Forside

Flixnet

3. semester projekt

Gruppe 1 (DMAB0917)
Andreas Larsen, Katrine Vindbæk og Mathias Lindberg

Dato: 17/12 2018
Antal anslag: 41649
Git navn + link: https://github.com/linaqx/Projekt-3-Gruppe-6.git
Git revisions nummer: 233
database navn: dmab0917_1026423

Indholdsfortegnelse

Indledning

3. Semester er semesteret hvori vi, de studerende lærer at kode i C# i den nye platform 'visual studio' for at lave den ukendte arkitektur, som ligger i det distribueret system samt arbejde i en agil udviklingsproces. Kulminationen af alle disse nye spændende elementer er blevet et slags film og serie wikipedia. Systemet skal udfylde et hul for film og serie entusiaster, hvor nogle af de tiltænkte funktioner blandt andet inkluderer fælles favoritlister, datoer for hvornår kommende afsnit eller film udkommer, generel information om filmene og anmeldelser/kommentarer på underholdningselementerne. For selvom der findes sider, som 'imdb' så føler gruppen, at der i det system er nogle mangler. Følgende projekt præsenterer blandt andet emner fra pensum og hvordan dette udføres i praksis.

Problemformulering

Problemformulering er udformet velvidende, at projektet først og fremmest er et læringsprojekt, hvorfor der ikke kræves en økonomisk strategi. Havde det dog været tilfældet, kunne inddragelse af reklamer og Premium brugere være relevant. Med det i mente lyder følgende problemformulering således:

Hvordan kan man lave en distribueret platform som samler information om film og serier?

- *Hvordan kan man identificerer problemer, specielt samtidigheds problemer og udfordringer i udviklingen af en sådan platform?*

- *Hvilke løsninger, herunder særligt hvilken arbejdsmetode kunne man anvende for at imødekomme disse problemer og udfordringer?*

- *Hvordan kan man lave et distribueret system, som både repræsenteres gennem en app og en web applikation med samme funktioner og som er koblet op til samme database?*

Da problemformuleringen er fælles for både systemudvikling rapporten og teknologi/programmerings rapporten besvares første underspørgsmål ikke i denne rapport.

Idegenerering

Gruppens idegenerering starter allerede før, der eksisterer en problemformulering. Gruppen starter med at være seks studerende, som fælles genererer ideer ved hjælp af en brainstorm. Da der er genereret 7-8 ideer, skal gruppen opdeles i to for at leve op til studieordningen, som dikterer at en gruppe maksimalt må bestå af fem personer. Dette gøres ved, at alle seks medlemmer stemmer på de idéer, som de finder bedst. Deri opstår en naturlig opdeling på to tre-mands grupper.

For at validere den idé som denne gruppe stemte på, bliver der lavet en mundtlig business model canvas, hvori gruppen gennemgår de tre af de ni byggeblokke. De ni byggeblokke hedder; Kundesegment, Værdi, Kanaler, Kunderelation, Indtjening, Nøgleressourcer, Nøgleaktiviteter, Nøgle samarbejdspartnere og Udgifter.

De tre som blev diskuteret mundtligt var Nøgleressourcer, Nøgleaktiviteter og Nøgle samarbejdspartnere. I den forbindelse diskuterede gruppen, hvilke ressourcer gruppen havde til rådighed. Ydermere hvilken aktivitet der skulle udføres for at realisere projektet og hvilke samarbejdspartnere gruppen havde adgang til.

Nøgleressourcer
Nøgleressourcer er udgjort af de tre gruppemedlemmer og deres kompetencer, som blev vurderet tilstrækkelig til et projekt af det omfang, som gruppens projekt har og de mere lavpraktiske elementer, såsom tre funktionelle computere med de nødvendige værktøjer installeret.

Nøgleaktiviteten
Aktiviteten er handlingen bag ressourcerne. Dette dækker over de specifikke kompetencer til at programmere, kommunikere internt i gruppen og til vejledere, samt udarbejdelsen af rapporten. Disse afdækker de planlagte elementer fra undervisningsgangen og projektet.

Nøgle samarbejdspartnere
Samarbejdspartnere består i dette tilfælde af vejledere fra de respektive fag: systemudvikling, teknologi og programmering.

Eftersom der findes ressourcer, aktivitet og samarbejdspartnere til ideen, er en realisering af denne derfor mulig. Idéen specificeres og uddybes i næste trin, hvori gruppen laver en 'mock up' og generer user stories, som specificerer idéen og gør den konkret, samt klar til at blive arbejdet med i sprinterne.

Delkonklusion

Ideen for programmet er et opslagsværk for film og serier med en forventet brugerbase af film entusiaster. Idéen overlever den indledende diskussion, hvori gruppen diskuterer om pensum kan repræsenteres gennem et program af denne type.

Plandreven versus agil udvikling ekspliceret gennem konkrete metoder

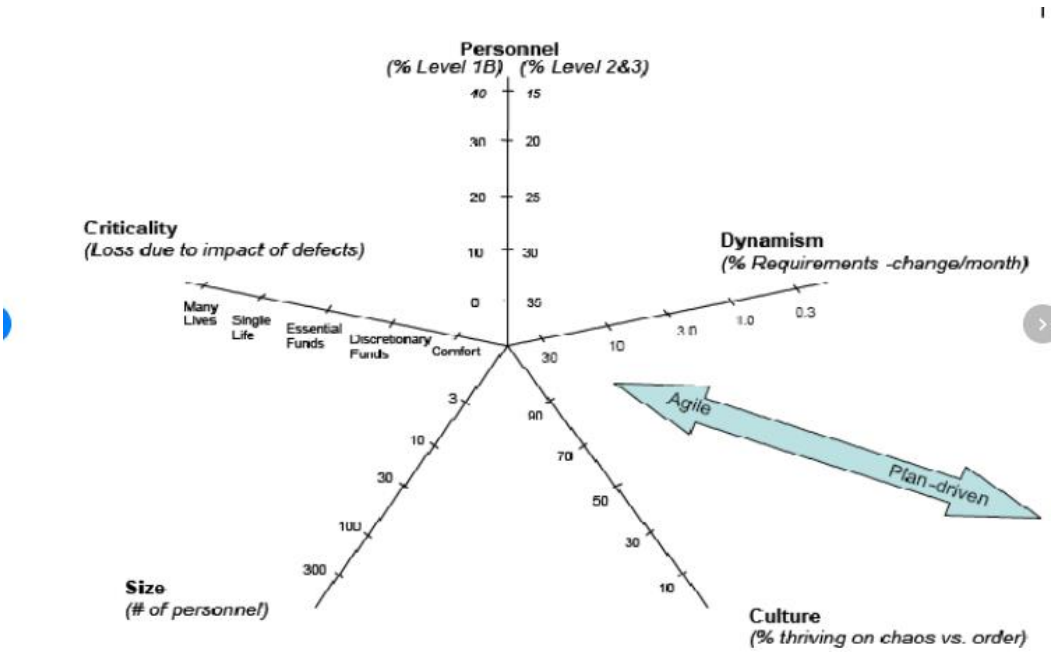
I et udviklerteam er metode en beskrivelse af den process, som teamet benytter. Der er to overordnede metoder, 'Agil programming' og 'plan driven programming'.

Agil programming er hurtigt og kun med et vag defineret mål. Der er plads til ændringer og kodning uden den store planlægning, hvilket også betyder, at produktet kan blive hurtigere færdigt, men at risikoen for fejl stiger. En metode i systemudvikling indikerer ikke et enten-eller valg. For det første er plan driven -> agile en skala, som går fra den mest plandrevne udviklingsmetode, som er vandfaldsmodellen til den mest agile metode som er Kanban. For det andet så kan metoder indenfor systemudvikling godt kombineres, et godt eksempel på dette er Scrum og XP. Den agile metode inkluderer eksempelvis Scrum, XP og Kanban. Udover en minimal planlægning har de også det tilfælles, at de alle udvikler uden et overblik over, hvornår projektet er færdigt. De kan godt have en deadline, men mængden af funktionalitet ligger ikke fast, i samme grad som et projekt udviklet gennem en plan dreven metode. Det skyldes, at dokumentationen bliver lavet inkrementalt, altså i takt med, at programmet udvikler sig. Heraf også den øgede chance for fejl, fordi planlægningen ikke er lige så grundig, som plan driven programming.

Plan driven programming er forskellige processer, som i langt højere grad planlægger før de koder. Da alt bliver planlagt før systemet bliver lavet, er det sub-optimalt, hvis man finder ud af, at en eller flere dele af systemet ikke virker. Hvis dette sker er der mange modeller og tabeller der skal ændres på og i nogle tilfælde skal man starte helt forfra. Her findes der blandt andet vandfaldsmodellen og UP (unified process), der kendetegnes ved, at alle modeller og tabeller og alt model dokumentation først skal udformes og efterfølgende opdateres, hvis der sker ændringer i systemet. Sådanne metoder bruges ofte, hvis sikkerheden i programmet skal være højt. Metoderne kan også benyttes, hvis kunden underlægger projektgruppen nogle constraints, eksempelvis en deadline eller et maksimum kapital, som projektgruppen har til rådighed. Det kan være fordelagtigt for kunden at kende prisen på systemet, men samtidig er muligheden for at tilføje mere funktionalitet, når først projektet er startet, meget lav.

De fem akser

For at beslutte om man som projektgruppe vil anvende agil- eller plan driven programmering kan de fem akser diagrammet benyttes se på figur 1



De fem akser definerer fem skalaer, som kan udfyldes for at skabe en bedre forståelse for, hvilket metode valg der skal træffes.

Personnel
Personnel beskriver den samlede viden i en udviklingsgruppe inddelt i niveauerne 1 og 2-3. Jo større en andel af uerfarne programmører eller specialister, desto større er sandsynligheden for, at en plandreven metode skal vælges og flere erfarne programmører muliggør en agil metode.

Criticality
Beskriver i hvilken grad systemet kritisk kan påvirke kunden økonomisk eller ved tabte menneskeliv. Et system der skal navigere militære missiler ville her have et højt "criticality" niveau, da forkerte programmeringsbeslutninger i værste tilfælde kan medføre omkomne mennesker.

Size
Størrelsen på udviklingsteamet er også vigtigt at tage i betragtning, hvis der eksempelvis er 100+ personer, som påvirker programmet, er den plandrevne metode fordelagtig. Begrundelsen er, at der i forbindelse med flere personer også kommer mere kommunikation og derigennem stiger sandsynligheden for fejl. Dermed skal der genereres solid dokumentation, som kan anvendes som reference.

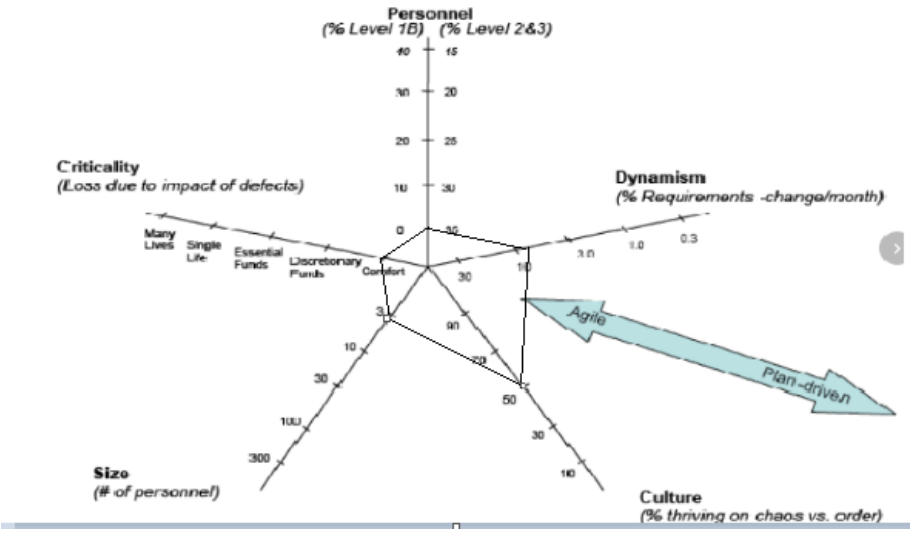
Culture
Ved kultur forstås teamets optimale arbejdsmiljø som orden mod kaos. Desto bedre teamet agerer i kaos desto bedre ville en agil metode være.

Dynamism
Det dynamiske aspekt omhandler, hvor mange ændringer der forventes at komme om måneden. Mange ændringer lægger op til anvendelse af en agil udviklingsmetode. Nogle plandrevne metoder tillader kun få eller slet ingen ændringer efter projektstart. For eksempel vandfaldsmodellen som ikke tillader, at man går tilbage til tidligere iterationerne. Ved iterationer forstås etaper i udviklingsfasen.

Diagrammet fungerer således, at der sættes et punkt på alle akserne og tegnes streger imellem disse så der bliver formet en femkant. Arealet af femkanten fortæller så om udviklingsteamet skal vælge en agil- eller plandreven programmerings metode. Et lille areal er lig med en agil udviklingsmetode.

De fem akser på gruppen

Denne gruppe har også efter deres idegenerering anvendt diagrammet med en femkant, som havde et lille areal som også er argumentet for, at XP og Scrum har været anvendt til dette projekt. Dette kan ses på figur 2



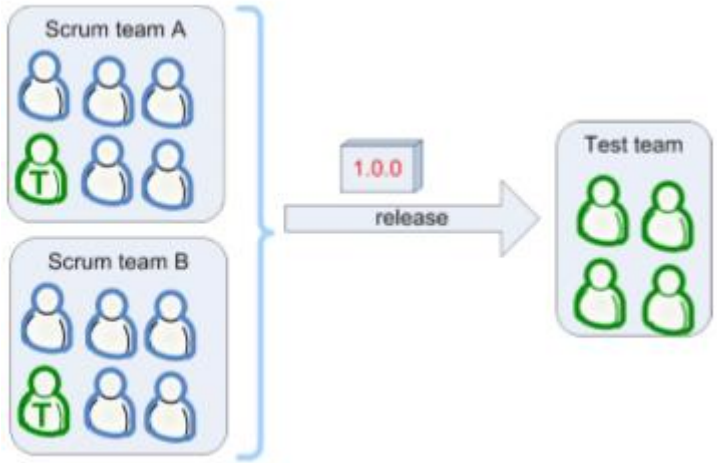
Gruppen består af tre personer og på grund af den størrelse har hver medlem også flere roller under projektforløbet. Den kulturelle akse er svær at specificerer fordi definitionerne af kaos og orden kan variere meget. Med den antagelse at en person som trives i ægte kaos aldrig ville lave dokumentation og planlægning og at en person, som trives i ægte orden ville rigidt genere dokumentation og planlægge til mindste detalje, så er gruppen placeret i midten. Der er enighed om, at der er et behov for en vis orden og struktur, men at projektet har plads til løbende ændringer. Det ses også i den dynamiske linje, hvori gruppen har estimeret, at der kommer cirka 10 større ændringer om måneder. Dette inkluderer ting som ekstra tabeller i databasen, refactoring af systemet, og ændringer i arbejdsfordeling. De tre medlemmer i gruppen er vurderet til at have kompetencerne til at kategorisere sig sig level 2 og 3. Det vil sige, at de er i stand til at udføre svære opgaver indenfor programmering. Dette er baseret på systemudviklings underviserens definition af level 1,2 og 3. Til sidst er det vurderet, hvis programmet fejler, så er tabet for brugere ikke mere end ubehag.

Som det kan ses på diagrammet er gruppens areal lille og det viser, at de agile arbejdsmetoder er oplagt for gruppen at anvende.

XP

I gruppens første sprint blev der arbejdet efter XP (extreme programming), som fokuserer på øget kode kvalitet og hurtige reaktion, hvis der sker ændringer i planlægningen. Et program lavet efter XP har typisk også kortere imellem releases/updates, som derfor også er mindre omfattende. Det sker igen for at maksimere kode kvaliteten og det gør programmet fleksibelt i tilfælde af fejl, da en release nemmere kan trækkes tilbage eller problemer kan løses. I XP arbejder en gruppe efter begrebet test first og koder derfor testen først og i takt med, at der bliver brug for det de nødvendige elementer til at få koden til at virke også kaldet 'test driven development'(TDD).

TDD er sammen med pair programming og incremental design, grundprincipperne i XP. Da XP ikke er et management værktøj på samme måde, som Scrum, indgår der ikke nogle specifikationer for størrelse af teams eller rollefordeling, men i Henrik Kniberg's: Scrum and XP from the Trenches, opstiller han et par krav til TDD, som styrker XP's kerneprincipper om øget kode kvalitet. Kniberg nævner for eksempel værdien af at have en kunde on-site for at komme med hurtigt feedback til udviklings-teamet. Den person skal sammen med et dedikeret test team, teste releasen fra udviklings teamet, før den bliver releaset til slutbrugeren. Om der bliver valgt et par udviklere, som er dedikeret til test eller om der er flydende rollefordeling i teamet, er op til det individuelle team og teamlederen. Fordelen ved et par dedikerede testere er tempo og at der undgås, at releasen bliver forsinket fordi test af systemet er blevet udskudt til sidst i et sprint. Det er i testfasen fejl bliver fundet, som skal rettes og det gør den fase svær at estimere i tid. Som Kniberg siger "Don't outrun your weakest link". Fordelen ved flydende roller i teamet er en lavere Truck factor (konsekvenserne for projektet hvis en udvikler bliver ramt af en lastbil) som ikke skal undervurderes i XP da der arbejdes i korte sprint. Kniberg foreslår også en anden struktur, som kan ses på figur 3.



Denne teamstruktur hjælper teamleder og product owner til at estimere tiden, som test teamet skal bruge fordi de kan få feedback fra testerne, som med denne struktur bliver en del af udviklingsteamet.

Pair programming(PP) er den udviklingsmetode, som bruges i XP. Den betyder i alt sin enkelthed, at der sidder to udviklere ved samme computer. Udvikleren som har tastaturet har til ansvar at programmere den nuværende opgave. Den anden udvikler skal observerer koden og stille spørgsmål ved de beslutninger, som den primære udvikler træffer, samt tænke koden ind i den større helhed. Rollen som primær udvikler og observatør skiftes mellem de to udviklere. PP er et værktøj, som øger kode kvaliteten markant, men koden bliver også produceret langsommere.

Incremental design(ID) er primært en automatisk konsekvens af TDD fordi at designet bliver refactoreret i forbindelse med ændringer i systemet. Det betyder, at dokumentationen holdes simpel og udvides i takt med at systemet gør det. Den behøver altså ikke at være fastlagt før udviklingen går i gang ligesom i plan driven development.

I dette projekt arbejdede gruppen i sprint 1 med XP. Gruppen opbyggede først sin test primært med pseudo kode altså kommentarer i koden om, hvad gruppen gerne ville have der skulle ske. Eksempel på pseudo kode kan ses på figur 4.

```
//Arrange

//CreatePerson
//Instance of DB
//create bool

//Act
//InsertPersonToDB
//set boolean true if personByID fundet

//Assert
//AssertTrue(boolean)
```

Testene følger også programmerings mønsteret AAA arrange, act, assert.

Gruppen pair programmede, men på grund af gruppens størrelse (3) var der en primær udvikler og to observatører. Gruppen anvendte ikke princippet incremental design fordi at dokumentation af henholdsvis domænemodel, relationelle model og mock ups allerede var lavet forudgående for sprint 1. XP styrkede kvaliteten af de første par user stories, men pair programming med 3 personer betød også, at gruppen ikke nåede i mål med de planlagte opgaver i sprintet.

Scrum

Modsat XP fokuserer Scrum mindre på den konkrete udvikling og mere på den overordnede planlægning og struktur af et projekt. Scrum kan kombineres med andre udviklings metoder for eksempel XP eller kanban. Scrum bliver også brugt fleksibelt i forskellige virksomheder. På 3 semester har gruppen for eksempel besøgt IT-virksomheden Logimatic, som arbejder i et stort Scrum team af 15 udviklere, som ville være et for stort team jævnfør eksperter på området. Logimatic tillod også, at der kom user stories ind midt i et sprint selvom det kunne betyde, at sprint loggen blev rykket og ikke kan gennemføres til den planlagte deadline. Dette er også frarådet af eksperter på området.

Disse tilpasninger kan i nogle tilfælde være nødvendige, det er derfor Scrum masterens ansvar at beslutte, om Scrum metoden skal tilpasses til den individuelle virksomhed og hvilke konsekvenser dette kan give.

I scrum metoden er der tre roller; Scrum master, product owner, og et development team.

Scrum masteren har ansvaret for, at development teamet overholder scrums regler, værdier og teorier. For eksempel skal en Scrum master sørge for, at user stories i backloggen lever op til de specifikation, som er aftalt og at de er detaljeret nok. Scrum masteren coacher også development teamet undervejs, hvis user stories skal uddybes. Scrum masteren har kort sagt det store overblik over udviklingsprocessen.

Product owners primære ansvar er prioriteringen af product backloggen, som product owner bestemmer efter et eller flere værdiskabende kriterier. Product owner skal derudover sørge for, at product backloggen er transparent og forståeligt for alle på teamet. Product owner kan uddelegere opgaver til for eksempel Scrum master eller development teamet, men ansvaret for at det er i orden er i sidste ende product owners.

Development teamet er ansvarlige for sprint backloggen og er selvstyrende, det vil sige, at ingen, ikke engang product owner eller Scrum master kan bestemme, hvordan development temaet skal opnå deres done kriterier. Et “Done” kriterier bliver defineret af product owner i product backloggen, men metoden og tilgangen for at opnå det kriterie er op til teamet selv. I et development team er der ingen titler. Det skal forstås sådan, at der ingen teamleder eller specifik tester findes i et development team. Der kan derimod godt opstå uofficielle roller i teamet såsom en stedfortræder, hvis Scrum masteren ikke er til stede til at dirigere det daglige stand-up møde eller én eller flere udviklere som får rollen som tester, men ikke titlen som tester - dette gøres for at teamet får et fælles ansvar, hvis et test done kriterie ikke er nået, så er det hele teamets ansvar og ikke kun vedkommende som havde rollen som tester.

I Scrum er teamet multifunktionelt, det vil sige, at i et optimalt team kan alle udviklere bruges til at udvikle alle de forskellige dele af projektet. Dette betyder også, at et optimalt team har en ikkeeksisterende truck factor.

Scrum er opdelt i fire formelle begivenheder og selve sprinten

1. Sprint Planning
2. Sprint
3. Daily Scrum
4. Sprint Review
5. Sprint Retrospective

Sprint Planning er planlægningen af sprinten og bliver udført af både teamet, Scrum master og product owner. Forudgående for planlægningen er product backloggen blevet lavet, så til sprint planning skal hele teamet kun fokuserer på at opbygge sprint backloggen ved hjælp af story points og tidsestimater. Scrum master hjælper teamet med at overholde Scrums regler og værdier samt at sørge for, at ‘sprint planning’ ikke tager mere end 8 timer til en 4 ugers sprint. Product owner skal hjælpe teamet ved at forklare og uddybe user stories således de bliver nemmere at prioritere for teamet.

Sprint. Sprinten bliver udført af development teamet. Her opdateres et Scrum board(se Scrum board) hver dag når en udvikler tager opgaver eller opdatere tiden på en allerede igangværende opgave.

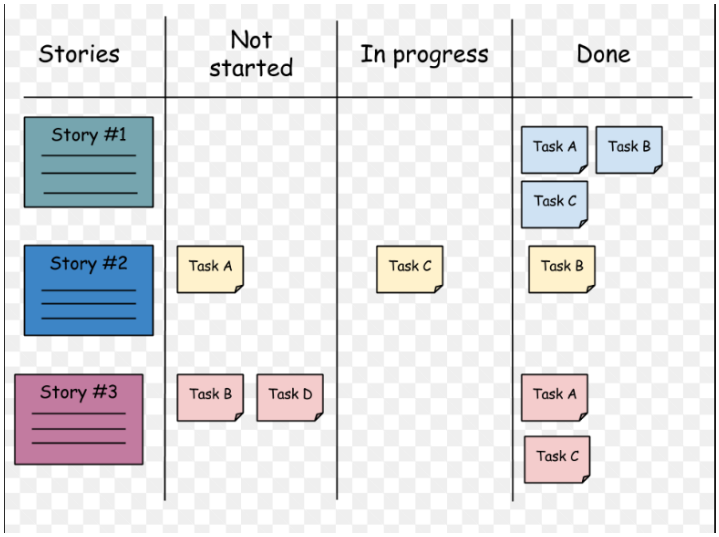
Daily Scrum. Daily Scrum eller stand-up meeting er 5-15 minutter i starten af hver arbejdsdag koordineret af Scrum master, hvor alle udviklere bliver spurgt om, hvad de lavede i går og hvad de har planer om at lave i dag. Det er her udviklere bliver sat på user stories og re-estimerer tiden, der er tilbage på en opgave. Spørgsmålene har ikke til formål at kontrollere, at udviklerne laver noget, men de har til formål at skabe en fælles refleksion og forståelse for, hvad der sker med projektet og hvis medlemmer af teamet har brug for hjælp, for eksempel pair programming kan det koordineres til dette møde.

Sprint review. Sprint review afholdes sidst i et sprint og har to primære formål, det at revidere sprinten som lige er afsluttet og som forudgående overvejelser til næste sprint. Sprint review dirigeres af product owner, som inviterer development teamet og stakeholders med til mødet. Scrum master kan også deltage med det formål, at mødet overholder den planlagte tid, samt at mødets mål bliver opnået. Til mødet ser deltagerne tilbage på, hvilke user stories der opnåede deres “done” kriterie og hvilke der ikke gjorde. Development teamet diskuterer, hvad der gik godt, hvad der gav udfordringer og hvordan de eventuelt blev løst i sprintet. Derefter bliver der i plenum givet feedback til, hvordan næste sprint kunne planlægges anderledes. Samt en redegørelse og diskussion indenfor deadline, budget og ressourcer.

Sprint retrospective er et møde som bliver afholdt umiddelbart imellem Sprint review og næste sprint. Mødet inkluderer development teamet og Scrum master, som har til formål at uddybe diskussionen fra sprint review om, hvad der gik godt særligt i forhold til process, værktøjer og relationer samt, hvordan der kan implementeres forbedringer i næste sprint.

Scrum board. Et Scrum board kan være mange forskellige ting for eksempel en tavle med post-it notes eller et digitalt Scrum board. Der findes flere forskellige eksempler på Scrum boards for eksempel trello, jira og icescrum. Formålet med et Scrum board er at repræsentere de user stories, som ligger i sprint backloggen. På boardet kan der være flere vertikale kolonner såsom “idle”, “in progress” “testing” “done”. I den optimale Scrum process er der ikke nogen “review” kolonne, men det kan variere for den individuelle virksomhed.

Et eksempel på et Scrum board kan ses på figur 5:



Delkonklusion

Valget mellem Plan driven development (PDD) og agile development kræver mange overvejelser for et udviklingsteam. For at besvare problemformuleringen og med den gruppesammensætning og værdier(fra de fem akser) så er 'agile development' blevet valgt. Den agile metode har været ny for gruppen, som tidligere kun har kendt til PDD metoder, men har efter en hvis læringskurve, vist sig at være fordelagtig til at udvikle et program med mange nye elementer som eksempel c#, visual studio og det distribuerede system.

Metodevalg

Hvordan har gruppe 6 brugt Scrum:
I projektet for 3. semester består gruppen af tre medlemmer. På grund af gruppens størrelse og det faktum, at projektets primære funktion er læring så har alle tre medlemmer af gruppen haft rollen som både product owner og Scrum master og det ansvar, som følger med disse roller. Det er blandt andet kommet til udtryk ved product backloggen, som er lavet fælles. Det samme er gældende for prioritering og story points.

Ansvaret for daily Scrum har også været fælles og er blevet afholdt hver dag. Daily Scrums varighed har i gennemsnit taget 5 minutter. Sprint review blev holdt i samarbejde med undervisere som aktivt agerer i rollen som Scrum master og med de andre studerende fra samme klasse som fungerede som en slags stakeholders. Der kunne komme med spørgsmål eller gode råd.

Gruppen har også brugt nogle elementer fra UP. Disse er mock ups, domænemodel og den relationelle model. Disse er blevet lavet for at gøre det nemmere for gruppen at designe GUI og nemmere for gruppen at se sammenhængen mellem de forskellige klasser og for at se, hvordan databasen skulle se ud.

Story points, planlægning og estimering
Da arbejdsmetoden Scrum var ukendt for gruppen før 3. semester er gruppen stødt på forskellige udfordringer, særligt inden for estimering og planlægning. Da et story point ikke repræsenterer en konkret værdi, er begrebet relativt abstrakt sammenlignet med use cases, som kun bliver prioriteret med muligheden for at estimere dem i tid. Gruppen besluttede under planlægningen og specificering af backloggen og de user stories, som skulle ligge deri, at et story point skulle repræsentere 1 dags arbejde. Gruppen vurderede user stories ved hjælp af Scrum poker, som betyder at alle medlemmer af gruppen har et antal kort, som repræsenterer forskellige værdier. I denne gruppes tilfælde 0.25, 0.5, 1, og 2 - som altså repræsenterede arbejdsdage. Den første udfordring gruppen stødte på var, at en arbejdsdag ikke var defineret. Derfor måtte gruppen re-estimere alle user stories. Der blev besluttet, at en arbejdsdag repræsenterer hele gruppens arbejdstimer. På en mødedag som, som et udgangspunkt skulle vare fra kl. 08.30 til kl. 15.00 altså 6.5 timer og med 3 gruppemedlemmer giver den totale tid 19.5 timer på en arbejdsdag. På grund af variationer i en arbejdsdag som sygdom, forsinkelser, pauser, pair programming med mere besluttede gruppen, at der i gennemsnit lagde 8 effektive arbejdstimer på en arbejdsdag.

Samarbejde

Her mødte gruppen to store udfordringer. Gruppesamarbejdet, særligt i den første sprint, hvor der blev arbejdet med 3 personers pair programming, hvilket betød, at de ønskede antal timer ikke kunne opnås og derfor at sprint loggens user stories ikke blev nået. Behovet for at samarbejde sluttede ikke efter XP og i højere grad end forventet var der kun 1-2 aktive computere. Til gengæld har det også betydet, at den generelle planlægning af systemet igennem domæne og relationel model har været god og at der er blevet videns delt meget.

Buggs har været en anden stor udfordring i forbindelse med at nå 8 timers effektivt arbejde, da kritiske fejl i systemet kan tage op til flere timer eller dage at rette. For eksempel var UCN's kraka server nede en fredag, som gjorde det vanskeligt at teste systemet. Andre buggs har inkluderet versions kontrol og integrerede fejl, altså fejl i visual studios native klasser.

Afvikling af udfordringer

Når gruppen har stødt på udfordringer er det blevet nævnt til daily Scrum og en primær person er blevet tildelt opgaven med eventuelt støtte, hvis der har været behov for det. Scrums agile tilgang har hjulpet med at omstrukturere prioriteter i takt med, at gruppen havde øget deres viden og kendskab til det distribuerede system og hvilke udfordringer dette giver. Denne omstrukturering kom for eksempel til udtryk i sprint review 2 efter sprint 2. Hvori gruppen på grund af mange udfordringer og den resterende tid til aflevering omprioriterede backloggen og dermed også sprint 3's backlog, så den i højere grad prioriterede de retningslinjer og læringsmål, som er beskrevet i syllabus for 3 semester.

Gruppen har i løbet af sprinten benyttet del-elementer fra flere forskellig udviklings metoder. UP(unified process) til alt planlægning før sprint perioden, det inkludere for eksempel fremstillingen af en domænemodel, relationelle model og mock-up. XP i den første sprint og Scrum i de resterende. Denne blanding af flere forskellige metoder indenfor samme projekt har det resultat, at metodernes fordele og ulemper bedre kan sammenlignes. Det var for gruppen en læringsproces at gå fra at benytte UP, som har været benyttet i to semestre og til at benytte Scrum og XP. Manglen på dokumentation på grund af incremental design betyder, at Scrums metode har en læringsperiode, som har givet udfordring undervejs i projektet. Som eksempel med prioriteringen af product backloggen måtte omprioriteres efter sprint 2.

Delkonklusion

Det konkrete metodevalg som primært er Scrum har givet gruppen værktøjerne til at løse løbende ændringer i systemet. Det har givet gruppen nogle værktøjer og aktiviteter som hjælper med at holde overblikket i udviklingsprocessen og kvaliteten i koden.

Version kontrol og værktøjer

I dette kapitel vil der blive beskrevet, hvilke værktøjer der er blevet brugt til at lave systemet, databasen og diverse diagrammer. Desuden vil der også blive beskrevet, hvad gruppen har brugt til versionskontrol.

Versionskontrol

Til versionskontrol har gruppen benyttet sig af github[github.com], som er en gratis tjeneste, hvis funktionalitet inkluderer; versionskontrol, branche control, og et merger værktøj. Disse funktioner giver muligheden for at arbejde på det samme projekt samtidig.

Versions konflikter opstår i løbet af et projekt af denne størrelse og omfang. Efter gruppen havde afsat en mængde tid til at lære og forstå github kunne disse konflikter udbedres.

Værktøjer

De programmer der er blevet brugt til at udviklet programmet i, er Microsoft SQL der er blevet brugt til databasen og Visual Studio, som er blevet brugt til at programmere selve systemet i. Github er som nævnt anvendt til versionskontrol og UMLet er anvendt til uml diagrammerne domænemodel og relationel model. Derudover er rapporterne skrevet i Word.

De værktøjer som gruppen har valgt, er valgt af to årsager. For det første fordi det er dem, som gruppen har fået undervisning i og for det andet fordi de er branchestandarder. Herunder har gruppen for eksempel valgt at benytte github i stedet for SVN, som gruppen ellers har modtaget undervisning i, fordi github er mere udbredt i branchen og giver derfor den største læring.

Delkonklusion

Versionskontrol har givet nogle udfordringer, som gruppen i løbet af projektet er blevet eksponentielt hurtigere til at løse. De værktøjer der er anvendt, er primært på grund af, at det er dem, der er undervist i og dem som er branchestandarder.

Hovedprincipper i planlægning og kvalitetssikring

Planlægningen for projektet ligger før sprint 1. Derfor kan der argumenteres for, at der i udarbejdelsen af dokumentation er anvendt UP (unified process), fordi dokumentationen ikke er lavet inkrementalt som XP og Scrum ellers dikterer. For at undersøge om problemformuleringen og projektet kan realiseres, har gruppen som nævnt tidligere anvendt en mundtlig diskussion af nøgleressourcer, nøglepersoner og nøgleaktiviteter som det forventes, at projektet kræver. Derudover er FURPS+ begrebet blevet anvendt. Planlægningen er indtil sprint 1 blevet revideret flere gange. Særligt product backloggen er revideret flere gange i forbindelse med nye retninger for programmet.

FURPS+

FURPS+ er et acronym for Functionality, Usability, Reliability, Performance og Supportability. Begrebet anvendes i flere forskellige produktions brancher herunder også software udvikling. Her anvendes det til at bekræfte software kvaliteten i et system. Dette kan også anvendes i præciseringen og planlægningen af et nyt program, som det er i dette projekt.

Functionality
Funktionaliteten i systemet repræsenterer de “ingredienser” gruppen lægger i det færdige program. I overvejelserne omkring hvilken funktionalitet gruppen ønsker, er det primært de mål, som er beskrevet i Syllabus, der er taget i betragtning. Diskussion om funktionalitet repræsenterer altså et ønske om at leve op til de mål, som Teknologi og Programmering formulerer for systemet.

Usability
Systemets anvendelse og brugbarhed er ikke et emne, der er diskuteret i høj grad. Da virksomheds faget ikke længere er på skemaet er overvejelser omkring kundesegment, design og dokumentation omkring brugere af systemet holdt til et minimum.

Alligevel har der været enighed i gruppen om at anvende KISS (keep it simple, stupid) begrebet, dette er gjort for udviklernes egen fordel. Da formålet med projektet er at præsentere, hvad gruppen har lært i løbet af semestret er det bedst at gøre lidt, men godt.

Reliability
Tilgængeligheden for systemet er tænkt som værende en international hjemmeside. Hvis der skulle arbejdes videre på projektet skulle programmet udelukkende være præsenteret på engelsk eller med sprog valgmuligheder på en server, således at alle med internetforbindelse kunne tilgå programmet. Derudover skal systemet være pålideligt. Deraf kommer værdien af test.

Performance
Ydeevnen for systemet skal være effektiv, da systemets primære grænseflade er en hjemmeside, skal en browser kunne håndtere ressourceforbruget, som systemet kræver. Endvidere skal systemet kunne skaleres til at håndtere store mængder data. Dette har været en vigtig overvejelse i designet af databasen. Disse overvejelser er beskrevet i teknologi/programmerings rapporten for systemet. Ydermere skal systemet være skalerbart, det vil sige, at systemet udover information om film og serier også på sigt skal inkluderer information om skuespillere og personerne bag lyd, kamera med mere.

Supportability
Support til systemet skal kunne varetages af andre udviklere end det originale udviklingsteam, da systemet forventes at være langtidsholdbart. Denne overvejelse er ikke understøttet af metodevalg, da det ville være nemmere at videreudvikle systemet, hvis der var yderligere dokumentation.

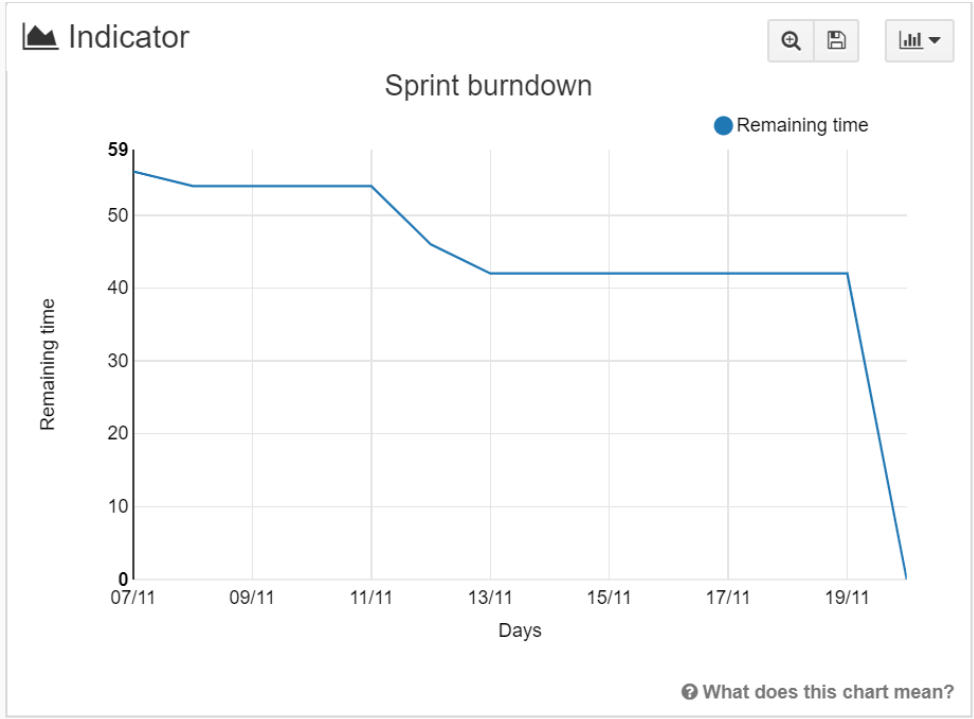
Overvejelser i FURPS emner viser altså, at systemet er planlagt således, at høj kvalitet teoretisk kan opnås. Der er i gennemgangen af FURPS emner ikke fundet store mangler eller uopnåelige mål for systemet.

Process

Sprint 1

Vi arbejdede i sprint 1 efter XP udviklingsmetoden (se XP), hvor vi tester først. Eftersom det er et læringsprojekt og første gang vi udvikler et distribueret system, gennemgik vi mange udfordringer i sprint 1 - den største var manglende konkret know-how i forhold til C# og arkitekturen i det distribuerede system. Udfordringen overraskede gruppen fordi den teoretiske forståelse for arkitekturen var på plads.

Løsningen var at gøre brug af faglige ressourcer (vejleder og lærebøger/videoer) og da vi gik ind i sprint 2 er forståelsen i højere grad på plads, men forståelsen har forsinket projektet, hvilket betød at sprint loggen for sprint 2 primært var udgjort af user stories, som var en del af sprint 1. På nedenstående billede kan vores burndown chart ses på figur 6, hvori det også kan ses at vi nåede cirka 50% af den forventede arbejdsmængde.



Vi lærte i sprint 1, at vores estimeringen var sat for højt i forhold til vores nuværende fælles arbejdstempo.

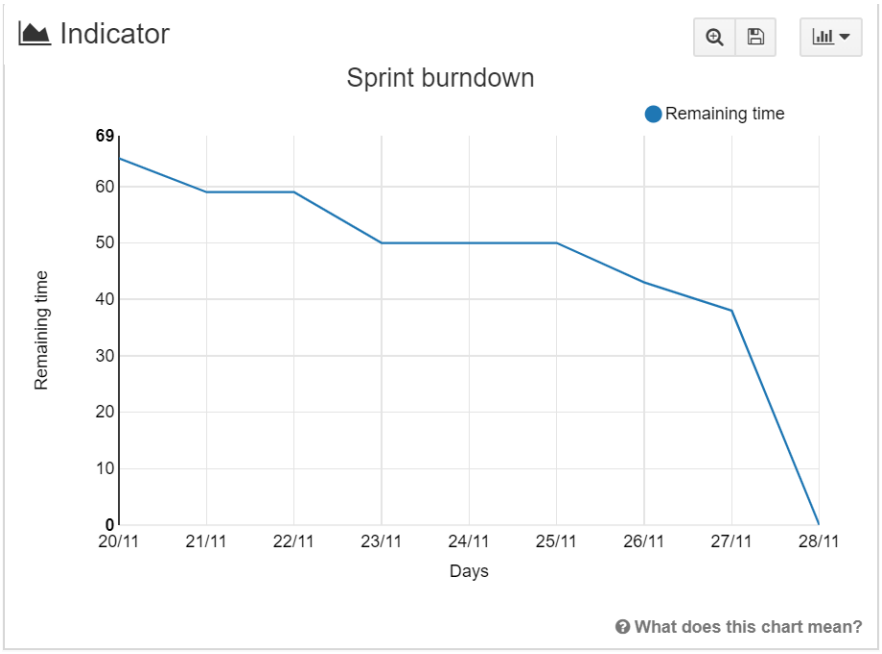
Sprint 2

Sprint 2 var det første sprint, hvor gruppen arbejdede med Scrum. Dette betød at det ikke længere var nødvendigt at par programmere.

UCN's netværk var nede fredag d. 23 november i starten af dagen og UCN's kraka server var nede hele dagen. Dette gjorde det svært for gruppen at teste programmet og var en faktor i, at gruppen ikke nåede den planlagte sprint log.

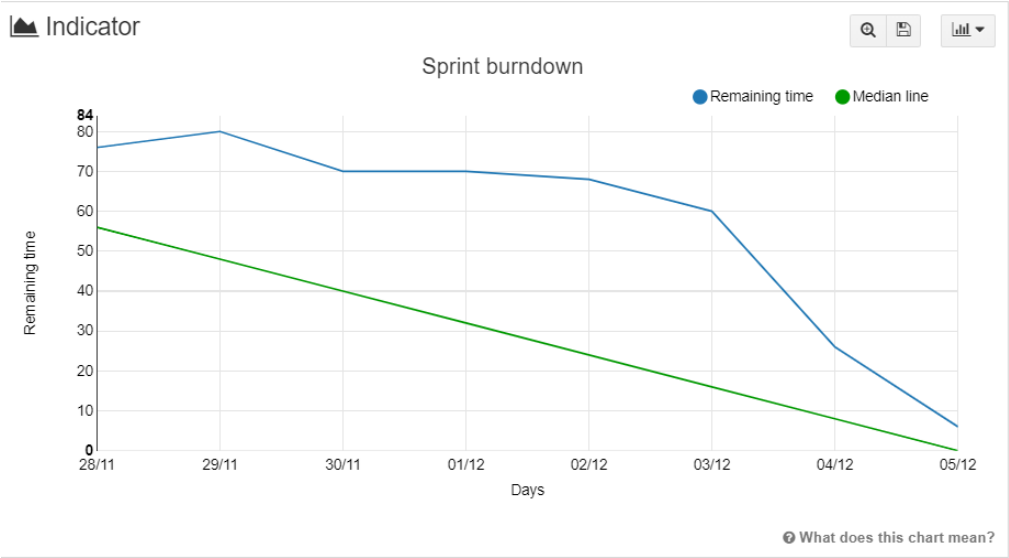
I sprint 2 besluttede gruppen at lave en Spike 'findAllEntertainment', som havde til primært formål at øge gruppens viden om det distribuerede system.

På grund af gruppens viden om det distribuerede systems arkitektur tog spiken "findAllEntertainments" længere tid end forventet og der var ikke plads til mange user stories ud over denne spike. Burndown chartet kan ses på figur 7



Sprint 3

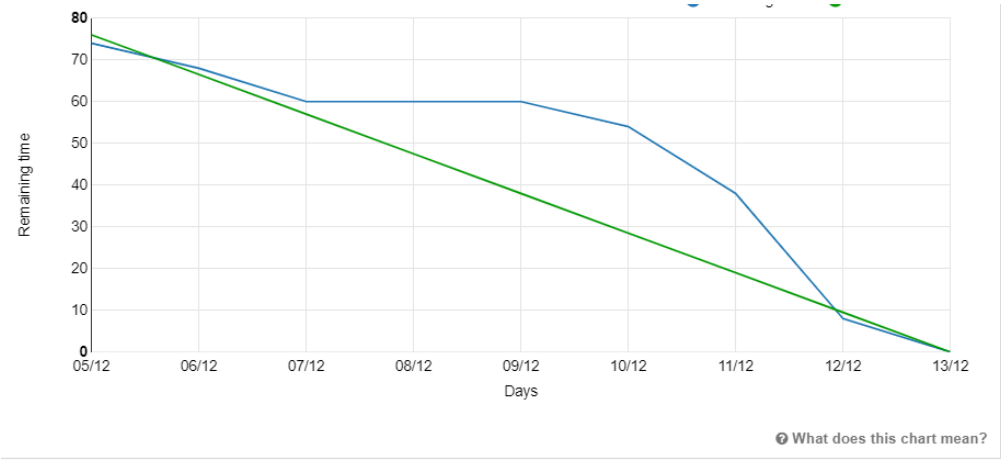
I sprint 3 kan deadlinen 17. december mærkes og gruppen vælger at omstrukturere både WCF solutionen og MVC web solution baseret på den udvidede forståelse af det distribuerede system, som gruppen havde opbygget i løbet af sprint 1 og 2. Omstruktureringen gav gruppen et bedre overblik over rækkefølgen, som gruppen skulle tilgå de tilbageværende user stories. Sprint 3 har på trods af dens korte periode (4 dage) været den mest produktive for gruppen figur 8.



Sprint 4

Det fjerde sprint er det sidste officielle sprint, der er dog et sprint 5, som ligger efter sprint review den 14. december til og med den 17. december altså to hverdage og to weekends dage. De mål der var sat til sprint 4 blev ikke nået fuldt ud. Dette betyder at gruppen re-prioriterer de tilbageværende user stories. Som på grund af deadline begrænses program delen af systemet, da det vurderes, at der skal afsættes tre dage til det resterende rapportskrivning. Samt de rettelser og opsætnings arbejde, som ligger i rapporterne.

I sprint 4 midt på ugen valgte gruppe at unplan fem tasks, da det blev vurderet, at disse tasks ikke kunne blive lavet færdigt i dette sprint, så disse tasks er blevet overført til sprint 5. Dette kan ses på figur 9.

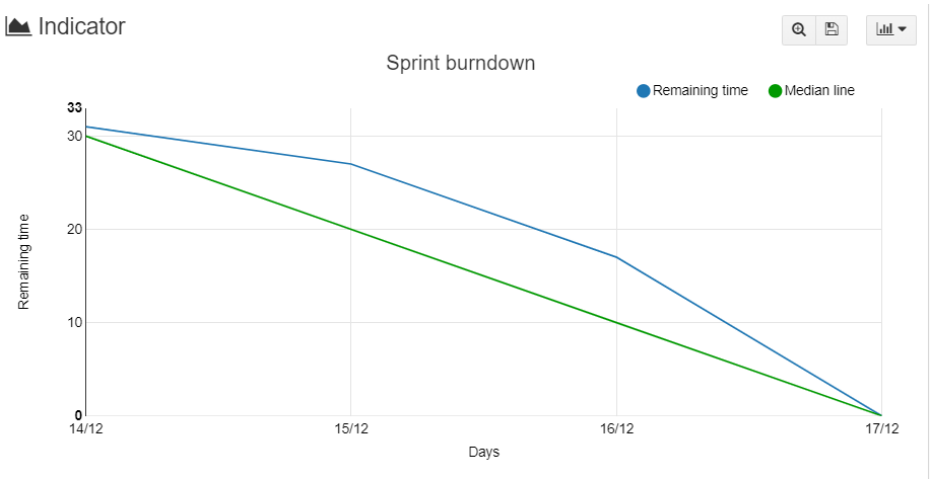


Sprint 5

I sprint 5 tog gruppen de tasks, som blev unplanned i sprint 4, ind i dette sprint. Denne sprint er også kortere end de andre sprinter har været, eftersom denne sprints varighed har været to arbejdsdage.

I dette sprint valgte gruppen desuden også at have nogle længere arbejdsdage, da gruppen gerne ville være sikker på at kunne nå de mål gruppen have sat sig for at nå i dette sprint.

Nogle af de udfordringer var at få lavet Login, så der kunne komme en bruger op fra databasen. Ved at få løst dette, blev det der var 'hardcoded' også løst, da programmet nu havde en fornemmelse af, hvad en bruger er. Dette kan ses på figur 10.



Generelt

Gruppen bliver allerede fastlagt den første dag i semesteret (01/09/2018).
Gruppen laver to mini rapporter i teknologi, som kan omskrives og anvendes i den endelige rapport. Derudover laver gruppen idegenerering og user stories til projektet, som bliver revideret flere gange både før og i sprintene.

Den første sprint, Sprint 0 ligger den 24/10/2018 og er på kun 1 dag. Scrum metoden kan ikke anvendes på denne dag på trods af, at den hedder "sprint". Der blev ikke forudgående lavet nogen sprint log og der blev ikke efterfølgende holdt et sprint review.

Sprint 1 starter den officielle skriveperiode den 07/11/2018. Fra den dato er der fire sprint med

dertilhørende Scrum begivenheder. De fire sprints bliver udgjort af 22 hverdage, da weekender, sprint review og undervisning stadig optager nogle dage fra d. 07/11/2018 til den 17/12/2018. Den gennemsnitlige sprint har altså været 5.5 hverdage. Dertil kommer der et sprint review på ca. 3-4 timer. Dette overskrider den anbefalede sprint review tid fordi hele klassen fremlagde deres sprint review og observerede de andre gruppers fremlæggelse, derudover deltog undervisere. På grund af deres vejledning kan de betragtes som nøgle samarbejdspartnere og Scrum master.

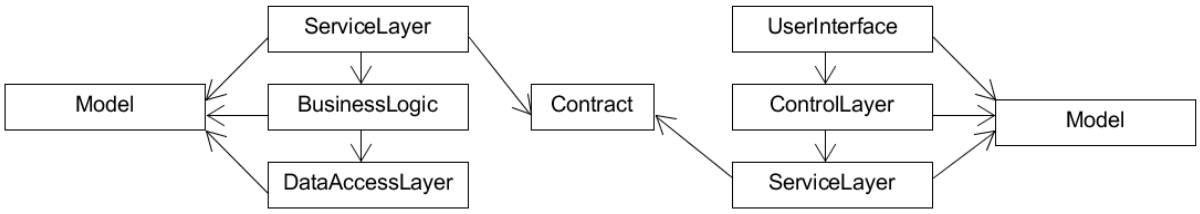
Delkonklusion

Processen er i projektet gået optimalt. Der har været mange udfordringer som har taget gruppens tid, men fordi gruppen har haft de rigtige redskaber er næsten alle udfordringer blevet løst undervejs. Gruppen nåede dog ikke deres oprindelige mål med hensyn til funktionalitet i programmet.

Kvalitetskriterier og arkitektur

Arkitektur

Arkitekturen i systemet har i dette semester været det distribuerede system ses på figur 11.



Det har haft den betydning, at projektet har flere dele end, gruppen har været vant til. Sidste semester havde projektet en struktur som hed Database → Database lag → model lag → kontrol lag(business logic) → Graphic user interface. I dette semester eksistere denne struktur stadig og er repræsenteret i gennem gruppens server solution. Som det kan ses fra billedet kommunikerer denne server solution med klientens solutions gennem en datacontract. Arkitekturen i det distribuerede system er til for, at nemt kunne udskifte klienter eller databaser og videreudvikling af systemet er gjort nemmere.

Risikostyring

Scrum og i særdeleshed XP inkluderer processer, som skal sikre kvaliteten i systemet og derigennem sænke risikoen for fejl. I XP arbejder gruppen ved hjælp af pair programming, som hvori observatørens ansvar er at styrke kvaliteten og bevare overblikket over arkitekturen. På grund af dette sænkes risikoen for fejl. Ydermere arbejdes der med test first princippet, som garanterer, at der er virkende test samtidig med at programmet opbygger funktionalitet.

I Scrum kan der også arbejdes med pair programming, særligt i de krævende funktioner kan dette være nødvendigt for at nedsætte risikoen for fejl. Scrums detaljerede user stories bidrager også til en lavere risiko, fordi desto mere detaljeret en user story er, des mindre er risikoen for usammenhængende dele af systemet. Her spiller prioriteringen af user stories også en rolle, fordi man kan have dependent user story, altså user stories, som er afhængige af, at andre elementer i systemet er lavet. Der vil altid før eller siden opstå fejl i en udviklingsprocess, men gruppen har begrænset fejl ved hjælp af spikes, pair programming og detaljerede user stories.

Delkonklusion

Risikostyring er primært gjort igennem pair programming, test first og detaljerede user stories. arkitekturen er opstillet efter det distirbuerede system og inkluderer en database, server og to klients.

Refleksioner over metoder og deres anvendelse i praksis

Gruppen har valgt en agil tilgang til løsningen af deres problemformulering. Der er anvendt Scrum og XP til programmering. Her er XP dog kun benyttet til sprint 1. Dette har betydet en langsom udviklingsproces i sprint 1 fordi XP benytter pair programming. Ydermere er det de basale opgaver, der er løst i sprint 1, for at opbygge en struktur der lægger op til de mere krævende og avancerede funktioner i programmet. I retrospektiv ville benyttelsen af XP og pair programming have været bedre placeret i sprint 3 eller 4, da det er her de avancerede funktionaliteter er programmeret. Derudover er XP en arbejdsmetode, som er svær at mestre fordi man tester først (se XP). Gruppen har i første- og andet semester været vant til at teste sidst eller undervejs.

Scrum har været gavnligt for gruppen særligt anvendelsen af user stories og daily Scrum har bidraget til, at gruppen har haft en forståelse for, hvem der laver hvad og en fælles enighed om hvad præcis en opgave/user story indebærer. Gruppens størrelse har gjort det svært at lave en meningsfuld rollefordeling. Det har betydet at rollerne Scrum master, product owner og udviklingsteam og deres tilhørende ansvarsområder har tilhørt alle gruppemedlemmer på det ene eller andet tidspunkt.

Delkonklusion

Metodevalget for gruppen har alt i alt været gavnligt primært på grund af den fleksibilitet, som det giver og har samtidig været et naturlig valg (Se de fem akser). Ved en videreudvikling af systemet, for eksempel betalende brugere, og ved en tilføjelse af flere udviklere kunne et mere plan drevet udviklingsmetode have været anvendt.

Perspektivering

I perspektivering vil gruppen overveje, hvilke andre værktøjer eller arbejdsmetoder, der kunne have været inkluderet for at hjælpe gruppen med at besvare problemformuleringen. Gruppen har haft et godt udbytte af at lave en spike i sprint 2, som havde til formål at øge gruppens forståelse af strukturen i et distribueret system. Flere spikes kunne have været anvendt for at opdele de distibuerede system i mindre dele. Dette ville sandsynligt have haft den effekt, at gruppens forståelse og effektivitet i arbejdet med det distibuerede system havde været bedre.

Gruppen kunne også have benyttet andre agile udviklingsmetoder som for eksempel kanban, hvor det primære formål er at balancere efterspørgslen med produktions rådigheden. Dette betyder at arbejdsmængden ikke overstiger den forventede produktion og efterspørgslen. At benytte 'Kanban' kunne i dette projekt have øget effektiviteten for dermed at opnå et produkt med højere kvalitet.

Konklusion

Projektets primære formål var først og fremmest at besvare problemformuleringen som havde til formål at afdække det materiale som gruppen havde fået undervisning i, i løbet af semesteret. Med henblik på problemformuleringen og dens underspørgsmål er det lykkedes gruppen at lave et distribueret system med to klienter ved hjælp af forskellige udviklingsmetoder og de værktøjer som de stiller til rådighed. Trods de udfordringer som gruppen er stødt på undervejs. Er det endelige produkt indenfor de fremstillede rammer, som gruppen havde sat op i deres planlægning.

Litteratur

Bøger:

Larman, Craig, Applying UML and Patterns, an introduction to Object-oriented analysis and design and the Unified Process, third edition

Henrik Kniberg & Mattias Skarin, Kanban and Scrum, making the most of both

Sommerville, Ian, Software Engeneering, 10. edition, Pearson, 2015

Kniberg, Henrik, Scrum and XP from the Trenches, 2. edition 2015

Cohn,Mike, User Stories Applied, For Agile Software Development Addison-Wesley

Miles, Rob, C# Programming: http://www.csharpcourse.com/ DK-students

Stallings, William, Operating systems: Internals and design principles, 8th edition

Hjemmesider:

http://www.agileadvice.com/2013/06/07/referenceinformation/the-rules-of-scrum-the-scrum-team-is-no-less-than-five-people-and-no-more-than-eleven-people/ (Sidst set den 11/12 2018)

https://www.scrumguides.org/scrum-guide.html#uses (Sidst set den 11/12 2018)

https://www.ibm.com/developerworks/rational/library/4706.html (Sidst set den 12/12 2018)

http://www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/1989-04.pdf (Sidst set den 12/12 2018)

http://www.agileprojectmgt.com/docs/Boehm20.pdf (Sidst set den 12/12 2018)

https://www.interaction-design.org/literature/article/kiss-keep-it-simple-stupid-a-design-principle (Sidst set den 17/12 2018)