|  |  |  |  |
| --- | --- | --- | --- |
|  | 11/20/2012 |  | |
|  | |  | |
| Rapport Prosjektarbeid1  *Denne Rapporten er inneholder dokumenter for faget IT1901, prosjektarbeid 1.* | | | |
|  |  | |  |
|  |  | | Gruppe 10, Tor Økland Barstad, Lars Erik Græsdal-Knutrud, Svenn K, Per Øyvind Kanestrøm og Jørgen Rugelsjøen Wikdahl |

Innholdsfortegnelse

Introduksjon 1

Om faget, om oppgaven og om gruppen 1.1 3

Beskrivelse av Prosjektet 1.2 3-4

Valg av prossesmetode 1.3 4

**Tidsestimering 2** 4-5

**Prosjektorganisering 3** 6

Ansvarsområder 3.1 6

Medlemenes kunnskapsområder 3.2 6-8

**Arkitektur forklaring 4** 8-9

**Risiki i prosjektet 5** 9

Hva slags risiko står man ovenfor i et prosjekt? 5.1 9

Splitting av gruppen 5.2 10

Uforutsette utfordringer 5.3 10

Deadlines 5.4 11

Misting av arbeid 5.5 11

Medlemmer som ikke møter 5.6 11

Kommunikasjon 5.7 11-12

Prosjektplan feiler 5.8 12

Sykdom 5.9 12

Forsentkomming 5.10 12

Avveining mellom forhåndsregler og effektivitet 5.11 12

**Produktbeskrivelsen 6** 13

**Prossesbeskrivelsen 7** 14

Sprint 7.1 14

Arbeids fordeling 7.2 15

Møter 7.3 15

Kildekodedeling 7.4 15-16

Facebook 7.5 16

**Systemdesign 8** 16

**Systemfunskjoner 9** 16

**Vektøy 10** 17-18

**Evaluering 11** 18

Hva gikk bra? 11.1 18

Hva gikk dårlig? 11.2 18

Lærdom til neste gang 11.3 18

**Kap.1 Introduksjon**

**Kap 1.1 Om faget, om oppgaven og om gruppen**

Faget *IT1901 - Informatikk prosjektarbeid 1* handler om gruppearbeid innen IT og utforming av prosjekter i arbeidslivet. En viktig del av dette er forholdet til kunden.  
  
Hovedmålene i dette faget er å gi studentene en forståelse for samspillet mellom produkt og prosses-orientert utfordringer og aktiviteter i programmeringsprosjekter. En viktig del av dette faget er å tilegne seg praktisk og teoretisk kunnskap om hvordan prosjektarbeid innen IT burde foregå, med henhold på bl.a. arbeidsmetoder.  
  
Da et av hovedmålene i dette faget er sammarbeid i gruppe, ser vi ikke problemet med at enkelte på gruppen har større kompetansenivå enn andre, siden vi som gruppe skal løse oppgaven sammen og ikke individuelt.   
  
Oppgaven har vært å lage et system som sauebønder kan bruke til å registrere og administrere sauene sine.  
  
Vår gruppe 10 består av 6 informatikk studenter som alle studerer BIT, Jørgen Rugelsjøen Wikdahl, Anders Sildnes, Per Øyvind Kanestrøm, Svenn K, Lars Erik Græsdal-Knutrud, Tor Økland Barstad. Det tok ikke langt tid før første gang hvor vi alle fikk samlet oss, og det gikk enda raskere å opprette en felles Facebook-gruppe for prosjektet. Alt i alt var vi godt fornøyd med gruppen og kom fort i gang med prosjektet.

**Kap 1.2 Beksrivelse av prosjektet**

Vi “later som” at vi jobber som programmerere i et programvareselskap der vi blir kontaktet av en sauebonde. Sauebonden har allerede digitale sendere til å feste rundt halsen til sauene og det er allerede fastsatt hvordan disse fungerer (hva slags data vi får fra dem og hvor ofte, osv). Fra disse sendere får vi tilsendt posisjonen til sauene og info om sauens hjertefrekvens tre ganger i døgnet og hver gang et angrep registreres. Bonden ønsker vår hjelp til å utvikle et system som ved hjelp av disse senderne hjelper bønder med å registrere, administrere og holde oversikt over sauene sine, bl.a. med henhold til rovdyrangrep. Dette systemet skal være brukervennlig for bøndene det er laget for, som ikke nødvendig har noen spesiell kompetanse innen IT eller PC-bruk. Vi får i oppgave å komme med forslag til et slikt system, basert på analysen vi har gjort i samarbeid med kunden, for å så utvikle dette systemet.  
  
Prosjektoppgaven vår er basert på et ekte prosjekt, men i vårt prosjekt skal vi lage en simulator som simulerer beskjedene vi får fra sendere som våre fiktive sauer har rundt halsen. I prosjektet vårt spiller studass på et vis spiller rollen som kunden/bonden vi utvikler systemet til.

**Kap 1.3 Valg av prosessmetode (beskrivelse)**

Vi valgte å bruke Scrum-metoden, slik vi ble anbefalt. Såvidt vi husker ble det sagt at studenter som hadde kjennskap til andre prosessmetoder som de heller ville velge hadde muligheten til det, men ingen av oss hadde noe særlig erfaring med prosessmetoder, så vi diskuterte aldri å velge noen annen prosessmetode enn den det ble lagt opp til.  
  
Vi fikk representert Scrum av faglæreren i en av de første forelesningene som ble holdt, men lette også opp info på egenhånd. Ingen av gruppemedlemmene hadde erfaring med Scrum og vi brukte noe tid i starten på å bli kjent med denne arbeidsmetoden og venne oss til å gjøre ting på Scrum-vis.  
  
Vi startet først med å bruke scrumtavle i Trello, men byttet fort over til iceScrum. Der hadde vi en oversiktlig tavle hvor vi fikk oversikt over alle oppgaver som lå til behandling og var under arbeid, og hva som måtte prioriteres til neste gang. Denne metoden brukes også av mange IT-selskaper verden rundt når de jobber tett med kunder, siden kunden da lett kan følge med på prossesen gjennom scrumtavlen, komme med innspill og være med gjennom hele prosjektet. I starten var vi veldig dårlige til å oppdatere trello-boardet, men da vi byttet om til iceScrum ble ting lettere. Rundt slutten av sprint 2 var vi blitt mye flinkere til å oppdatere boardet.

**Kap 2. Tidsestimering**

I dette prosjektet har gruppen måttet lære seg nye grensesnitt og modeller å jobbe etter. Ingen på gruppen hadde tidligere hatt prosjektarbeid i IT. Tidligere studenter hadde heller ikke hatt samme type oppgave som det vi fikk, så selv ikke de kunne komme med råd for å hjelpe oss. Tidsestimeringer ble med andre ord vanskelig å gjøre nøyaktig. Til og med kunden vår måtte innrømme å ha bommet litt i tidsestimatet for prosjektet; kravspesifikasjonen ble krympet etter en måned.  
  
Vi startet prosjektet vårt med å lage en «Work Breakdown Structure»[[1]](#footnote-1), WBS, til prosjektet. Denne laget vi i form av en liste over ting som måtte bli gjort innen prosjektet hadde deadline. Dette ga oss en oversikt som skulle hjelpe oss å få en anelse over hvor langt vi var i prosjektet til enhver tid. På denne måten kunne vi også bli hjulpet til å lage realistiske tidsestimeringer senere i prosjektet. Det kan for eksempel være ønskelig å bruke mye tid på å lage ekstra funksjonalitet til databasene, men dersom man ser at vi i forhold til WBS-listen vår ligger litt bak i prosjektet, kan det hende man innser at annen funksjonalitet må prioriteres først.  
  
I de første sprintene valgte vi å gjøre tidsestimater felles, slik SCRUM anbefaler. Vi gjorde opp estimater i antall timer, men valgte ikke å dele inn i vanskelighetsgrad. Behovet for å inndele etter vanskelighetsgrad var ikke tilstede i og med at de fleste på gruppen viste en generell interesse i alle oppgavene som ble oppgitt. Vanskelige oppgaver og andre oppgaver ble dermed sidestilt.  
  
Tidsestimatene viste seg etterhvert å være unøyaktige, og de ble i økende grad vanskeligere å gjøre ettersom kompleksiteten til oppgaven var ukjent for resten av gruppen. Derfor ble det til at vi gikk over til å selv gjøre tidsestimat innenfor de oppgavene vi gjorde selv.  
  
En periode ut i prosjektet ble det klart for oss at vi hadde mye resterende arbeid, og det ble i tillegg satt vekt på at ikke alle medlemmene på gruppen gjorde like mye arbeid i hver sprint. Vi bestemte oss derfor for en regel om at vi skulle lage hver sprint til å ha 120 arbeidsstimer, eller 8 timer per person hver uke.  
  
Idéen virket god, men det ble også til at flere på gruppen ikke klarte å holde seg til tidsrammene sine. Sprintene måtte heretter forskyves med flere dager.

**Kap 3. Prosjektorganisering**

**Kap 3.1 Ansvarsområder**

Helt til å begynne med identifiserte vi hva de forskjellige ansvarsområdene ville være. Ansvarsområdene vi kom frem til var GUI, databaser, dokumentasjon, kart og prosjektets fremdrift (Scrum-master). At man ansvar for noe betyr ikke at man må gjøre det alene, men det betyr at man har ansvar for å passe på at det blir gjort.  
  
Vi gikk på rundgang med å si tankene våre om å ha de forskjellige ansvarsområdene vi hadde identifisert, både med tanke på erfaringen og kompetansen vi hadde og med tanke på hvordan vi tror vi ville likt å ha et slikt ansvarsområde. Basert på dette fordelte vi ansvarsområdene.  
  
Det ble til at de som hadde mer kompetanse enn andre innen spesielle områder (se kap. 3.2) fikk ansvaret for disse. Dette var noe alle syntes virket fornuftig, inkludert gruppemedlemmene som tok på seg de gjeldene ansvarsområdene.  
  
Ingen hadde spesielt lyst på rollen som Scrum-master, men flere av oss sa at vi kunne leve med rollen og prøve å fylle den på en god måte hvis den ble tildelt oss. Blant tingene faglærer nevnte som Scrum-masterens roller er å sørge for at det er kaffe på møtene. På møtet hvor Scrum-master ble bestemt ble dette humoristisk påpekt med henhold til at Per Øyvind allerede hadde sørget for at det var kaffe. Også generelt trodde vi Per Øyvind kunne egne seg godt som Scrum mester. Per Øyvind gikk med på å være Scrum-master og det skulle vise seg at vi hadde rett.  
  
Selv om Per Øyvind var Scrum-master tok han også med-ansvar innen andre områder som kart og GUI. Per Øyvind fylte ansvaret som Scrum-master slik han skulle. Det betyr ikke at han er den eneste som har tatt initiativer, men slik skal det heller ikke være.

**Kap 3.2 Om de forskjellige medlemenes kunnskapsområder**

Tor Økland Barstad  
Alder: 22  
Hjemsted: Ås  
Ansvarsområde: Progammering  
  
Kvaliteter: Tor har vist god forståelse for programmering innen Java og objektorienterte metoder. Han hadde også litt erfaring med klasse- og ER-diagrammer og har vist god vilje til å tilegne seg kunnskap om andre typer diagrammer som vi har hatt bruk for. Sammen med Jørgen har han stått for det meste av diagramproduksjonen i starten av prosjektet.  
  
  
Anders Sildnes  
Alder:   
Hjemsted:  
Ansvarsområde: GUI  
  
Kvaliteter: Anders har en bred bakrunn innen web- og it-utvikling og har gjennom sine engasjement i ITK Samfundet og IUG NTNU tilegnet seg nyttige programmeringskunnskaper som gruppa har kunnet nyte godt av. Han har en grundig forståelse for hvordan man programmerer effektivt og har erfaring fra Java såvel som AS3 og C++ fra før. Han har også hatt erfaring med bruk av Qt4 så han fikk ansvaret med å utvilke det grafiske brukergrensesnittet. Dette har han gjort på en strålende måte.  
  
  
  
Per Øyvind Kanestrøm  
Alder:  
Hjemsted:  
Ansvarsområde: ScrumMaster. Kart.  
  
Som ScrumMaster har Per Øyvind fungert veldig bra. Han har vist gode evner til å kommunisere ikke bare med gruppa, men også med «kunde» (stud.ass/fagstab). Han har god evne til å holde oversikten over hva som må gjøres og får folk med seg på det. Som medansvarlig for kartdelen har han vist en god evne innenfor “research” til å finne metoder og klasser som kan brukes. Erfaring fra C#, Silverlight og Windows Forms har også gitt en god bakgrunn for å kunne sette seg inn i kode og har gitt ham en bra forståelse for hvordan et program er bygget opp.  
  
  
Lars Erik Græsdal-Knutrud  
Alder:  
Hjemsted:  
Ansvarsområde: Database  
  
Kvaliteter: Lars Erik har vært uvurderlig for hvordan gruppa har klart å produsere kode. Han har vist en stor vilje til å jobbe med koden og har bred erfaring fra systemutvikling, SQL-databaser og generell programmering. Han har vært et arbeidsjern som har stått på og programmert mye av både databasedelen og logikken som ligger til grunn for programmet. Han hadde også bra forståelse for ER- og UML-modellering.

Jørgen Rugelsjøen Wikdahl  
Alder:21  
Hjemsted:Trondheim  
Ansvarsområde: Dokumentasjon  
  
Kvaliteter: Jørgen har bidratt med god vilje til å jobbe og lære, og har sammen med Tor hatt ansvar for forskjellige diagrammer. Han har også hatt overoppsynet med dokumentasjon og raportskriving, og har vist evne til å få folk til å gjøre det som må gjøres for at prosjektet skal komme i havn. Grunnleggende erfaring i Java og objektorientering, samt litt erfaring med databaser har kommet godt med utover i prosjektarbeidet.  
  
  
Svenn Kvelstad  
Alder: 33  
Hjemsted: Trondheim  
Ansvarsområde: Kart  
  
Kvaliteter: Svenn hadde erfaring med prosjektarbeid og rapportskriving før i tillegg til grunnleggende kunnskaper innen Java og objektorienterte metoder. Svenn tok på seg oppgaven med å integrere kart inn i systemet, og var nødt til å sette seg inn i hvordan dette fungerte, siden ingen hadde erfaring med å integrere kart inn i IT-systemer fra før. Han har også driftet nettsider og jobbet med grunnlegende databaser som kom godt med i prosjektet.

**Kap 4 Arkitektur-forklaring**

under systemdesign

Front end

Programmet er skrevet i programmeringspråket Java. Vi har forsøkt å implementert det i MVC stil.   
  
Egentlig skulle dette prosjektet bli helt plattformuavhengig, men på grunn av et teknisk valg på bruk av en Java Swing komponent for kartvisning i programmet så er programmet avhengig av Windows plattformen.  
  
Qt Jambi krever også at Java versjonen som blir brukt er 32bit.  
  
Den grafiske delen er håndtert Qt Jambi, ett Java bibliotek med bindinger mot C++ biblioteket Qt.  
  
Kravspesifikasjonen er altså;  
- Java 32bit  
- Windows 32/64 bit  
  
Kun testet på Windows 7 32bit og Windows 8 64bit.

Back end

Databaser + server

**Kap 5. Risiko i prosjektet**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Sansynlighet | | | | | |
| Alvorlighetsgrad |  | Svært liten | Liten | Moderat | Stor | Svært stor |
| Kritisk | Misting av arbeid | Misting av arbeid |  |  |  |
| Alvorlig  Splitting av gruppa | Frafall av gruppemedlemm  Frafall av gruppemedlem |  |  |  |  |
| Moderat |  | Kommunikasjonssvikt | Store uforutsette utfordringer | Store uforutsette utfordringer |  |
| Liten |  | Sykdom |  | Forsentkomming |  |
| Ubetydelig |  |  |  |  |  |

**Kap 5.1 Hva slags risiko står man ovenfor i dette prosjektet?**

Det er masse som kan gå galt i et prosjekt som dette, men noen “farer” er viktigere å tenke på enn andre, deriblant kommunikasjonsfeil, sykdom, at noen faller fra og misting av arbeid. Kanskje er det også slik at splitting av gruppen noe man burde vurdere som en viktig risiko, men vi oppdaget ingen tilløp til konflikt.

**Kap 5.2 Splitting av gruppen**

Hvis gruppen vår hadde opplevd problemer som splitting så hadde dette mest trolig vært et tegn på samarbeidsproblemer og kommunikasjonsfeil, noe som kan unngås ved å håndtere uenigheter og diskusjoner på en god måte og generelt kommunisere bedre. Imidlertid har vi har verken sett noen antydning til splitting av gruppen eller tiløp til konflikter. Dette er kanskje fordi vi har vært flinke til å kommunisere og generelt sett har kommet godt overens. Hvis vi hadde hatt konflikter eller tiløp til konflikter innad i gruppen hadde vi først prøvd å løse disse ved å snakke sammen om dem på en konstruktiv måte, men som nevnt har ikke dette vært en aktuel problemstilling for oss.

**Kap 5.3 Uforutsette utfordringer**

/\*Siden dette er en ganske viktig risiko syntes jeg det virket riktig å opprette et eget delkapittel for den, selv om den riktig nok kan puttes under Kap 5.4 Deadlines og Kap 5.8 Prosjektplanen feiler. Hva mener du?.\*/  
  
En risiko som både kan få store negative konsekvenser og har en sannsynelighet for å ramme et IT-prosjekt er at man støter på utfordringer for å gjennomføre en oppgave som man ikke hadde forutsett eller tatt med i beregningene, eller at en oppgave viser seg å være mye mer utfordrende enn antatt. Denne risikoen var ekstra stor for oss siden vi hadde så liten erfaring fra før og gjorde så mange av oppgavene for første gang.  
  
Vi hadde ventet at det ville være uforutsette utfordringer som ville ta av tiden vår og fikk rett i dette. Eksempelvis hadde vi lenge problemer med å få kartet til å fylle hele ruta det ble tildelt. /\*Putt gjerne inn flere eksempler dere kjenner til.\*/  
  
/\*Eksempel med overføring av objekter med pekere(Dette kommer senere)\*/  
  
Det er kanskje mer vi kunne gjort for å beskytte oss mot uforutsette utfordringer. En ting vi kunne gjort, som vi ikke gjorde i stor grad, er å forhøre oss med individer som har gjort lignende prosjekter tidligere (f.eks. studenter som har hatt faget før oss, selv om de riktig nok ikke hadde samme oppgave) og individer som har gjort lignende oppgaver, men det ville tatt tid om vi skulle gjort det på en aktiv og gjennomgående måte. En ting vi bevisst gjorde var å prøve å bli ferdig med ting så fort som mulig. Målet har ikke vært å være ferdig med prosjektet vårt innen fristen, men å være ferdig i god tid på forhånd. Da er man mindre sårbar for uforutsette utfordringer, og selv hvis man i liten grad støter på uforutsette utfordringer får man mer tid til finpussing og bug-testing av programmet og til rapportskriving.

**Kap 5.4 Deadlines**

Som faglærer advarte om forekom det, spesielt under de første sprintene, at vi overvurderte hvor mye vi kunne få gjort og undervurderte hvor lang tid ting ville ta. Dette har nok ikke bare å gjøre med uforutsette utfordringer (som vi skrev om i kap 5.3) men også det faktum at man generelt har en tendens til å undervurde hvor lang tid tar selv hvis ingen uforutsette utfordringer dukker opp. Vi er jo også studenter som har andre fag hvor deadlines er innvolvert og flere av oss har også ting på siden av studiene, men vi har prioritert dette prosjektet høyt, og tatt fristene alvorlig selv om de har blitt satt av oss selv.  
  
Som anbefalt rundet vi opp når vi var i tvil om hvor lang tid vi antok at noe ville ta, men likevel skjedde det fra tid til annen at vi ikke rakk deler av oppgavene innen deadlinsene.

**Kap 5.5 Misting av arbeid**

I prosjektet vårt brukte vi Git til å koordinere og lagre arbeidet vårt, og pushet arbeidet vårt ut dit jevnt og trutt. Dette hjalp oss å koordinere arbeidet vårt, men det gjorde også at arbeidet var trygt hvis noe skulle skje med en eller flere av filene vi hadde lagret på våre egne maskiner (som skjedde ihvertfall en gang). Hvis noen pushet en ødelagt versjon av filene våre opp på Git kunne vi alltids hente opp eldre versjon og på den måten unngå tap av alt for mye arbeid.  
  
Når vi enda ikke hadde en fungerende versjon av programmet jobbet vi kun i en “branch”. Det forekom i denne lange perioden at vi måtte gå tilbake til tidligere versjoner av ting, men vi klarte oss bra med alt det Git automatisk gjør for å lagre tidligere versjoner av arbeidet vårt. Så snart vi hadde en fungerende versjon av hele systemet ble denne versjonen lagret som en “branch”, mens noen av oss jobbet med forandringer og oppdateringer på andre brancher. Slik unngikk vi å ødelegge den fungerende versjonen. Så snart vi begynte med “branching” sørget Lars Erik for å lære seg det og deretter lære det bort til resten av oss. /\*Dette er bare å forandre på dette hvis noen av dere mener det er feil eller upresist, hva mener du?.\*/

**Kap 5.6 Medlemmer som ikke møter**

Dette har ikke vært noe problem for oss. Alle har satt av tid til å møte opp på møter og møtt opp hvis ikke annet har blitt meldt ifra. I de tilefellene der personer på gruppen ikke har møtt opp har den aktuelle personen hatt en god grunn og sagt fra i god tid. Det har heller aldri vært noe problem å få kontakt med noen i gruppa utenom møter når vi har ønsket det.

**Kap 5.7 Kommunikasjon**

Noe som kan slå feil når man jobber på gruppe er at man på grunn av mangel på god kommunikasjon gjør oppgaver på nytt eller på feil måte. En ting vi har gjort for å unngå dette er såklart å ha møter. En annen ting vi har gjort er å arbeide mye sammen, slik at det bare er å spørre hvis man lurer på noe mens man arbeider. Vi har også gjort god bruk av klassediagrammer for å sørge for at vi har en noen lunde lik forståelse av hva vi skal lage og hvordan det skal henge sammen.

**Kap 5.8 Prosjektplanen failer**

**Kap 5.9 Sykdom**

Tidlig ut i prosjektet vårt skrev Svenn: *“Er det noen som kan bli med meg å finne ut hvordan vi kan integrere Kartverkets karttjeneste i programmet vårt? Jeg kjenner jeg trenger en til til å være med og grave, og så er det sikrere å ha to på det i tillfelle jeg blir syk.”* Senere skjedde nettopp dette: Svenn ble syk i en kort periode. Itilegg advarte studass om at vi burde ha 2 individer på hvert arbeidsområder.

**Kap 5.10 Forsentkomming**

Først flere uker ut i prosjektet identifiserte vi en tendens til at det generelt sett var flere medlemmer som kom for sent til møter. Dette ble tatt opp i et SCRUM-møte, der vi fastslo at tendensen ikke hadde hatt noen skade for prosjektet. Vi diskuterte derimot hvordan det kunne påvirke prosjektet dersom tendensen enten fortsatte eller vokste mer. Konklusjonen ble at vi ga en felles verbal notis til gruppen der vi sa oss enige i å forsøke enda hardere å møte opp til rett tid. Vi følte det ikke var nødvendig med noen strengere måte å behandle risikoen på, ettersom tendensen fortsatt ikke hadde gjort noe skade på prosjektet. Vi ble også felles enige om at ingen med intensjon hadde møtt opp forsinket, men heller vært uheldig med å kikke på klokken. Det ble normert en felles aksept for at uhell kan skje, så streng etterfølging ble heller ikke gjort dersom noen fortsatt møtte opp til feil tidspunkt, så lenge det ble begrunnet og beklaget for.

**Kap 5.11 Avveining mellom forhåndsregler og effektivitet**

Vi har tatt forhånsregler for å redusere risiko, men ikke så mange forhåndsregler som vi kunne ha gjort. F.eks har det blitt til at man i stor grad jobber på hver sin del til tider, siden det er mer effektivt. Dette gjør at vi er mer sårbare hvis noe skulle skje som hindrer et av gruppemedlemmene å bidra videre på en ordentelig måte, men samtidig bør man ikke optimalisere prosjektet med henhold til *kun* risiko-reduksjon for enhver pris.

**Kap 6 Produktbeskrivelsen**

Bruksområder

|  |  |
| --- | --- |
| Steg | Beskrivelse |
| 1 | Bruker logger inn på systemet, ved hjelp av riktig brukernavn og passord. Hvis dette er riktig blir man overført til steg 3. Er det ikke riktig, steg 2. |
| 2 | Kontakt administrator for brukernavn og passord. |
| 3 | Bruker er logget inn på system, her kan han browse kart, sauer og se alarmer. |

|  |  |
| --- | --- |
| Steg | Beskrivelse |
| 1 | Add sheep funksjon, her går bruker inn på eget GUI for å legge til ny sau(er). Legger til navn, dato født, vekt, farm og om den er i live. |
| 2 | Det blir lagt til en ny sau, ny ID genereres automatisk for at 2 sauer har samme ID. |
| 3 | (blir sau marker i sheep list? blir så kanskje markert på kart?) |

|  |  |
| --- | --- |
| steg | beskrivelse |
| 1 | Message funksjon, her får bruker info om sau(farm, dato født, ), siste alarm, location |
| 2 |  |
| 3 |  |

**(er ikke i nærheten ferdig her)**

**Kap 7 Prosessbeskrivelse**

**Kap 7.1 Sprint**

Til å begynne med måtte vi definere sprint-varighet. Her defineres en lang sprint som noe mellom 3 og 4 uker, og en kort sprint som noe mellom 1 og 2 uker.

|  |  |
| --- | --- |
| **Fordeler med lang sprint** | **Fordeler med kort sprint** |
| Mer tid brukes på programmering framfor møter | Kontinuerlig oversikt over prosjektet. |
| Med mer tid kan programmerere legge mer fokus på kvalitetsløsninger framfor å kun fokusere på at kode skal fungere. | Lettere å unngå Parkinsons Law. (*“Work expands so as to fill the time available for its completion.”*) |
|  | Dersom noen blir syke vil det være enklere å plukke opp arbeidsoppgavene til denne personen. |
|  | Lettere å unngå utsetting (procrastination). |
|  | Ettersom man jobber iterativt, blir det enklere å oppdage forbedringer til programvare underveis. |

Nøkkelordene til korte sprinter er oversikt og mer fleksibilitet. Gruppen hadde dog kun mulighet til å samles i fellesskap fast én gang i uken. Hvis dette møtet skulle brukes til sprint-planlegging ville vi ikke ha noe tid samlet til å diskutere kode og få hjelp av hverandre. Dette gjorde at gruppen bestemte seg for sprinter på 2 uker i lengde. Oversikten var viktig å få med tanke på at gruppens mangel på erfaring.

**Kap 7.2 Arbeids fordeling**

*/\*Hvis noen er uenige i det som skrives her er det som vanlig bare å forandre på det eller fjerne det.\*/*  
I kapittel 3.1 skrev vi om annsvarsfordelingen. Annsvarsfordeling er ikke det samme som arbeidsfordeling, men likevel ble det til at mange arbeidet mest med sitt annsvarsområde. Et klart unntak fra dette er Per Øyvind som var Scrum-master men som likevel gjorde en betydelig del av arbeidet utover organiseringen av gruppa.  
  
Når et ansvarsområde har vært for mye for en person er det flere som har jobbet sammen på oppgaven. For eksempel er det flere som har jobbet på diagrammer, flere som har jobbet på kart og flere som har jobbet sammen på GUI. Alle har vært med på å skrive dette dokumentet og på forbredingen av midtveispresentasjonen.

**Kap 7.3 Møter**

Vi ble enige om at hver uke skal inneholde en til to møter. Årsaken til at vi kom frem til dette er at vi ikke har tid til å sette av flere sprintmøter i uken. Dette kom veldig synlig frem når vi brukte nettjenesten doodle ([doodle.com)](http://www.doodle.com/). Der ble det lagt opp en uke der hver dag var delt opp i to-timers-økter fra klokken åtte på morgenen til klokken seks på dagen. På grunn av timeplanene våre var det ikke mange økter som ikke overlappet. Onsdager klokken tolv ble valgt som fast møtetid.  
  
I tillegg til dette har ekstraordinære sprintmøter samt arbeidsøkter blitt kjørt når det har passet for de fleste. Det har vært vanligere at en uke har inneholdt et slikt “ekstraordinært sprintmøte” enn at dette ikke har vært tilfelle. Tirsdag har vært en populær dag for slike møter.  
  
Det har også forekommet at to gruppemedlemmer som har jobbet på en felles del av prosjektet har møttes for å arbeide sammen.

**Kap 7.4 Kildekodedeling**

Som nevnt i kap 5.5 brukte vi Git til både kildekode og andre filer vi genererte (diagrammer og den slags). Dette er et program som bare noen på gruppa kjente til fra før og som tok litt tid å lære. Heldigvis finner man info om Git på nett, og de som kunne Git hjalp til med å lære opp resten. Etter ikke så lang tid var alle komfortable med Git og arbeidet gikk *mye* lettere enn det hadde gjort uten et slikt hjelpemiddel. I det hele tatt kan Git være et veldig praktisk og kraftig verktøy for å koordinere arbeidet sitt på IT-prosjekter.

Kodingen gjorde vi i stor grad i Eclipse. Eclipse er et verktøy som egner seg godt for dette og alle var kjent med Eclipse fra tidligere kurs. Vi hadde en felles prosjektmappe i Eclipse sin “Project explorer”-sidebar som ble oppdatert via Git på samme måte som de andre mappene i vårt felles Git-repository. Så vi delte kildekoden på en automatisert og effektiv måte.

**Kap 7.5 Facebook**

Innad i gruppa har vi valgt Facebook som kommunikasjonsmiddel for alt av generelle spørsmål, møtekalling og andre relevante kommentarer og opplysninger. Vi opprettet en private Facebook-gruppe slik at det kun var vi som kunne se den informasjon vi delte.  
  
Hovedgrunnen til at vi valgte å bruke Facebook er at alle på gruppen oftest har denne nettsiden lett tilgjengelig eller får notifikasjoner på mobilen når det skjer noe nytt der.  
En annen fordel er at man kan se hvem som har sett hva. Da vet man fort at alle har fått med seg den aktuelle informasjonen, eller må bruke andre kommunikasjonsmiddel for å få tak i den personen som ikke har fått det med seg. Dessuten er det veldig lett å nå ut til hele gruppa. Kun et klikk fra Facebook sin forside er nødvendig for å komme inn på gruppa, og deretter kan man skrive nye poster eller kommentarer på poster rett inn. Mye lettere enn å bruke e-mail. Det er også mye lettere å få oversikt over hva som blir sagt osv enn hvis man bruker e-mail.  
  
Et problem kan være med gruppesystemet til facebook kan være at det blir uoversiktelig og rotete med mange brukere og man mister det personlige, men i en gruppe på seks stykker så har ikke dette blitt noe problem.  
  
Ved å bruke facebook som kommunikasjonsmiddel så har vi gjennom arbeidstiden aldri hatt ett problem med å komme i kontakt med og få hjelp fra andre. Facebook fungert utmerket for vårt bruk gjennom hele prosjektoppgaven.

**Kap 8 Systemdesign**

Her må vi henvise til UML digrammer og slikt

**Kap 9 System-funksjoner**  
  
**Liste opp alle klasser og hva de gjør??**  
  
**Sytsem-implementasjon**

**Kap 10 Verktøy**

Følgende verktøy synes vi at vi har forklart tilstrekkelig mye om tidligere i teksten: Qt Jambi, Kartverkets WMS-tjeneste, Trello, IceScrum, MySql, Git, Eclipse, WBS (skal redigere her og skrive på en penere måte)

**JavaDoc**

JavaDoc er blitt brukt for dokumentasjon av koden.

**ER-modeller**

**Dia**

Design av diagrammer er gjort i det GTK+ diagram baserte programmet Dia. Dette var et veldig godt program som passet prosjektet vårt perfekt, da det hadde innebygd spesiale objekt for UML, entitet diagram, skjermflyt diagram og diverse flere diagram. Ingen utenom Lars hadde noen erfaring med dette programmet, men ble fort kjent med det. Grunnen for at vi valgte dette var vell også får at Lars hadde forkunnskaper igjennom et annet fag med dette programmet.

**UML**

Unified Modeling Language er en datarelatert modelering innenfor objekt orientert programmvare. Dette har vi brukt for å designe våre diagrammer, så vi fikk en generell forståelse ovenfor hvordan det ferdige produktet skulle se ut. Klasse diagrammet har vi brukt flittig når vi kodet, mens et screen flow diagram har gitt oss ulike ideer vi ikke hadde tenkt på.

**Google Docs**

I skrivingen av denne rapporten og i forbredelsen av midtveispresentasjonen, samt i planleggingen av enkelte av sprintene, har vi brukt Google Docs. Dette har veldig praktisk siden dokumentene lagres automatisk og man kan skrive og redigere på det samme dokumentet samtidig og se hva de andre gjør i real-time.  
  
Forklaring av verktøyene vi har valgt å bruke?  
Men går ikke dette litt inn på arkitektur  
**Brukermanual (teknisk og ikke-teknisk)**  
  
**Vedlegg-oversikt**

**Kap 11 Evaluering**

**Kap 11.1 Hva gikk bra**

* sammarbeid i gruppa
* alle var arbeidsvillige
* strukturerte i startfasen og fikk kartlagt mye så vi kom fort igang.
* gode møter (scrum fungerte bra)
* Romres, godt å ha.
* fikk bruke serveren til Lars, perfekt. satt opp icescrum

**Kap 11.2 Hva gikk dårlig**

* alle endte ikke opp med å gjøre like mye
* litt dårlig start med dokumentasjon av arbeid (trello)
* burndown-chartene kunne vært penere
* ble ikke ferdig så tidlig som vi ønsket/burde
* møtte på problemer (men det var vel mest uflaks?)
* qt var vanskalig (mer research?)

**Kap 11.3 Lærdom til neste gang**

* lære rammeverket bedre før vi starter prosjektet, da er vi enda mer gardert for å ikke miste arbeid som vi desverre gjorde under en git fuck up.
* lært mer om, og fått erfaring med, estimater
* viktigheten av å møte ofte

1. WBS er en mate å dele opp prosjektet I faser på, gjerne i ”barn” og ”foreldre”. [↑](#footnote-ref-1)