gjorde det "heldigvis" ikke og vi blev færdig med alle stories i slutningen af sprintet.

BURNDOWN DIAGRAM



RETROSPEKTIV

Gode

- Par-programmering (selektiv)
  Vi valgte i nogle af opgaverne (specielt forsat rejse) at arbejde i par. Det har fungeret rigtig godt den måde vi har udvalgt hvilke opgaver der har været behov for vi par programmerede omkring. Så det ikke var ved simple ting, vi sad to og løste noget én lige så godt kunne.
- God til at afslutte opgaver i "in-progress"
  Vi har i de forgangne uger ofte haft nogle opgaver til at hænge i in-progess i langtid, hvor der kun manglede at blive lavet lidt på, men hvor det ikke var muligt pt. Dette blev meget bedre i sprint 3 og vi oplevede slet ikke dette med vi pludselig havde gang i alle stories på én gang.

Kunne have været bedre

- Estimater (points), men arbejdsmængde passer!
  Som nævnt nogle gange ovenfor, har vores estimater på de forskellige stories ikke være helt optimale. Det passede dog arbejdes mæssigt, med antal stories vi havde regnet med vi kunne nå i dette sprint og vi havde hverken for mange eller for få opgaver.

- Proces dokumentation

  Dette var desværre noget der stort set var ikke eksisterende under sprintet og derfor burde der have været lagt meget mere vægt på dette.

- Kvalitetssikring

  Vi havde desværre endnu engang fejl i vores kode, da vi skulle lave review med kunden/product owner. Dette er selvfølgelig ikke acceptabelt og derfor burde alt materiale være gået igennem inden det skulle fremvises.

- Kommunikation i tilfælde af fravær

  I starten af dette sprint, havde vi et gruppe medlem som var nødsaget til at arbejde hjemme fra på grund af sygdom. I forbindelse med dette, blev opgaverne uddelegeret af gruppemedlemmerne der var mødt op og derefter kommunikeret videre til det fraværende gruppemedlem via Skype. Der skete et brist i kommunikationen her og da dagen var omme. Viste det sig at vi havde arbejdet på samme story og dermed var 5-6 timers arbejde spildt. Dette skal selvfølgelig være bedre fremover, så alle er 100% med på hvad der skal foregå og alle ved hvem der laver hvad.

Forbedringer

- Bedre proces dokumentation ved at alle laver "logbog"

  Det stort set kun blevet ført fælles logbog, skrevet af et gruppemedlem. Dette ville have gjort proces dokumentationen noget nemmere, hvis der var blevet gjort mere ud af dette, så alle medlemmer af gruppen skrev en logbog for dagen.

- Der skal måske laves stories omkring dokumentation over Scrum- practises, så folk tvinges til at lave det!

  I forbindelse med overstående punkt, kunne man indføre en "practice" som gjorde at efter hver opgave eller story er afsluttet skulle der skrives en hvis mængde proces dokumentation, simpelthen for at komme det her problem til livs med det ikke bliver gjort.