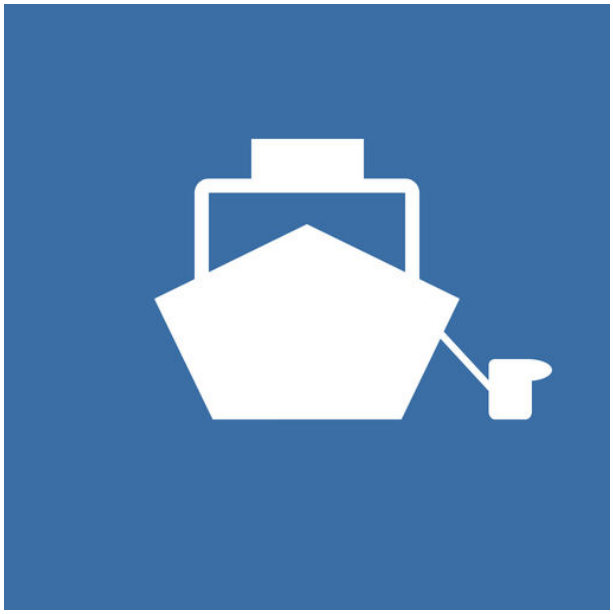
Chalmers tekniska högskola

Slutreflektion Grupp ABC

Johan Erlandsson, Linus Nilsson, Fredrik Mile
Karin Malmgren, Gustav Nilsson, Rasmus Lindy
Sebastian Ringqvist, Tove Mannheimer,Oscar Edvinsson

25 maj 2018



Innehåll

1 Introduktion

Detta projekt är en samling utav den lärdom som de kontinuerliga veckoreflektionerna har genererat. En tydlig skillnad i reflektioner från vecka till vecka har märkts, något som även skildras i denna slutreflektion. Utöver att detta är en samling utav all den kunskap vi tar med oss till framtida projekt, är det även den slutgiltiga veckoreflektionen för vecka 8.

Under projektet och kursens gång har svaren varierat kraftigt från vecka till vecka. En tydlig skillnad märktes i varje sektion vartefter arbetssättet med reflektionerna ändrades. Från början delades reflektionen upp så att alla medlemmar skriva ett stycke själv. Detta gav dock ingen möjlighet att diskutera frågorna med varandra, så vi bytte till att hela gruppen gick igenom och skrev varje stycke tillsammans. Detta tog tyvärr väldigt lång tid och kändes ineffektivt. Istället testade vi att gruppen reflekterade över sprinten översiktligt gemensamt, för att sedan dela upp oss i grupper om tre där själva skrivandet skedde. Efteråt reflekterade gruppen tillsammans över resultatet. Detta arbetssätt kändes väldigt bra, och var det vi använde för resten av reflektionerna.

Gruppen har i stora delar av projektet arbetat i par eller grupper med de tasks som funnits i sprint backlogen. Då vissa av gruppmedlemmarna länge hade problem med utvecklingsmiljön kändes parprogrammering som ett lämpligt verktyg för att kunna påbörja utvecklingen av koden. Då olika personer i gruppen hade olika erfarenhet av versionshantering och att arbeta i Git fördelades arbetet med att pusha främst till dem. De personer som kände sig osäkra kunde då lära sig hur man enklast arbetade med Git och när de kände sig mer bekväma kunde även de börja pusha. Därför kan statistiken gällande commits i repot antas vara fördelat till större del på vissa individer.

2 Omfånget av applikationens utveckling inkluderat prioritering av funktionaliteter och för vem värde skapas

Den första veckan kände gruppen att det var oklart vad som avsågs med applikationens omfång. Redan i den andra veckan klarnade det upp en aning, då gruppen blev tilldelad projektet Vessel 2. Tilldelningen gav en förståelse för vilket fokus applikationen skulle ha. Det gav även en trygg känsla att vi insåg att vi skulle ha kontinuerliga möten med vår produktägare och slutanvändare.

Vidare uppstod utmaningar med den givna utvecklingsmiljön vilket tog tid att reda ut. Till slut fick samtliga gruppmedlemmar igång utvecklingsmiljön och kunde påbörja programmeringen. Det ledde till att vi kunde göra mindre modifikationer i applikationen till demonstrationen för produktägarna vilket gav värde för vårt projekt och produktägarna. Det var även i detta skede som vi insåg att den primära uppgiften var att identifiera de existerande funktionerna som gav värde till vår applikation-modifikation och behålla dessa och vidare ta bort de funktioner som inte ansågs nödvändiga.

Efter samtal med vår slutanvändare Robert insåg vi att hans främsta önskan var att skapa en vy där kaptenen enkelt kan rapportera så kallade ”ETA’s”, Estimated Time of Arrival på en 24 timmars basis innan skeppet kommer till hamnen. Gruppen prioriterade därför utvecklingen av ETA-vyn.

I de sprints som genomfördes efter skalades applikationen ner ordentligt för att bara visa information som är relevant för slutanvändaren, samtidigt som fler idéer framkom, såsom att implementera notiser för att påminna användaren om när en ETA skulle genomföras.

Reflektionen kring applikationens omfång och funktioner gav oss en klarare bild utav vad vi hade fått gjort i den gångna veckan samt vad utav det vi hade gjort som faktiskt skapat värde för pro-

duktägarna och slutanvändare. Genom att genomföra retrospective med slutanvändare när det var möjligt kunde vi få feedback på det vi hade gjort under sprinten. Vidare fick vi även en klarhet i hur vi hade prioriterat, om något område hade fått för mycket eller för lite tid. Genom att reflektera gemensamt kunde gruppen också konstatera att utvecklingen av notiser skulle vara alltför resurskrävande, och gruppen valde därför att inte implementera dem. Till framtida projekt tar gruppen med sig värdet av att utvärdera vad en funktion tillför för värde och för vem den är värdefull. Genom att återkomma till reflektionen, och ha kontakt med slutanvändare, kunde gruppen undvika att lägga till funktioner som inte efterfrågades, eller som inte skapade värde för någon part.

Det som uppdagades under projektets gång var att produktägare och slutanvändare inte själva vet vad de vill ha. Det hände även att produktägarna och slutanvändarna sade motstridiga saker vilket gjorde det svårt att veta vem man skulle lyssna på. Mycket tid gick därför åt till att navigera mellan önskemål och behov och balansera det med den kunskapen gruppen hade.

3 Socialt kontrakt

Under första mötet bestämdes gruppens interna förhållningsregler som satte ramarna för projektets arbetsgång. Vi kom fram till att möten skulle bokas gemensamt och att det var viktigt att komma i tid till dessa. Fasta möten sker någon gång i veckan och uppdateringar ska ges minst varannan dag (mån, ons, fre).Om man inte kunde närvara krävdes en giltig anledning och att det meddelades i god tid i förväg.

Varje dag vi arbetade ihop skulle daily Scrum genomföras under cirka 15 minuter. Om samtliga inte kunde närvara görs daily Scrum via Slack-kanalen.

Alla i gruppen var villiga att lägga ned den tid på kursen som krävs och siktar på att prestera efter bästa förmåga. Det viktigaste var ett bra och fungerande resultat. Prestationen utvärderades efter halvtid för att vi skulle kunna bedöma hur vi låg till.

Skulle problem uppstå med utebliven närvaro skulle det först kommuniceras inom gruppen för att förstå orsaken. Går det av någon anledning inte att lösa internt kontaktas handledare. Problem med närvaro definierades som att man vid upprepade tillfällen var frånvarande. Engångsföreteelser kunde lösas med fika. Genom projektet skulle vi försöka att bibehålla en jämn arbetsbelastning mellan gruppmedlemmarna. Om problem uppstod så hanteras det på samma sätt som med närvaron.

Under projektet följdes innehållet i det sociala kontraktet delvis. Vissa medlemmar kom sent till möten utan att på förhand meddela gruppen. Då vi var ett stort antal gruppmedlemmar var det sällan något som ställde till problem för arbetet, däremot kunde det störa planeringen något.

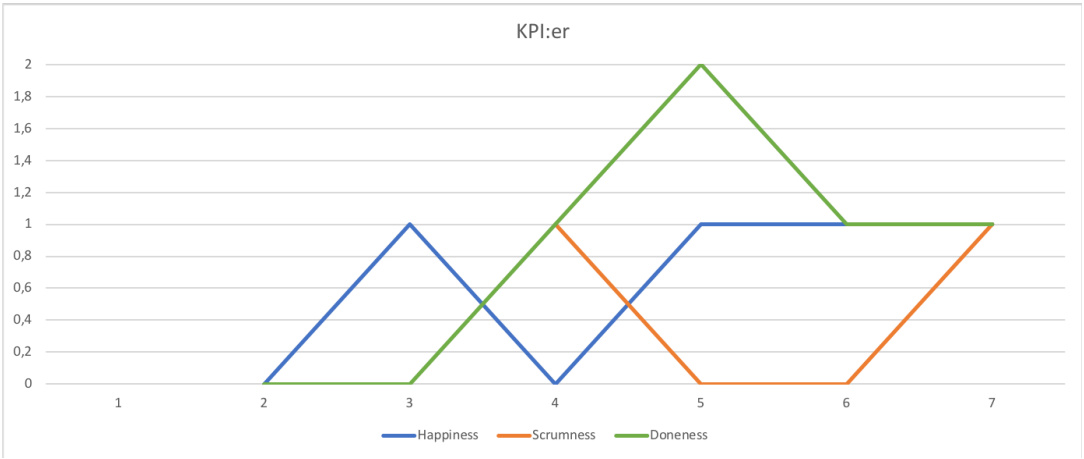
Till framtida projekt tar gruppen med sig en lärdom om att det är viktigt att förtydliga konsekvensen av att komma för sent, exempelvis genom någon mildare bestraffning för att motivera gruppmedlemmarna att komma i tid. I övrigt anser gruppen att ett socialt kontrakt är ett lämpligt verktyg för att säkerställa och tydliggöra ambitionsnivån i gruppen.

4 Framgångskriterier för gruppen i termer av vad vi vill uppnå med applikationen

Tre stycken KPI:er utformades i början av projektet som utvärderades och mättes under varje sprint. KPI:erna skapades för att analysera arbetet avseende: Produktivitet, Välmående och Metod. Vidare bedömdes KPI:erna enligt en skala 0-2, där 0 betyder att en KPI följts prickfritt och 2 medger att något bör förändras inom vederbörande kriteries område.

Första veckan bestämde sig gruppen för att mäta hur bra vi var på att genomföra uppgifterna i sprint backloggen, hur bra vi trivdes med arbetstempot och hur strukturerad den producerade koden var enligt en bestämd koddoktrin. Inledningsvis kunde den senast nämnda KPI:n inte mätas, eftersom inte tillräckligt mycket kod lades till. Inte heller under efterföljande veckor kunde denna KPI mätas då koden som skrevs var till majoriteten återanvänd från existerande program, varför den togs bort. I stället lades en KPI till för att mäta huruvida överenskommen scrummetodik följdes, således även kallad ”Scrumness” som också mäts enligt en skala 0-2.

Figur 1 visar hur resultatet har varierat från vecka till vecka för de olika KPI:erna. Under den första sprinten sattes enbart KPI:erna upp för att möjliggöra utvärdering i nästkommande sprint. Under den andra sprinten blev framgångskriteriet mött eftersom alla uppgifter genomförts och stämningen i gruppen var mycket god.



Figur 1: Resultatet av gruppens KPI:er under projektets gång. Varje färg motsvarar ett KPI och värden varierar mellan 0-2 beroende på hur gruppen utvärderat sprinten.

Sammantaget varierade KPI:erna mycket under projektets olika faser. Vi upplevde att det gick bra att mäta KPI:erna gemensamt för gruppen då det som påverkade utfallet ofta påverkade hela gruppen och inte enskilda individer.

Lärdomarna vi dragit och tar med oss till nästa projekt är att sätta upp tydliga och mätbara framgångskriterier. En idé som väcktes i slutfasen var att individuellt fylla i värden på exempelvis ett Google-formulär för att undvika att gruppen påverkar varandra. Vi insåg även vikten av att framgångskriterierna stämmer överens med det arbetet som görs.

För att på ett effektivt sätt arbeta med framgångskriterier tror vi att det viktigaste är att sätta upp kriterier som är tydligt mätbara och mäter olika aspekter av projektet. Det behöver inte vara rätt kriterier från början, men genom att börja någonstans finns det saker att bygga vidare på kommande sprintar och tillslut förstår man vilka kriterier som är viktiga för arbetet. Att kunna titta tillbaka och få en överblick över hur kriterierna har varierat över tid har också gett gruppen viktiga insikter. Mycket tid gick åt till icke-värdeskapande aktiviteter i början av projektet och en jämnare arbetsbelastning och bättre välmående hade kunnat skapas om arbetet spritts ut mer. För att åstadkomma detta tror vi att det är nyttigt att inledningsvis sätta upp en tidsplan och gå igenom de dagar som faller bort på grund av helger osv. för att få förståelse över vilka mål som är rimliga i förhållande till given tidsram.

5 Acceptanstester

Under projektet har acceptanstester satts upp och genomförts för alla sprintar efter att arbetet med applikationen dragit igång. De första veckorna gjordes inga acceptanstest för att vi inte kände att vi hade tillräckligt med underlag. Såhär i efterhand hade det varit bra att ändå utforma acceptanstest. Trots att vi inte kunde göra det specifikt för applikationen, så arbetade vi med uppgifter runt omkring som hade kunnat testas. Det hade gjort att vår utveckling och vårt arbete dessa veckor hade synligjorts bättre.

Övriga sprintar gjordes acceptanstest framförallt i kommunikation med produktägarna och slutanvändare. Vid varje sprintutvärdering kollade vi tillbaka på det testet vi utformat veckan innan och formulerade ett nytt för den kommande sprinten.

Acceptanstesten gjordes anpassade till produktägarnas och slutanvändarens önskemål eftersom vi ville kunna generera nytt värde till dem varje sprint. De byggde framförallt på att kunna visa upp ny funktionalitet i applikationen. Vid retrospective med slutanvändaren lät vi honom använda appen och bedöma om han tyckte att funktionen var bra. Om Robert tyckte funktionen var väl utformad, samt gruppen ansåg sig nöjda med utformningen hade funktionen passerat acceptanstestet. Acceptanstesten var viktiga för kommunikationen med produktägarna och slutanvändaren eftersom vi fick feedback på vad som var bra men också på det som kunde bli bättre.

Vi utformade ingen ordentlig plan för vad vi skulle göra om acceptanstesten inte klarades. I kommande projekt tror vi att det kan vara bra eftersom det annars lätt blir att samma test ligger kvar sprint efter sprint eller att massor med tid går åt för att klara ett acceptanstest som kanske egentligen är fel utformat. I framtiden tror vi att det är viktigt att tidigt uttala i gruppen att acceptanstester måste tillåtas att omformuleras samt att precis som med framgångskriterierna börja någonstans. Att inte sätta så stor press på vad ett acceptanstest behöver vara eller hur stor nytta det måste generera och för vem.

Några sprintar hade vi inte personlig kontakt med slutanvändaren och kunde därmed inte visa upp de funktioner vi definierat som acceptanstest. Vi skickade då istället skärmdumpar på det vi ville visa för att få feedback. Kommunikationen med produktägarna och slutanvändaren var väldigt viktig för att kunna sätta upp bra acceptanstest vilket gör att vi i efterhand gärna hade sätt att denna kommunikation skötts tätare. Det är något som vi kommer tänka på i framtida projekt genom att vi har lärt oss vikten av input från olika aktörer.

För att öka kommunikationen tror vi det krävs en öppenhet för att mötas på olika kanaler. Det kommer inte alltid vara möjligt att träffas i person utan det är viktigt att nyttja verktyg som e-mail,

Slack och telefon.

6 Applikationens design

Applikationen har utökats och utvecklats på liknande sätt som den existerande applikationen var utformad. API:er finns dokumenterat på utvecklarnas hemsida, samt de grundläggande API:er för bland annat Android, Java och de andra komponenter som redan finns i applikationen. Vi har försökt i största möjliga mån använda den kodstil som redan finns i applikationen och återanvända de delar vi kan. Detta är önskvärt just för att genom att återanvända befintlig kodstil behåller vi applikationens utseende och enhetlighet vilket gör det lättare att följa. Det har uppnåtts genom att applikation i grundutförande har brutits ner och analyserats för att förstå hur applikationen är uppbyggd.

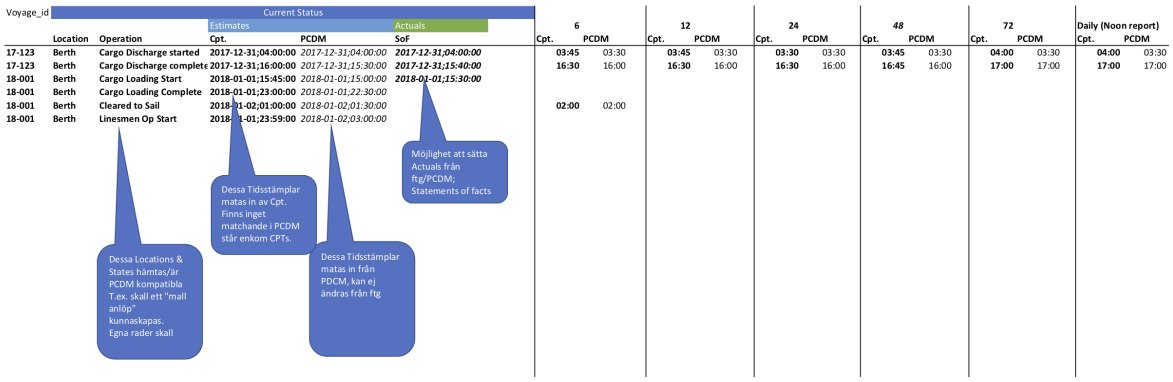
Från detta tar vi med oss att vid ett projekts start se till att alla är väl införstådda i vad som gäller angående designen. I början av detta projekt förstod vi inte riktigt hur mycket utav den befintliga koden/designen vi kunde använda och var i vissa fall införstådda i att vi skulle ta fram en ny egen design. För att undvika detta i framtida projekt så ser vi till att ha god kommunikation med den som har hand om den befintliga applikationen och ser vad vi kan ta med oss därifrån.

Vidare tar vi med oss hur man lär sig att använda kod/design som redan är applicerad i applikationen. Genom vårt projekts gång har det stundvis vart tillfällen då vi har påbörjat en funktion/design som vi inte har vart medvetna om att de redan existerar på andra instanser i applikationen. Detta har lett till onödigt arbete och förvirring. För att undvika detta i framtiden ska vi göra en noggrannare analys utav den befintliga applikationen och utvärderat de existerande funktionerna mer noggrant.

7 Beteendeöversikt av applikationen (genom användarscenarion, interaktionsdiagram eller liknande)

Under inledningen av projektet togs ett antal user stories fram, varav delar sållades bort vid utformningen av en produkt backlog, då dessa ansågs vara bortom tids- och/eller kunskapsramar.

Slutanvändare gav gruppen tillgång till hur en ETA-rapportering kunde se ut redan vid första mötet, vilket hjälpte gruppen i utformningen av de nya funktionerna. Bilden finns presenterad i figur 2.



Figur 2: Slutanvändarens bild av hur ETA-rapportering kan se ut

Gruppen arbetade sedan utefter det scenario som fanns att tillgå på kurshemsidan och skapade användarscenarion. Vid färdigställande av applikationen kunde en lista över implementerade användarscenarion tas fram, vilka lyder:

- **Favorite overview + Select Vessel:**
  Kaptenen önskar att få en översikt över vilka portCalls som gjorts av sitt fartyg. Hen vill även att fartyget är sparat som sitt fartyg eftersom det inte kommer att förändras och därmed är inte andra fartyg relevanta att titta på.
- **Estimated Time of Arrival:**
  Kaptenen behöver rapportera ETA till VTS area regelbundet under ett portCall och vill veta när dessa rapporter ska skickas in. Det är nu dags att skicka in en daglig rapportering, så kallad noon-report, och kaptenen är osäker på när den senast skickades och således när det är dags igen.
- **Favorite-States:**
  - Kaptenen ska påbörja ankring och ska rapportera det i applikationen för att göra andra uppmärksamma på detta. Hen vill hitta rapporteringsfunktionen så smidigt som möjligt.
  - Kaptenen vill beställa vatten till fartyget i applikationen. Hen är osäker på vad funktion heter i applikationen och vill hitta den så fort som möjligt eftersom det är en funktion hen använder ofta.
- **Actuals-timeline:**
  Kaptenen ska skriva in tider i sin loggbok men är osäker på när olika händelser faktiskt ägde rum och vill ha en översikt på de tidsstämplarna som ska skrivas in.

Genom att se till att dessa användarscenarion fungerar väl i appen skapas värde till både slutanvändare och produktägare, eftersom de alla har ett intresse i en app som är så anpassad och användarvänlig för kaptenen som möjligt.

Ett framtida användarscenario skulle kunna vara att implementera ett, som projektgruppen kallar det, Blocket för PortCDM". Det är en plattform där fartygspersonal och aktörer som önskar göra affärer med fartyget kan kontakta varandra. Det skulle vara en betydelsefull funktion för en sjökapten, enligt kontakt med vår slutanvändare. Genom fortsatt tillämpning av agila metoder inom gruppen

kan fler funktioner implementeras på ett effektivt sätt. Gruppen tar med sig till framtida projekt att det är värdefullt att arbeta utefter användarscenarion för att säkerställa att rätt funktioner skapas. Genom att ta del av riktiga dokument från slutanvändare kunde en bättre förståelse skapas, varför gruppen anser att kontakten med användaren är viktig.

8 Strukturell överblick av applikationen (genom klassdiagram, domänmodeller, komponentdiagram eller liknande)

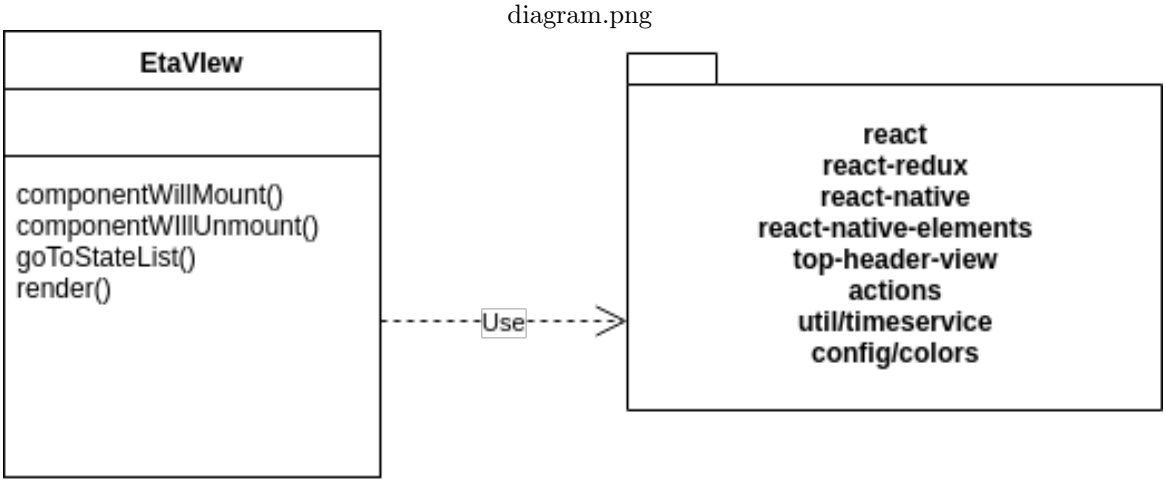
De funktioner som lades till utgick från strukturen i den befintliga applikationen.

Under projektets gång gjordes många skisser på förändringarna vi ville göra i applikationen för att få en tydlig gemensam målbild att utveckla utefter men också för att kunna kommunicera våra tankar till produktägare och slutanvändare.

Skisserna var främst till för att få en övergripande bild. UML-diagram gjordes sedan för att underlätta utvecklingen av det vi ville åstadkomma och för att se vad som kunde återanvändas. Nedan beskrivs utformningen av två nya vyer i UML-diagram.

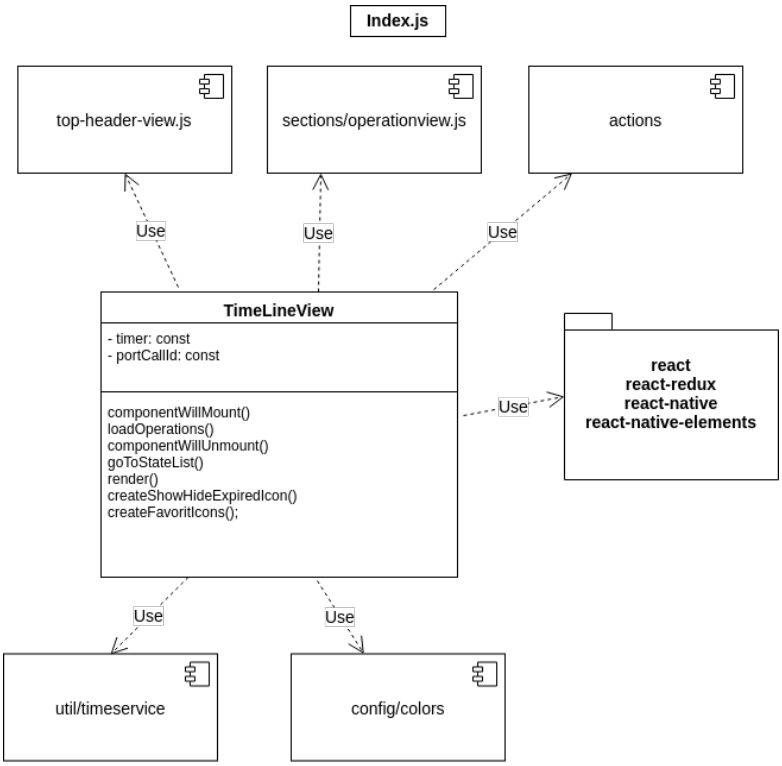
Figur 3 visar strukturen för ETA-vyn med de funktioner och paket som används.

diagram.png



Figur 3: UML-diagram över ETA-vyn vecka 6

I Figur 4 nedan presenteras ett UML-diagram över index.js filen för vyn som visar en tidslinje med tidsstämplar för de faktiska tiderna.



Figur 4: UML-diagram över vy för tidslinjen vecka 7

Då en befintlig struktur fanns att utgå ifrån ansåg inte gruppen att det var värt att förändra den i någon större utsträckning. Möjligen hade strukturen sett annorlunda ut om applikationen utvecklats specifikt för Vessel 2 från början, men den befintliga strukturen ansågs fungera väl även för de förändringar vi ämnade att göra. Att förändra strukturen hade varit mycket tidskrävande och ansågs inte addera värde till användare eller produktägare.

9 User stories

Under inledningen av projektet formulerades user stories, vilka både skulle vara tillräckligt tydliga för vem som helst i gruppen att förstå, men också slutanvändaren. Att slutanvändaren förstod var viktigt för att kunna ge bra feedback att jobba vidare med. Dessutom skulle dessa user stories vara enkla att bryta ned, för att enkelt kunna definiera tasks att arbeta vidare med.

Att skriva user stories ger en bra uppfattning om projektets omfattning. Vi utgick ifrån förra årets demostrationsscenario för att ta fram user stories samt kommunikation med slutanvändare/produktägare.

Det var betydligt enklare att bryta ner epics till user stories jämfört med att bryta ner user till tasks, troligtvis för att det sistnämnda är mycket mer på detaljnivå. I vissa fall upplevde gruppen att de misslyckats i sin nedbrytning av user stories till tasks. Vissa av kraven på user stories:en blev

inte tillräckligt definierade och task:en krävde därför mer tid än vad som var avsatt. Vissa tasks blev också beroende av varandra, vilket gjorde att det var svårt att parallellisera arbetet. En annan svårighet var att estimera tidsåtgången för task:en. Problematiken var tätt sammankopplad med att task:en var otydligt formulerade, och därmed svåra att estimera.

Ett annat problem vi upplevde med våra tasks var *DoD, Definition of Done*. En del av tasks:en saknade någon DoD och ansågs klara efter att en diskussion hade förts angående huruvida vi upplevde tasken som klar eller ej. Det blir betydligt enklare när DoD är definerat och tydligt, vilket vi tar mes oss in i framtida projekt. Att tydligt bestämma när något är klart bidrar positivt till arbetssättet, om DoD inte specificeras är det lätt att det blir stora frågetecken och komplikationer som hade kunnats undvikas.

Gruppen tar därför med sig till kommande projekt att det är värt att lägga extra tid på att tydligt definiera och förtydliga innebörden av olika tasks för att effektivisera arbetet. Då många olika personer är inblandade i arbetet är det problematiskt att anta att alla har samma förståelse för vad uppgiften innebär. Dessutom kan det vara bra att försöka estimera uppgifterna noggrant för att undvika att gruppen åtar sig allt för många tasks under en sprint.

10 KPI:er

Gruppen bedömde sin arbetsinsats genom tre olika KPI:er varje sprint. Två av KPI:erna var samma under hela arbetets gång, medan en fick bytas ut efter två sprintar. KPIn ersattes då med en ny som ansågs mer relevant för projektets utformning. KPIn som togs bort var ”Kodkvalitet utifrån vår koddoktrin”. Anledningen var att projektet utgick från kod som var skriven enligt en viss stil. En stor del av det arbete som gjorts innebar också att den befintliga koden endast skalades ner, varför vår kodstil inte uttrycktes i någon större utsträckning. Gruppen enades därför om att byta ut KPIn ”Kodkvalitet utifrån vår koddoktrin” till ”Scrumness”, vilken avsåg att mäta hur väl gruppen arbetade utefter Scrum-metodiken. Parametrar som togs i beaktning var: huruvida daily Scrums hade hållits, Scrumboarden använts på rätt sätt, samt om review och retrospective genomförts under sprinten. KPIn blev en motiverande faktor för gruppen att hålla sig till arbetssättet och följa Scrum-metoden effektivt.

De KPI:er som bestod under hela projektets gång var ”Doneness” och ”Happiness”. ”Doneness” avsåg att mäta huruvida gruppen slutfört de uppgifter som låg i sprintbacklogen under sprinten. Genom att lägga de slutförda uppgifterna i ”Done”-kategorin på gruppens Scrumboard kunde resultatet av KPIn enkelt avläsas. De kort som fortfarande var i ”sprint backlog”, ”in progress” eller ”for review” när sprinten tog slut var inte klara, och KPIn bedömdes därefter.

KPI:erna utformades med tanken att de skulle täcka in olika aspekter av arbetets gång. Hur bra vi mådde som grupp, hur mycket fick vi gjort och hur väl följde vi den uttalade metodiken?.

Resultaten av KPI:er varierade något under projektets gång. I takt med att gruppen lärde känna varandra och projektet bättre kunde arbetet fördelas och genomföras på ett bättre sätt. Gemensamt kan gruppen i efterhand konstatera att ett dåligt värde på en av KPI:erna ofta även ledde till sämre på de andra. Exempelvis om gruppen inte hann slutföra samtliga uppgifter under en sprint och ”doneness” fick ett dåligt värde, fick ofta även ”happiness” ett dåligt värde.

Gruppens önskan var självklart att under samtliga sprintar sätta värdet 0 på de tre KPI:erna. Att göra på det viset hade dock varken varit sanningsenligt eller lärorikt. Genom att utvärdera

situationen gemensamt efter varje sprint kunde gruppen sätta ett värde som speglade situationen väl och utifrån det lära sig hur man skulle justera arbetssättet till kommande sprintar.

Det gruppen tar med sig till andra projekt är hur mätbara variabler kan vara ett effektivt verktyg för att utvärdera det pågående arbetet och förstå hur situationen måste justeras för att uppnå gruppens mål. Genom att reflektera och fundera på varför mätvärdet blev som det blev hjälpte gruppen i att arbeta bra tillsammans. Gruppen är överens om att det är viktigt att KPI:erna är relevanta för projektet som genomförs, samt att det är bra att byta KPI om gruppen gemensamt anser att det inte är en lämplig parameter för att mäta arbetets framsteg.

För att säkerställa användning av KPI:er i framtida projekt kommer gruppen vara öppna för att testa nya KPI:er eftersom vi tar med oss att det är möjligt att byta efterhand om de inte skulle passa eller täcka in de relevanta mätbara faktorerna som för projektet vidare.

11 Kodkvalitet

Gruppen valde att använda två stycken olika verktyg för att kontrollera kodens kvalitet. Dock användes bara verktygen på den kod som skrevs eller blivit modifierad av gruppen. Verktygen applicerades nära slutet av projektet, på grund av att gruppen kände att det var viktigare att fokusera på utveckling och sedan kvaliteten.

Verktygen som applicerades var Codacy och Deepscan som finns närmare beskrivna i Appendix A. Båda verktygen har stöd för ett antal språk och appliceras på projektet på samma sätt. Användaren länkar sitt Github-konto till respektive plattform. Efter länkningen får användaren välja vilket repo som ska analyseras. Varför just dessa verktyg användes berodde framförallt att på att projektet utgick ifrån en nästintill komplett kodbas, vilket gjorde det svårare att skriva egna tester själva, om inte än onödigt med tanke på att bara små delar av koden är skrivet av projektgruppen.

Problemet med Codacy för gruppen var att det inte riktigt hade stöd för react-native. Konsekvensera av det var att majoriteten av dom fel som verktyget hittade inte var exakta. För att hitta dom korrekta problemen fick gruppen gå igenom varje enskilt problem för att se vad som kunde lösas, utifrån react-native.

Då debug verktygen började användas av gruppen skapades en egen debug branch i verionshanteringsverktyget Git. När en ny feature var klar mergades den in i utvecklingsbranchen och därefter in i debug. På debug branchen bearbetades koden utifrån analyserna gjorda av Codacy och Deepscan. Anledningen till att en egen debug gren skapades var för att undvika konflikter i Git. Sammanlagt har verktygen resulterat i bättre kodkvalitet. Det hade underlättat om verktygen infördes tidigare i projektet för att minska antal problem vid varje commit.

Vi har under detta projekt inte skrivit speciellt många rader kod och därav har kodkvaliten aldrig blivit något bekymmer. Till framtida projekt tar vid med oss att det kan vara bra att börja tänka på kodkvalitet redan från start och arbeta med det kontinuerligt. Vid projekt där det skrivs mer kod hade det nog varit att föredra att utveckla egna tester för att säkerhetställa att kodkvaliten är hög. Genom att introducera verktyg lämpade för projektets typ från start och sedan se till att utvärdering av kodkvalité ingår i utvärderingsfasen av sprintarna kommer våra lärdomar hjälpa oss i framtida projekt.

12 Rollfördelning i gruppen

Under projektets gång har gruppen använt sig av olika roller. Vi har haft en Scrummaster, en rumsbokare, en kontaktperson med slutanvändare och kontaktperson med produktägare. Scrummasterns uppgift har varit att gå på de veckoliga Scrum of Scrums samt medlat med de övriga projektgrupperna. Rollen har varierats så att flera fått testa på att vara Scrummaster. Rumsbokarens uppgift har varit att boka grupprum till daily-Scrum-meeting och grupparbetstid. Kontakten med slutanvändare har i stor utsträckning skett via email och därför har den rollen av praktiska skäl inte varierats mellan gruppmedlemmarna. Vid möten med produktägare har alla deltagande gruppmedlemmar agerat kontaktpersoner.

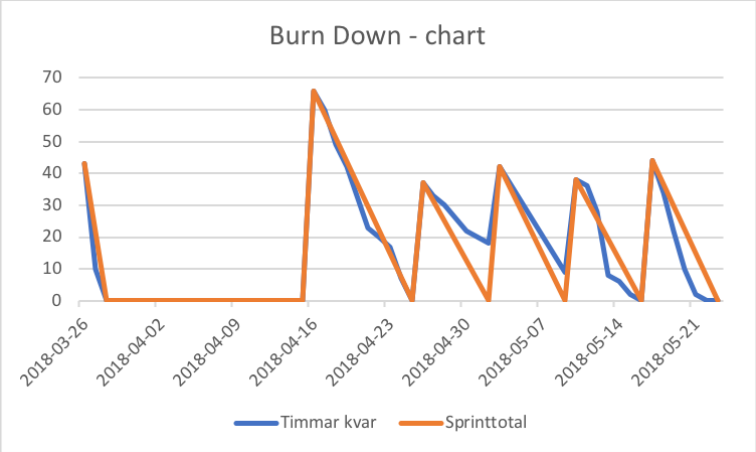
Genom att ha olika roller fördelades arbetsbördan och det fanns alltid någon ansvarig för respektive moment. För att gruppen alltid ska ha någon medlem ansvarig på respektive roll har det tagits upp vid varje veckoreflektion för att se till att det inte står tomt.

Vi gick från att i början byta roller vid varje sprint till att mot slutet ha samma roller under flera sprintar. Vi lärde oss att detta var smidigare och gav en bättre struktur på arbetet. Det tillkom även roller under arbetets gång när vi insåg att behovet fanns, ett exempel är rollen som rumsbokare. De erfarenheter vi fått kan appliceras på andra projekt där agila metoder används och vi har då från start möjlighet att ta oss an de roller som behövs. Alla projektet är däremot unika vilket gör att andra roller kan behövas.

13 Agila arbetsprocesser

Gruppen har anammat de agila arbetsprocesserna som blivit introducerade i kursen. Scrummetodiken har använts genomgående i alla sprintar. Vi har haft daily Scrums där vi kunnat hänvisa till vår Scrumboard när man har förklarat vad man har gjort och vad man ska göra. User stories har delats upp och estimerats till tasks. Arbetet har fördelats utefter sprintbacklogen i Scrumboarden. Sprintarna har avslutats med review av sprintens uppgifter och genom kontakt med slutanvändare har någon form av retrospective genomförts.

Gruppen använde sig även av parprogrammering när vi arbetade med tasksen och för att summera upp projektet har vi använt oss av en burndown chart som visas i figur 5.



Figur 5: Burndown chart över projektets tasks

Önskvärt skulle varit att jobba mer med metodiken så att Scrumboarden kan användas på ett effektivare sätt. Till en början var det svårt att organisera upp Scrumboard och kunna estimera en task gentemot andra tasks, funktioner och gruppens velocity. I och med detta var det en utmaning att få in rätt mängd tasks baserat på en given tid. Scrumboardens sprint backlog blev nu istället styrande faktor på vad som skulle åstadkommas under en sprint snarare än gruppens velocity under given sprint. Lego-övningen kring hur velocity sätts och hur korrekt estimering av uppgifter går till var givande för projektet men vi upplevde ändå att våra estimat ofta var felaktiga. Gruppen tar med sig att en utmaning i projekt är svårigheten att estimera omfattningen på tasks och funktioner. Hur problemet underlättas görs genom många utvärderingar och konstant omvärdering av tasks. Det är svårt att lära sig de agila arbetsätten enbart genom teori vilket gör detta projekt väldigt betydelsefullt. Vi har fått egen erfarenhet av vad som funkat bra och mindre bra för oss. Det gör att tröskeln att tillämpa ett agilt arbetssätt i nya projekt är betydligt lägre.

14 Tidsåtgång

Gruppen har valt att ha möten vid tre tillfällen under varje sprint. Vid mötena har en kort stund spenderats på daily Scrum där varje gruppmedlem kort har berättat för gruppen vad den har gjort, vad den planerar att göra och om den behöver hjälp. Vid mötet som varit sista dagen i sprinten har gruppen gemensamt genomfört en sprint review och skrivit den gruppreflektion som är grunden till denna sammantagna beskrivning av projektet. Vid de tillfällen som produktägare och slutanvändare varit tillgängliga har ca 30 min diskussion skett med dessa. Gruppens möten har varit ca 3-5 timmar långa, vilket resulterat i ca 14 timmar per person och sprint. Utöver det har vissa gruppmedlemmar mötts utöver de gemensamma mötena för att arbeta med tasks. Totalt sett har gruppens medlemmar spenderat ca 20 timmar per person per sprint på projektet. Vissa sprintar blev kortare, på grund av helgdagar, vilket i vissa fall ledde till att gruppen hade färre möten alternativt arbetade enskilt i större utsträckning.

Önskvärt hade varit att sprintarna hade haft samma längd. När helgdagarna inföll är dock inte något som gruppen kunde påverka, utan istället anpassades arbetsbördan i största möjliga mån

efter tillgänglig tid under sprinten. Vid Valborg hade vi exempelvis inte sprintens kortare längd i åtanke vid planeringen av sprintbackloggen. Det resulterade i många ofärdiga tasks och stress i gruppen. Vi tog sedan med oss detta när Kristi himmelsfärd kom och anpassade då sprintbackloggen till tiden vi hade.

En annan ideal lösning hade varit att ha ett kort möte varje dag under projektet för att kunna hålla samtliga medlemmar i gruppen uppdaterade på det pågående arbetet. Dock uppgick projektets omfattning endast till ca 20 timmar per vecka per person, och gruppen var eniga om att möten varje dag skulle resultera i allt för mycket ställtid. Gruppen hittade därmed ett för oss välfungerande arbetssätt med möten varannan dag. Stora förändringar som gjordes däremellan kunde individerna meddela varandra om i Slack-chatten för projektet. Överlag upplevde gruppmedlemmarna att gruppens arbetssätt möjliggjorde ett effektivt arbete, och arbetsbelastningen blev bra.

En lärdom som gruppen tar med sig till framtida projekt är att det är bra att försöka logga arbetstiden, dels individuellt och gemensamt för att säkerställa att tillräckligt med tid läggs på arbetet. Det är också ett enkelt sätt att se till att gruppen inte spenderar för mycket tid på projektet, utan arbetar i enlighet med kursens omfattning.

15 Sprint review

Gruppen har konsekvent genomfört en sprint review vid slutet av varje sprint under projektet. Då möjlighet fanns hölls retrospective med produktägare och slutanvändare.

Det gjorde att vi fick kontinuerlig feedback på sprintens arbete och kunde sedan planera inför nästa sprint. Exempelvis tog vi inledningsvis bort funktionen att skapa nya portcalls eftersom vi fått uppfattningen att det inte gjordes av kaptenen. När vi presenterade detta för slutanvändaren framgick att det ibland hände att nya portcalls behövde skapas varpå vi fick ta tillbaka funktionen igen. Det gjorde att vi lärde oss vikten av att utvärdera tillsammans med produktägare och slutanvändare. Vi fick flera gånger även positiva reaktioner vid utvärderingarna vilket gjorde att vi förstod att vi var på rätt väg.

Det optimala hade varit mer frekventa avstämningar men då detta inte var möjligt löstes problematiken med att eposta slutanvändaren och skicka över skärmavbilder på det vi gjort. Slutanvändaren var inte alltid snabb på att svara, och skillnaden i responsen mellan mail och personlig kontakt gjorde att gruppen föredrog att träffa användaren. Det hade möjligen varit bättre att i framtida projekt sköta kontakten via någon annan kanal, tex telefon för att få mer återkoppling. Vi har lärt oss vikten av frekvent och kontinuerlig återkoppling.

Utvärderingarna som gjorts veckovis i gruppen har också varit väldigt viktiga för projektets utformning. Genom att identifiera våra problem har vi i varje sprint kunnat se en positiv utveckling. Ett exempel på detta var när utvecklingsmiljön strulade och vi vid utvärderingarna kunde kommunicera hur olika problem lösts men också gå vidare med att arbeta på de datorer där vi fått det att fungera.

Att varje vecka göra utvärderingar har också gjort det möjligt att gå tillbaka och titta på vad som skrevs föregående sprint och därmed se tydligt vad vi åstadkommit. Det har varit kul att varje vecka kunnat se att utveckling skett vilket även fungerat som en morot till att genomföra reflektionerna. Det gör även att vi är inställda på att använda oss av ordentliga reflektioner även i andra projekt.

16 Praxis för arbetssätt och verktyg

Gruppen satte omgående upp en Slack-kanal och en Scrumboard via Trello. Utöver det skapades ett gemensamt repo på GitHub. Genom att diskutera vilka behov gruppen hade av att kommunicera konstaterades att kanalerna var lämpliga.

För att underlätta arbetet med nya tekniker utformade Linus en guide för att arbeta med Git. Genom att läsa guiden kunde de i gruppen som inte arbetat med Git tidigare enkelt förstå grunderna till versionshantering, vilket säkerligen minskade antalet misstag som gjordes. Guiden beskrev även grundläggande reglersom att alltid köra PULL innan PUSH.

I Scrumboarden kunde gruppen skapa user stories, epics, product backlog och sprint backlog. Trello visualiserade Scrumboarden och gav en bra överblick över vad som skulle göras, vad som hade gjorts och vad som pågick. Genom att utforma en Scrum-guide kunde tasksen utformas på ett lämpligt sätt, där det var viktigt att beskriva uppgiften förståeligt, samt att förklara vart resultatet hamnade, och hur det dokumenterades. På så sätt skulle vem som helst i gruppen förstå hur en task skulle hanteras, och var resultatet från uppgiften fanns. Gruppen följde dessvärre inte alltid den guide som utformats, utan gjorde till viss del slarviga kort på Scrumboarden som var svåra att förstå för någon annan än den som skrivit kortet själv.

Gällande versionshantering använde gruppen sig av Git. Genom att konsekvent skapa nya branches för nya uppgifter kunde merge-konflikter undvikas i största möjliga mån. Dessutom var det lätt att, som ovan beskrivet, hitta den plats i koden där utvecklingen av en specifik feature fanns. Brancherna namngavs efter kortet på Scrumboarden för att underlätta identifikationen. Branchernas namn var i sig inte informativa, så till framtida projekt kan det vara föredra att även ge dem en mindre beskrivning för att slippa referera till Scrumboarden. Det skapades även en ny branch för varje liten förändring, vilket potentiellt kan läggas upp på ett bättre sätt.

Om samtliga i gruppen redan från början hade känt sig bekväma med versionshantering hade arbetet möjligen varit mer effektivt. Det var dock svårt då olika förkunskaper fanns. De som är kunniga på ett område var en stor tillgång för andra i gruppen genom att dela med sig av sin kunskap, att dra nytta av denna fördel tar vi med oss till framtida projekt.

Önskvärt hade varit att ännu mer utförligt beskriva och dokumentera de user stories och uppgifter som skapades i Scrumboarden. Tydligare beskrivningar hade kunnat underlätta samarbetet och förståelsen för vad uppgiften innefattade. För att göra detta är det lämpligt att gruppen gemensamt kommer överens om exakt hur väl beskriven en uppgift behöver vara, samt i vilken utsträckning den ska dokumenteras för att undvika oklarheter. Dessutom är det viktigt att gruppen faktiskt följer det arbetssättet som satts upp. Det kan exempelvis uppnås genom att gemensamt gå igenom de olika uppgifterna inför varje sprint och säkerställa att samtliga medlemmar förstår innebörden av en beskrivning. I ett större eller längre projekt kan detta också vara nyttigt av kontinuitetsskäl, då olika personer kan vara involverade i projektet. En enhetlig beskrivning av uppgifterna kan då effektivisera när personerna byts ut.

17 Litteratur och gästföreläsningar

Jämfört med andra kurser på Chalmers har det varit relativt få föreläsningar och gästföreläsningar. Eftersom det är en projektkurs är det fullt rimligt att det primära fokuset ligger på projektet. Det

var dock en del föreläsningar i början av kursen som gick igenom de agila processer som skulle användas samt olika exempel för att illustrera dem.

Lego-övningen som hölls tidigt i kursen var väldigt lärorik och åskådliggjorde många av de problem som kan uppstå i projekt. Det var ett bra sätt att introducera Scrum-metodiken eftersom det blev tydligt för gruppen vilka funktioner den fyller och vikten av att använda sig av den under kursens gång. Övningen fungerade även bra för att gruppmedlemmarna skulle lära känna varandra.

Vidare fanns det mycket relevant litteratur på kurshemsidan, bland annat om hur man kom igång med utvecklingsmiljön, länkar till vart man kunde lära sig mer om Git, med mera.

Det hölls en gästföreläsning under kursen där en anställd hos Trine föreläste om hur agila processer används i industrin. Denna gästföreläsning gav gruppen mycket värde i mån av att förstå hur dessa processer appliceras i arbetslivet, samt poängterade värdet via hur man lätt kan ändra fokus vid omställningar eller marknadsförändringar.

I början var det lite svårt att ta till sig eftersom det är så olikt de arbetssätt vi tidigare använt, men alla dessa delmoment utgjorde en bra grund för att arbeta vidare på egen hand i gruppen och få applicera verktygen på riktigt. Det är ett arbetssätt som vi kommer ta med oss till framtida projekt där ett agilt arbetssätt lämpar sig då vi upplevt att det fungerat väldigt väl.

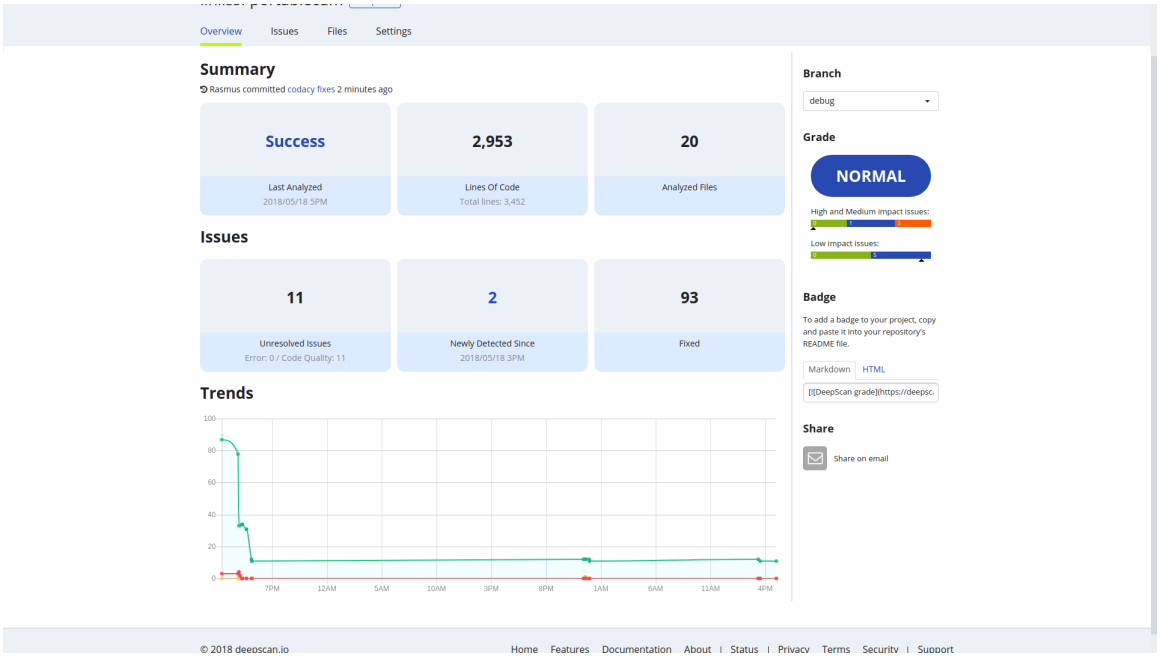
Gruppen fokuserade främst på att läsa kurslitteratur under projektets början. Något som möjligen hade varit värdefullt hade varit att löpande återkomma till kurslitteraturen för att bättre förstå hur Scrum, git och andra verktyg fungerade. Gruppen ansåg dock att det var bra att tidigt läsa och ta till sig av utbildningsmaterialet för att kunna starta projektet med mycket kunskap.

A Appendix

En beskrivning av de verktyg som använts för att analysera kodens kvalité.

A.1 Deepscan

Deepscan använder sig av andra projekt för att analysera och jämföra kodkvalitét och anger sedan vad som behöver ändras samt vilken påverkansgrad, impact, problemet har. Det finns sammanlagt tre olika grader low, medium och high. Low impact meddelar om kodkvalitet och underhållningsfel men problem inom denna kategori kommer inte leda till något kompileringsfel. Medium impact varnar om fel som gör att koden inte kommer att exekveras så som användaren har tänkt. High impact berör problem som leder till exekveringsfel eller ett undantag, Exceptions, kastas. Förutom påverkansgraden har även Deepscan ett betygsystem totalt sett på alla filer som den analyserar. Graderna är Poor, Normal och Good. Det är detta rankningssystem som använder sig av andra projekt och rankningen är baserad på densiteten av antalet impacts per 1000 rader kod.

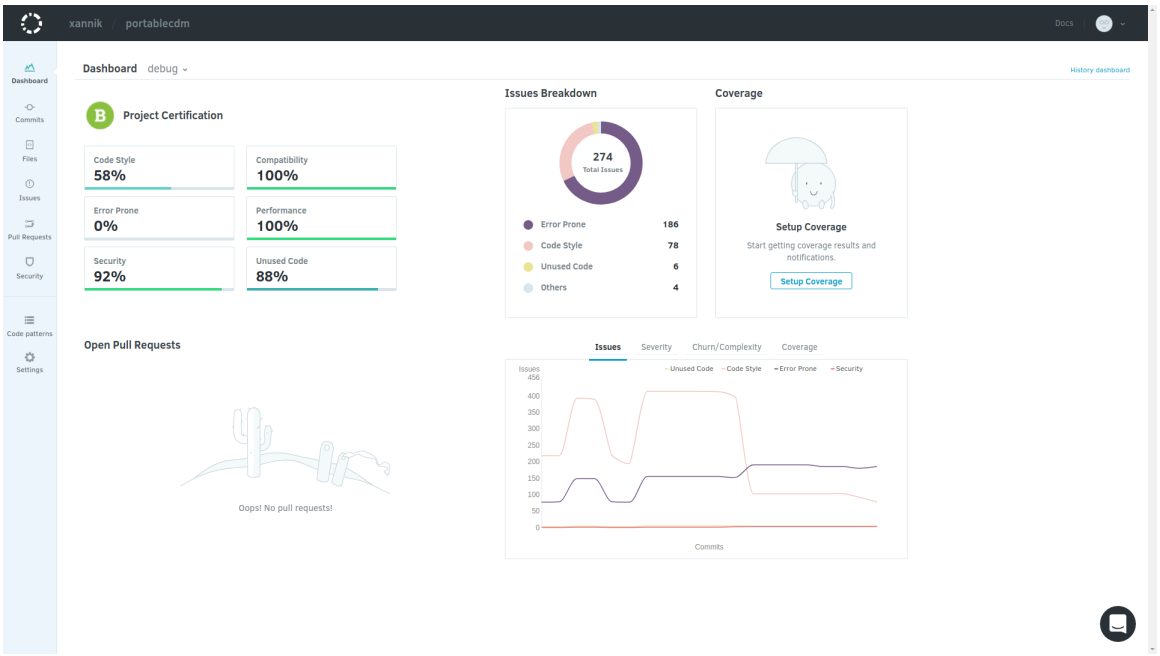


Figur 6: Skärmbild som visar sammanfattningen av analysen gjord av Deepscan över gruppens kod

Bilden ovan är tagen efter att de delar som skrivits från gruppen var fixade. Under rubriken trends kan man se en tidslinje på hur report har förbättrats under tiden vid varje git commit. Förutom rubriken Trends fanns även issues där användaren kan se kvarstående fel, nya fel och antal fel som blivit lösta. Under rubriken Summary fås en status på hur analysen har gått, antal rader kod samt antal filer som blivit scannade. När gruppen konstruerade nya vyer för applikationen var det filer som behövdes för att använda vyn i applikationen. Dessa filer var redan skrivna och innehöll också en del low issues. Dessa issues valdes att inte lösas av gruppen då dom inte blivit modifierade.

A.2 Codacy

Codacy var det andra verktygen som användes och är mer avancerat än DeepScan. Codacy har features som kan läggas till i projektet genom att skriva konfigurationsfiler i root-mappen av projektet, som den sedan läser av innan analysen. Codacy använder sig av kodmotorer för att analysera koden och kategoriserar problemen inom Code Style, Security, Compatibility, Performance, Error Prone och Unused Code. Det finns stöd för ett antal olika språk och användaren måste specificera vilket språk som används i projektet, så att analysen blir så exakt som möjligt. Allting som blivit ändrat under en git commit analyseras och får ett betyg mellan A till F. Kalkyleringen av betyget sker på liknande sett som i DeepScan.



Figur 7: Skärmbild som visar sammanfattningen av analysen gjord av Codacy över gruppens kod

Codacy ger precis som Deepscan en överblick över analysen av projektet där användaren kan välja att titta på olika typer av grafer utifrån vad som ska undersökas. Graferna beskriver hur kvaliteten av projektet har ändrats under tidens gång. Under det totala betyget finns en procentuell skala i dom kategoriserade problemen, motsvarande det totala antal problem som hittades på projektet. Dessutom, kan användaren gå in på enskild kategori för att se var i koden problemen finns.