Slutlig Reflektion

DAT255 - Dreamteam

Andreas Opedal Eriksson
Anton Albinsson
Filip Wendelin
Jenny Sorsa
Martin Ahlberger
Mattias Eriksson
Max Nyman
Staffan Hellsvik

Introduktion

Följande rapport behandlar processen i ett mjukvaruprojekt för kursen Software Engineering Project, i vilken en prototyp för applikationen PortCDM utvecklats. Applikationen ska ge praktisk tillämpning av tidigare programmeringskunskaper och samtidigt ge nya kunskaper inom agil arbetsmetodik applicerat inom IT-projekt. Projektgruppen har under kursen samarbetat med RISE Viktoria, vilka kommit med information och feedback genom hela utvecklingsprocessen.

Bakgrund

Den uppgift som tilldelats projektgruppen är att undersöka lotsens behov och därefter utveckla den befintliga applikationen utefter dessa. Lotsen ansvarar för att ”leda” fartyg in och ut ur hamnen och har ett stort ansvar för att se till att inga fartyg går på grund eller på andra sätt hamnar fel vid infarten- och utfarten i hamnen.

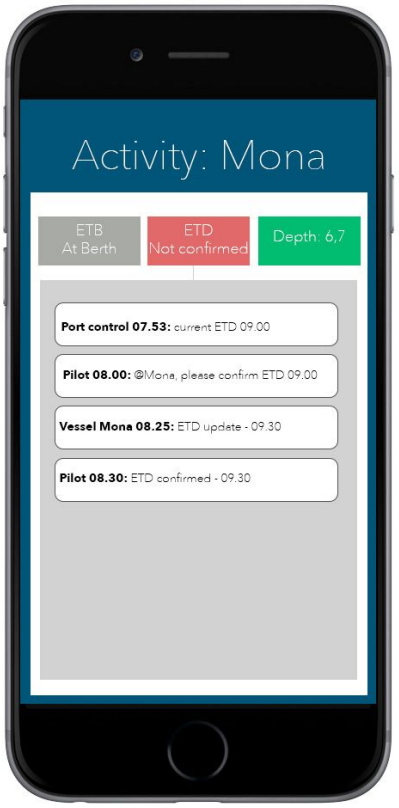
I dagsläget styrs lotsarna från en lotscentral varifrån även andra funktioner i hamnen styrs. Från lotscentralen planeras de olika lotsarnas scheman, så att de hinner mellan sina uppdrag. I hamnen kan det uppstå förseningar av flera olika anledning t.ex. dåligt väder, att andra båtar är försenade eller att en lastning tar för lång tid. När det sker förändringar i schemat är det lotscentralen som planerar om. De är därför intresserade av att så fort som möjligt få reda på avvikelser från schemat. De upplever problem med att fartygens förändringar inte når fram till lotscentralen utan kanske bara går till fartygens agent. De hade därför gärna sett en centraliserad kommunikationsplattform där fartygen, agenterna och lotscentralen kunde kommunicera om förändringar i planeringen.

Denna kommunikationsplattform skulle kunna vara PortCDM och flera idéer om hur möjligheten att förbättra för lotsen har behandlats under kursens gång.

Vision

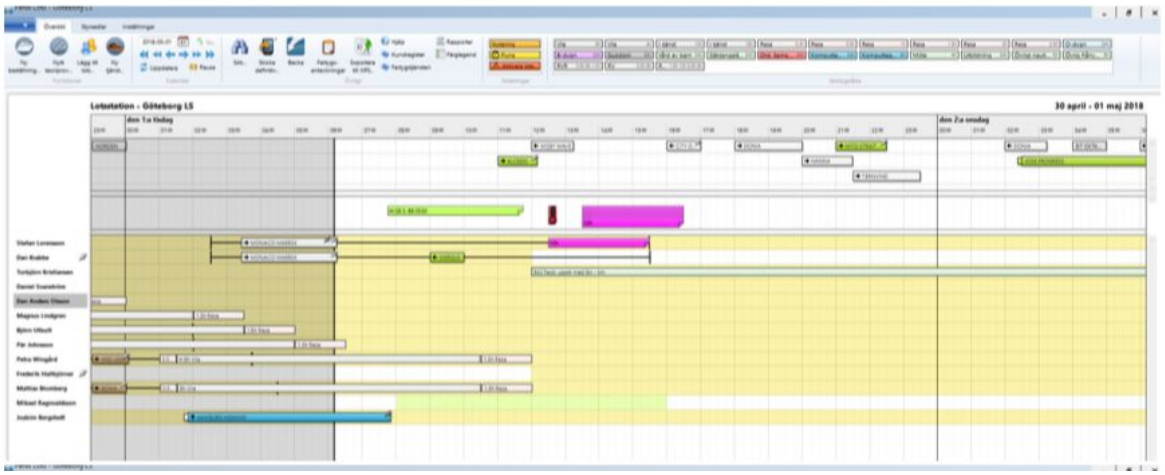
Projektgruppens ursprungliga vision för applikationen var främst tre olika funktioner:

1. Ett visuellt Gantt-inspirerat schema där de fartyg lotsen valt att se visas, med syftet att på ett tydligare sätt ge en översikt av den dagliga tidslinjen för vilka fartyg som anlägger eller lämnar hamnen och när detta sker.
2. En chattfunktion där varje aktivitet (en aktivitet för varje fartyg) via ett kommentarsfält kan diskuteras av de olika aktörer som lotsen samarbetar med, med syftet att fastställa den faktiska tidpunkt som gäller för den aktuella ”statusen” (ETA eller ETD). Se visionsbild nedan.
3. Möjlighet att skicka notiser till fartyget och andra delar av kedjan, för att på så vis kunna få en uppdaterad status gällande de tider som gäller.



Figur 1: Visuell mock-up av ursprunglig idé med kommentarsfält för varje aktivitet.

Visionerna blev framtagna efter diskussioner från möten med lotspersonal och product owners. Under processen fattades beslutet att enbart fokusera på en av dessa visioner, då tidsomfattningen av utvecklingsprojektet inte var tillräcklig för att hinna med att implementera samtliga funktioner. Den första punkten ansågs genomförbar och användbar och kunde dessutom inspireras av ett mjukvaruprogram som användes av lotscentralen i dagsläget, fast via deras datorer. En skärmdump från detta program, kallat Fenix, illustreras nedan.



Figur 2: Skärmdump av befintligt mjukvarusystem (Fenix) använt av lotscentralen.

Samtliga av dessa funktioner har representerats som epics i Trello, men det är enbart Gantt-schemat som utvecklats till user stories och implementerats i applikationen.

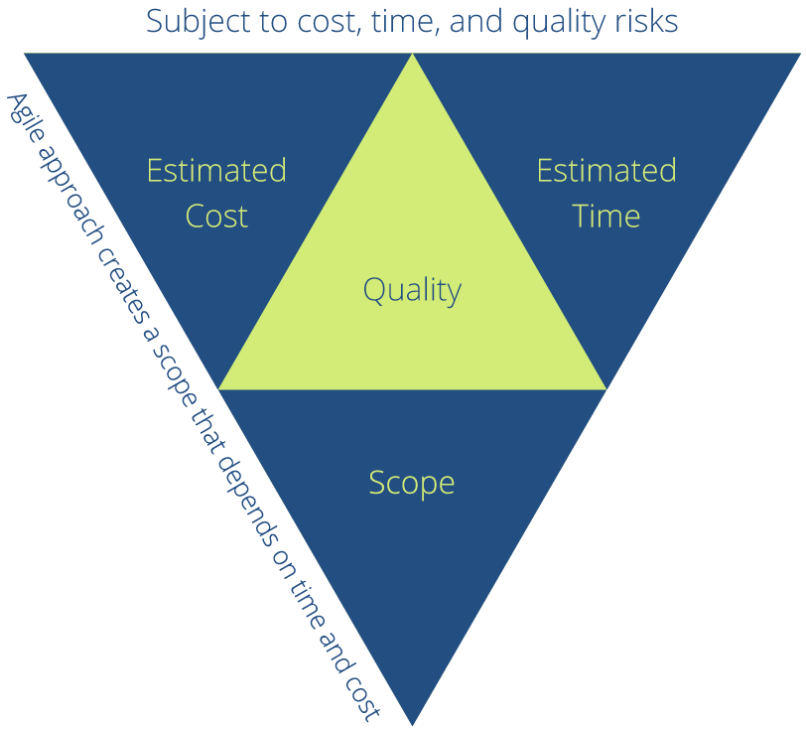
The chosen scope of the application under development including priority of features and for whom you are creating value

Now

Att skapa sig en bild av projektets omfattning var svårt under de första två sprintarna, då vi varken mött lotspersonal eller product owners och därför inte nått en förståelse för vad som efterfrågades eller behövdes. Det första vi gjorde i sprint 2 var att möta lotsen på lotscentralen. Väl där gavs en grundlig genomgång av inte bara lotsens arbetssätt och uppgifter, utan även hur samarbetet mellan lots och andra närliggande delar av ”kedjan” fungerade. Detta var hjälpsamt i den senare utvecklingsprocessen då det gav en helhetsbild av varje aktörs bidrag i hamnarbetet och hindrade oss från att isolera tankarna kring enbart lotsens roll. Vi kunde efter detta möte börja formulera en första tanke kring projektets scope, vilket ledde till de epics som beskrevs tidigare. Under den tredje sprinten hade vi det tredje mötet med PO’s, vilket gav ytterligare en bättre bild av vilka features som önskades. Här kunde också en prioritering av epics diskuteras, varefter vi själva kunde besluta om en definitiv prioritering. Projektets scope beslutades täcka in endast den högst prioriterade epic, vilket sågs som rimligt baserat på den tid som skulle läggas på kursen samt de resurser vi i teamet hade till förfogande (se figur nedan för samspelet mellan scope, resurser/kostnad och tid). Denna epic inkluderade, som tidigare nämnt, ett schema för de fartygsanlöp som kräver lots. Värdeskapande skulle ske främst för personalen som planerar in lotsning till fartyg, genom att de med hjälp av denna funktionalitet skulle få en översikt på all relevant data som finns i PortCDM.

När vi påbörjat utvecklingen av user stories insåg vi att vi gjort för stora user stories. Vi bestämde oss då för att bryta ner dessa i mindre delar för att hinna färdigställa fler user stories under sprintarna framöver. Detta beslut testades under följande sprint och utvärderades senare att vara ett väl anpassat beslut då det bidrog till färdigställandet av en user story. Under processens gång har varje skapad user

story utvärderats och stundtals reviderats för att matcha den kunskap vi nått under processens gång. Genom att möta PO's varje vecka har vi kunnat kontrollera att vår bild av värdeskapande överensstämmer med PO's behov och önskemål, vilket det gjort varje vecka. De avslutande sprintarna har handlat om att varje vecka kunna påvisa förbättringar och framsteg gällande applikationen från föregående och utvärdera utvecklingen under veckoplaneringen. Överlag har varje mål gällande applikationens omfattning och features uppnåtts nästan varje vecka.



Figur 3: Visar hur ett projekts scope beror på kostnaden och tiden i ett agilt mjukvaruprojekt.

Goal

I framtida projekt är målsättningen att i ett tidigt skede få koll på omfattningen av projektet och försöka uppskatta vilka delar av projektet som är värdeadderande för de berörda parterna, samt definiera vilka som det skapas värde för. Detta för att kunna få en gemensam målbild med produktägare och användare och på så sätt kunna utveckla user stories som skapar värde redan från början. Till exempel skulle möten med både slutanvändare och produktägare kunna vara prioriterat för första veckan av ett nytt projekt, eftersom att dessa parter förmodligen har ännu större beslutsfattande ansvar i "verkliga" projekt.

Gap

För att uppnå målet behöver vi ställa många frågor, möta product owners tidigt, läsa på om förväntningar och förklaringar kring projektet. "Prioriteringslistan" av vilka delar av applikationen som är viktigast kan göras tidigt och justeras efterhand, något som kan vara bra för att skapa sig en bild av hur väl man uppskattar vad som är viktigt och inte. En möjlighet är att detta skulle kunna beskrivas i en KPI där produktägaren får bestämma på en skala hur värdeadderande olika produkter är, för att ge en tydligare bild om vilka delar som är viktigast för att nå ett lyckat projekt.

Your social contract, which means you should create one in the first week

Now

Den första veckan skapades ett socialt kontrakt vilket utvärderats och utökats under projektets gång. Fokus har främst riktats mot att se till att arbetsfördelningen gruppmedlemmarna sinsemellan har varit jämn, samt att samtliga medlemmar har en specifik roll. Under processens gång har vi lagt till regler gällande tilldelning av tasks, så att varje medlem själv ansvarar för att ta sig an egna uppgifter då det behövs eftersom att vi ville undvika att någon stod utan arbetsuppgifter eller upplevde förvirring gällande vem som gör vad. Då det sociala kontraktet skapats för att följas under projektet har det inte skett så mycket utveckling under de senare delarna av processen. Den främsta utvecklingen var alltså i inledningsfasen, därefter har det sociala kontraktet främst använts som en säkerhet för att samtliga medlemmar utför sina arbetsuppgifter på rätt sätt under hela processen. Samarbetet i teamet har fungerat smärtfritt, vilket troligtvis kan förklaras av att medlemmarna tillhör en relativt homogen grupp med samma ambitionsnivå och vana av arbetssätt. Därför har gruppen aldrig behövt hantera en konflikt, då varje motsättning tagits upp till diskussion direkt och eventuella beslut har fattats på plats. Om gruppen stött på mer motsättningar internt hade det sociala kontraktet varit mer värdeadderande vid konflikthantering eller liknande situationer.

Vid en tillbakablick över projektet finns det delar som saknas, men som hade platsat väl i det sociala kontraktet. Exempel på punkter vi gärna lagt till, för att ha något att hänvisa till om eventuella konflikter eller ifrågasättanden uppkommit (vilket det dock inte gjort), är:

- Vilka programmeringspar vi valt att arbeta i
- Vilka som testar vems kod
- Hur vi testar koden

Goal

I framtiden är målsättningen att ha ett socialt kontrakt där varje del av projektet skrivs ner, exempelvis valet av programmeringspar, metod för testing samt hur gruppen väljer att hantera diverse konflikter. Ju mer man arbetar med sociala kontrakt, desto naturligare kommer det att vara att använda dessa och följa de överenskomna regler som finns förebygga konflikter och förenkla konflikthantering, då det blir en vana att ha det som ett element i varje projekt. Om gruppmedlemmarna inte känner varandra speciellt bra sedan innan och om dessa har olika syn på projektets ambitioner blir ett gruppkontrakt och dess innehåll troligtvis mer nödvändigt. Sociala kontraktet bör tydligt definiera vilka konsekvenser som ska vidtas om någon medlem inte fullföljer sina åtaganden, för att tydliggöra vikten av att följa kontraktet.

Gap

Att i ett tidigt skede implementera ett socialt kontrakt som samtliga medlemmar skriver under och följer under projektets gång. Genom att diskutera hur teamet vill arbeta och därefter fatta tydliga beslut gällande kommunikation i gruppen kan man därefter kontinuerligt se över att det sociala kontraktet följs samt ha tydliga beskrivningar för att enkelt kunna bestämma ifall en punkt inte har fullföljts.

The success criteria for the team in terms of what you want to achieve with your application

Now

Under projektets tre första veckor utvecklades ingen success criteria eftersom att vi först behövde kännedom om applikationen och projektet, samt ha fått igång utvecklingsmiljön innan det kunde skapas någon kod. Efter det första mötet med lotsen skapades en första bild av success criterias, då vi fick en bättre förståelse för vad som krävdes i utvecklingsprocessen. Senare i processen skapades två framgångskriterier för den epic som täcktes in av projektets scope:

- *Grafisk framgång:* att vyn i appen ser ut som vår prototyp och att allt är korrekt representerat.
- *Logisk framgång:* Att visningen i appen fungerar som vi har föreställt oss det. Till exempel att rätt states är representerade och visar rätt tid eller att du kan klicka på ett portcallet och överföras till tidslinjen för det portcallet.

Goal

Det har varit en tankekrävande process att komma fram till framgångskriterier och därför dröjde det några veckor innan vi gjorde det. På så sätt tappades värdefulla veckor, eftersom vi till en början inte visste vad vi skulle fokusera på. Till nästa projekt ska vi försöka komma fram till tydliga framgångskriterier i ett tidigare skede för att ge oss mer värdeadderande arbetstid. Exempel på framgångskriterier som vi i framtiden ser kan vara användbara handlar både om scope, budget och tid. Syftet med framgångskriterier är att att vägleda utvecklarna så att de arbetar i samklang med produktägarna.

Gap

Att i nästa projekt ta fram tydliga framgångskriterier tillsammans med relevanta intressenter i ett tidigt skede, som där det är relevant beskrivs i kvantitativa termer för att undvika eventuella missförstånd. Effekten av detta blir att arbetet får ett tydligare mål och att arbetet som presteras får ännu större relevans för produktägaren och slutanvändaren.

Your acceptance tests, such as how they were performed and with whom

Now

En genomgående diskussion vi haft under våra sprint plannings har varit huruvida vi bör och är redo för att utföra acceptanstester med slutanvändarna. Vi har varje sprint känt att vi inte skapat tillräckligt med värde för att ett acceptanstest skulle vara rimligt att göra, detta eftersom funktionalitet som vore redo att integreras i en slutapp inte varit färdig förrän den sista sprinten. Därför tog vi det strategiska beslutet att avstå från att göra acceptanstester då vi under detta projekt haft en tydlig utveckling och stor förbättringspotential gällande de övriga aspekterna i processen. Vi valde således att lägga våra begränsade resurser där de kunnat ge bäst nytta i utvecklingen av gruppen istället för att sprida gruppen tunt över ett stort antal parallella aktiviteter i projektet. Vi är väl medvetna om att vår uppfattning av vår produkt inte nödvändigtvis uppfattas på samma sätt hos slutanvändaren. Tack vare de avstämningar som gjorts mot slutanvändare och produktägare under hela projektet var dock den slutgiltiga feedbacken från både slutanvändare och produktägare positiv.

Goal

Med det sagt ser vi definitivt värdet med att utföra acceptanstester för att få värdefull feedback och kunna optimera värdet för slutanvändarna. Målet vi till slut fastställde för det här projektet gällande acceptanstester var att utföra ett acceptanstest efter vi är färdiga med vår första epic. Denna epic blev dock inte klar förrän sista sprinten, och då fanns ingen tid för att utföra ett acceptanstest. Till framtida projekt tar vi med oss att utformningen av acceptanstester, samt hur frekvent de ska utföras, varierar ganska mycket med typen av projektet och dess scope. Målet blir därmed att i framtiden utforma acceptanstester vid lämpligt tillfälle och baserat på hur projektet ser ut. Kopplat till detta bör ett mål vara att bestämma om vilken nivå av resultat som krävs för att resultera i en förändring av applikationen.

Gap

Förutsatt att projektets utformning är tillräckligt omfattande för att förväntas nå slutanvändarnas krav kan man i ett tidigt skede diskutera lämpliga acceptanstester. Beroende på vilket typ av projekt det är bör acceptanstesterna anpassas utefter vem som ska använda applikationen och på vilket sätt, något som förmodligen görs bäst genom att läsa på kring de olika typer av acceptanstesterna och försöka matcha “rätt”. Vilket threshold som ska sättas kring resultaten bör anpassas efter vilken typ acceptanstest samt hur vilken viktig respektive funktionalitet är. Om gruppen är osäker på vilket test som är bäst anpassat bör man utgå ifrån “learning by doing” och helt enkelt testa sig fram till bäst lämpade acceptanstester genom att sätta upp “mål” för vad man vill ha ut av testningen och se vilket test som uppfyller målen bäst. I projektet hade vi exempelvis kunnat använda ett user experience test, där vi låter slutanvändaren använda appen och ge värdefull feedback. En A/B- eller multivariate-testing hade troligtvis varit svårt då dessa tester kräver ett stort urval för att ge ett statistiskt säkerhetsställt resultat.

Ett annat sätt att testa vår design hade varit genom metoden *Design walkthrough* som kunde gjorts med 10-15 personer som ska använda vår app. Med den metodiken hade vi kunnat testa om våra slutanvändare hade förstått vår design och därmed upptäcka eventuella brister.

The design of your application (choice of APIs, architecture patterns etc)

Now

Inledningsvis var det problematiskt att förstå de designmönster som ligger bakom applikationen, då samtliga delar av applikationen utvecklats av någon annan och presenteras utan vidare beskrivning för varje del. Gruppen har arbetat utifrån ”learning by doing”-metodik och därför utvecklat förståelse för applikationens delar under projektets gång.

Dock har fokus inte legat på att nå en djupare förståelse för designmönster eller API:er, eftersom att det helt enkelt inte funnits någon dokumentation på existerande kod utöver det som skrivits för backenden. Backendens dokumentation har nyttjats för att t ex hitta information om de olika states som varit relevanta för lotsen, men det har inte varit tillräckligt för att förstå strukturen i appen, t ex så används Redux för kopplingen till backend, detta framgår inte någonstans utan kräver förkunskap om appens uppbyggnad - nyttjandet av Redux var något vi fick reda på genom handledningen, som på så sätt varit helt avgörande för att kunna få en rimlig arbetstakt under sprintarna.

Denna del av våra vecko-utvärderingar har främst tolkats som vår förmåga att förstå och använda den utvecklingsmiljö och språk som koden utvecklas i (då dokumentationen som sagt varit allt annat än utförlig). Ur detta perspektiv har det skett en tydlig utveckling gällande kunskap om Javascript och React Native. Processen har med tiden blivit effektivare, då varje individ helt enkelt lärt sig utvecklingsspråket bättre och fått en bättre förståelse för hur appen är uppbyggd genom learning-by-doing och kursens handledningstillfällen.

Goal

Beroende på hur projektet ser ut, kan det vara bra att tidigt i utvecklingsprocessen göra en genomgång av att identifiera eventuella designmönster som finns. När ett nytt scope presenteras bör en del vara att undersöka vilken funktionalitet som behövs hämtas externt och vilka APIer som krävs för detta.

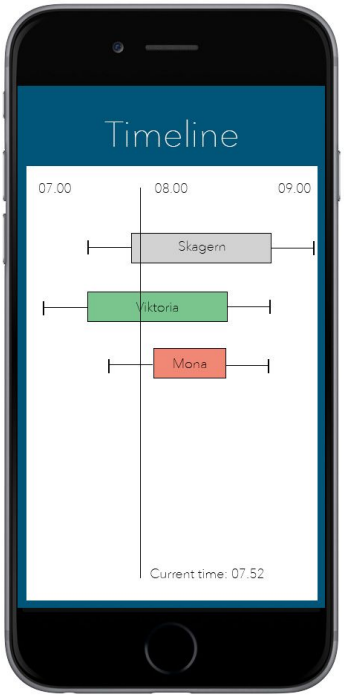
Gap

Tydligt analysera vilka förutsättningar vardera projekt medför, då den tid man kan och ”borde” (ur ett resursfördelningsperspektiv) lägga på att planlägga detta beror på projektets struktur. Om man ska bygga en applikation från grunden bör varje utvecklare ha en mycket bra bild av hur alla delar samspelar med varandra för att kunna skapa en bra applikation. Det är också viktigt att om ett API ska användas för att hämta extern data så bör API:et utvärderas efter dess tekniska implementation. Hur hög tröskeln är, hur länge man kan förväntas hämta relevant data från API:et och hur stabilt det förväntas vara över tid.

The behavioural overview of your application (for instance through use cases, interaction diagrams or similar)

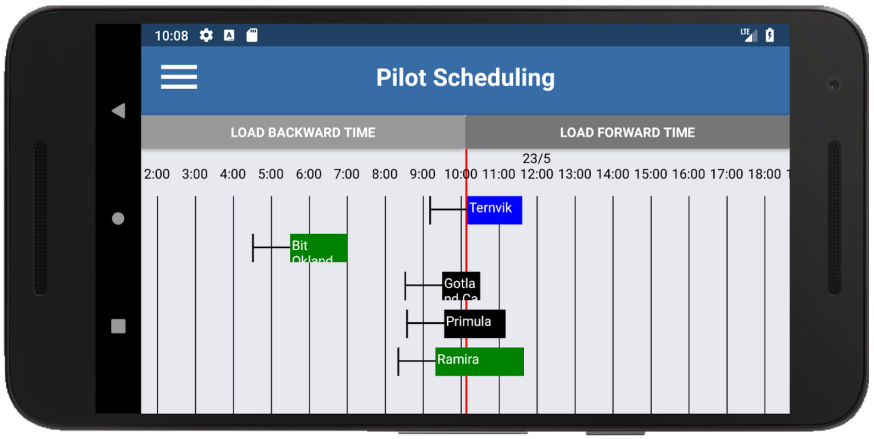
Now

Användarens förväntade interaktion med applikationen kartlades vid två olika möten med slutanvändaren. Den inledande hypotesen, att det var lotsen som var målgrupp för appen, kunde då förkastas eftersom användaren visade sig vara planeringsfunktionen i lotsorganisationen. Vi fick en bra bild av det förväntade användningsområdet, och det gränssnitt som lotsorganisationen var van att arbeta mot. Efter det första mötet fastställdes sedan i grupp en ny hypotes kring förväntade användningsområden, funktionalitet och gränssnitt. Detta kommunicerades till slutanvändaren och produktägaren, som båda var nöjda med den riktning vi valt. Vid ett andra möte med slutanvändaren bekräftades att det spår gruppen valt var det rätta. Tidigt kunde gruppen och produktägaren enas i en gemensam bild av det arbete som skulle tas fram tack vare att gruppen i ett bildbehandlingsprogram demonstrerade våra tankar kring den funktionalitet vi tänkt oss för slutanvändaren:

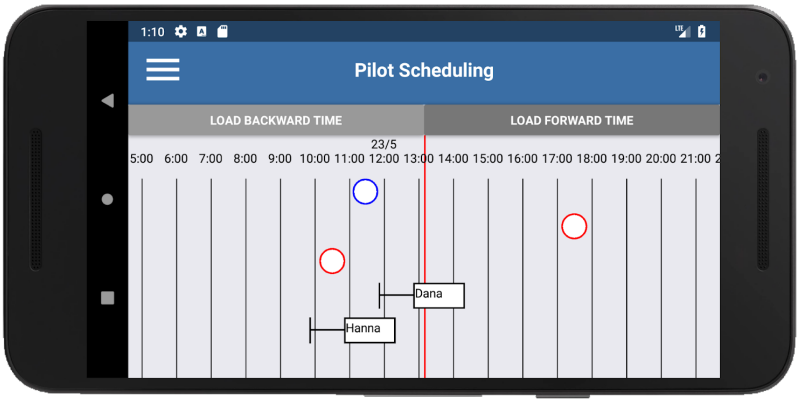


Figur 4: Visuell mock-up av det Gantt-schema som senare utvecklades i projektet.

Den slutgiliga produkten ser ut enligt följande:



Figur 5: Skärmdump från slutgiltig produkt som illustrerar ett gantt-schema med fartygsanlöp som har events av typen "pilotage_operation".



Figur 6: Skärmdump av produkten med funktion för requests, representerat i cirklar.

.

Vi har valt att färglägga portcallen för att visualisera korrektheten på datan. Färgerna på boxarna betyder följande:

- Grön = estimerad start- och sluttid
- Blå = pågående (starttid är aktuell men sluttid är estimerad)
- Svart = avslutad lotsning (både start- och sluttid är aktuella)
- Vit = Saknar starttid där vänstra strecket är utritad 1.5h före sluttiden.

Samtliga boxar har ett streck på vänster sida som visar 1h innan lotsen påbörjas. Detta ska förenkla planeringen av lots då det framgått att det är 1h inställelsetid.

Färgerna på cirklarna betyder följande:

- Röd cirkel = lots har blivit förfrågad
- Blå cirkel = lots har mottagit förfrågan
- Grön cirkel = lots har bekräftat förfrågan

När ett portcall som är en cirkel får en första tid transformeras den till en box och cirkeln försvinner. Vid klick på antingen en cirkel eller en box navigeras användaren till portcallets timeline-vy.

En rött streck är draget vertikalt för att visa den aktuella tiden. Tider är symboliserade som svarta vertikala streck. För att navigera till olika tider kan användaren dra med ett finger i sidleds. Det finns även två knappar, “LOAD BACKWARD TIME” och “LOAD FORWARD TIME“ , som justerar vilka datum som visas. Det är alltså möjligt att se historik samt långtgående framtida lotsningar.

Goal

I framtiden, i detta fall om projektet hade fortsatt, skulle vi också vilja implementera de andra visionerna/epics vi tog fram. Däribland finns notifikationer för att se att ett portcall har uppdaterats, antingen med en ny tid, en ny state eller liknande.

Lotsen var också intresserad av en chattfunktion mellan aktörer. Beroende på intresset från resterande aktörer, skulle chatten kunna implementeras för ett specifikt fartygsanlöp för att undvika ett ständigt brus.

Genom att fortsätta implementera ovan nämnda funktioner hade applikationens funktion som helhet förmodligen skapat större värde för slutanvändaren.

Gap

För att skapa notifikationer krävs research gällande vad som behövs för att implementera notisfunktioner i appen. Det första steget är att stämma av med slutanvändaren hur behovet av notiser ser ut. För att skapa detta hade vi sedan behövt läsa in oss på vad som krävs och hur man skapar det. Därefter behöver den inlästa personen lära sig de verktyg som krävs för att skapa notis-funktionen och slutligen se till att implementera den. Implementeringen skulle exempelvis kunna visa sig som en symbol på boxen (ex. ringklocka) som visar på en händelse skett för portcallet. En push-notifikation skulle fungera som en förlängningen på notifikationer som skulle ske inne i appen.

Samma lärandeprocess kan appliceras vid implementationen av en chattfunktion. Chattfunktionen skulle lämpligtvis implementeras för varje specifikt fartygsanlöp, för att på så sätt hålla konversationen relevant. En oro som dök upp i diskussioner kring chattfunktionen var att en sådan skulle kunna bidra till att viss information endast skrivs i chatten (och alltså inte läggs till som tidsstämplar). Detta problem skulle kunna lösas genom att man i chatten också implementerar

standardiserade meddelanden som automatiskt uppdaterar tiddstämplar. På så sätt behöver inte informationen skrivas in dubbelt, vilket troligtvis skulle uppskattas i hamnpersonalens ibland stressiga vardag.

The structural overview of your application (such as class diagrams, domain models or component diagrams)

Now
Frånvaron av class diagrams/domain models för den existerande applikationen medförde att det tidiga målet som sattes upp var att försöka kartlägga hur delarna i appen samverkar. Vid projektets start hade vi i gruppen mycket begränsade kunskaper om JavaScript vilket också bidrog till att hastigheten i denna kartläggning påverkades, och under de veckor som projektet pågått har fokus legat på att förstå vilka delar i programmeringsspråket som fungerar på samma sätt som det språk vi tidigare lärt oss, det vill säga Java.

Under gästföreläsningen från TRINE nämndes att denna typ av arbete inte bör vara prioriterat, och sällan är av värde, varför fokus lades om till att endast lära oss det som var nödvändigt för att klara av våra user stories och på så sätt nå så långt som möjligt mot produktägarens önskningar för appen. Genom den här approachen kunde vi också, indirekt, få en kunskap om appens struktur och sammansättning.

Som grupp är vi efter den sista sprinten vid en punkt där vi, med de insikter vi fått från vårt utvecklande av user stories, besitter tillräcklig kunskap för att rita grova UML-diagram eller liknande på appens sammansättning. Dock valde vi att lägga kvarvarande tid på att färdigställa en genomarbetad rapport då det inte anses värdeadderande för slutprodukten att skapa UML-diagram när kod-utvecklingen är avslutad.

Goal
Målet för nästa projekt, förutsatt att det liknar detta, blir att inte fokusera på den övergripande strukturen för programmet, om det inte ses som nödvändigt. Anledningen är att det inte skapar något värde för produktägaren och resurserna hellre läggs på user stories. På gästföreläsningen var inställningen till dokumentation av denna typ att skriva den men vara medveten om att ingen någonsin kommer att läsa den.

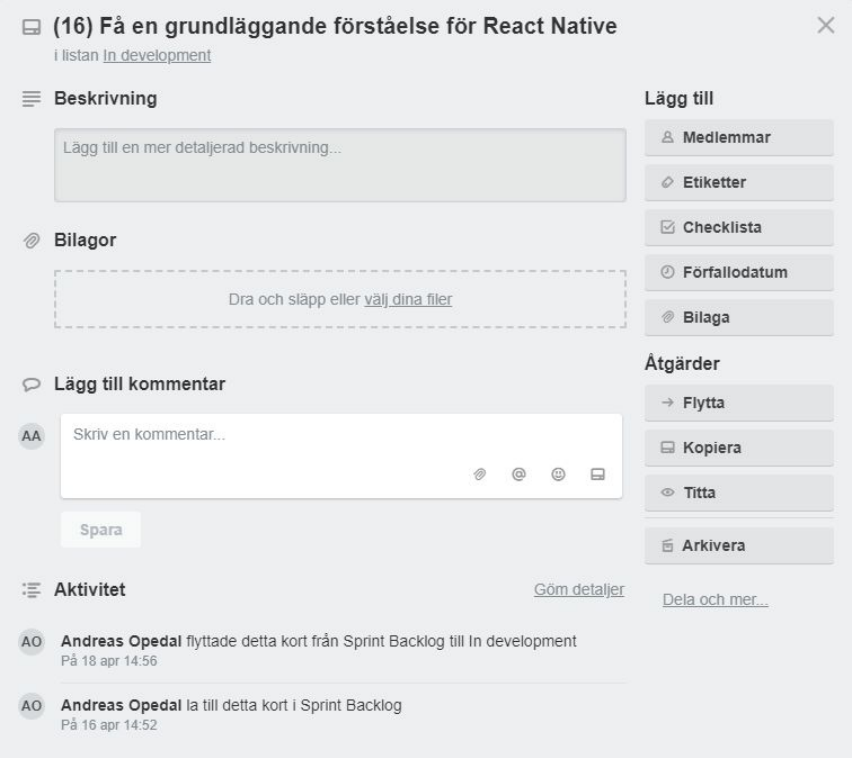
Gap
Ta med sig insikten att det troligtvis inte är av högsta prioritet att bygga en djup förståelse kring ett programs struktur och sammansättning, utan att det är av mer värde att fokusera på den del där man själv utvecklar. Förståelse för strukturen kommer sedan efterhand. Effekten av detta förhållningssätt är att det arbete som läggs ned får maximal värdeadderande effekt för produktägare och slutanvändare.

Your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation

Now

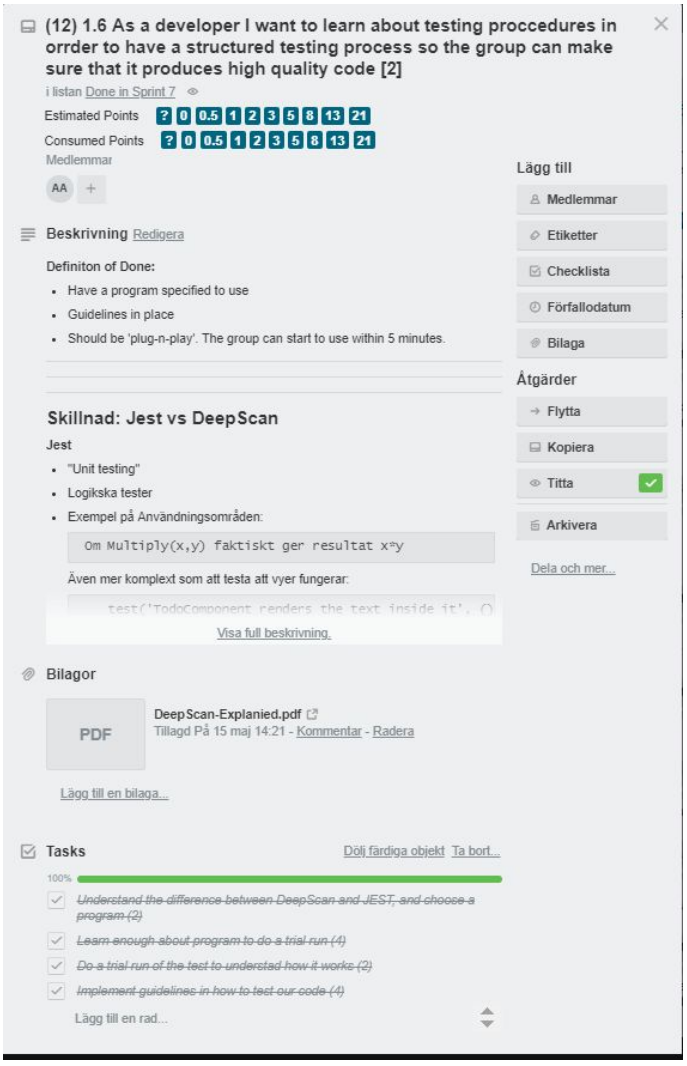
Utvecklingen av user stories korrelerar tydligt till hur väl projektgruppen förstått product owners behov. Fram tills att vi mött både lots och PO's fanns det ingen anledning att försöka skapa ”korrekta” user stories, istället lades fokus på att bygga upp en vision gällande lotsens behov utifrån det vi kunde läsa oss till. Senare justerades dessa utifrån de verkliga behov vi kunde identifiera efter mötena med lots och PO's . Därefter övergick processen till att försöka prioritera de user stories som ansågs rimliga att hinna utvecklas under projektets gång och samtidigt vara mest värdeadderande för lots och PO's. När dessa godkänts av PO's påbörjades arbetet med att skapa acceptanskriterier, vilket visade sig vara svårt utan att skapa Definition of Done's för varje task. När denna insikt nådde gruppen lade vi till DoD's vilket gav tydliga resultat i effektiviteten. En utmaning med utvecklingen av user stories och tasks var att estimera tidsåtgången i förväg, vilket förmodligen grundar i att ingen i teamet arbetat med varken React Native eller app-utveckling tidigare. I takt med att samtliga gruppmedlemmar blev mer bekanta med utvecklingsmiljön, arbetssättet och språket var det enklare att göra korrekta estimat. För att skapa motivationen hos teamet att göra bättre uppskattningar infördes detta även som en KPI. Standardmönstret gällande våra estimat har varit att först bryta ner en user story i tasks, uppskatta tiden för varje task och därefter lägga till alla task-beskrivningar för att kunna estimera tidsåtgången för hela user storyn.

Initialt saknades en vana att kunna skapa bra user stories i Trello vilket innebar att många user stories saknade exempelvis Definition of Done vilket gjorde det svårt att avgöra om vad som faktiskt var avklarat under sprinten. Här nedan beskrivs ett exempel på en tidig user story där det saknas en Definition of Done, var svår att estimera och hade tveksamheter kring testning. INVEST kriterierna (Independent, Negotiable, Valuable, Estimable, Small, Testable) för att utveckla bra user stories hade här inte tagits hänsyn till.



Figur 7: Skärmdump från Trello som visar en mindre bra user story tidigt i processen.

Nedan följer en user story som vi tog fram under senare delen av projektet med fokus om att kunna följa INVEST kriterierna. Den är oberoende från andra user story, kan förhandlas om vilken omfattning den bör utföra, skapar värde för oss som utvecklare, kunde estimeras, var tillräckligt liten för att hinnas med under en sprint och gick att testa ifall den var genomförd.



Figur 8: Skärmdump från Trello som visar en user story med estimat, faktisk tidsåtgång, DoD's, tasks och som följer INVEST-kriterierna.

Goal

Att ha en standardiserad metod (dvs ett arbetssätt som alla i teamet tagit fram och följer) för att bryta ner user stories i tasks och att estimera dessa på ett sätt som inte skiljer sig markant från det verkliga utfallet. Genom att ha denna målsättning kan teamet enklare mäta utveckling och effektivisering av metoden under projektets utveckling.

I ett framtida projekt är förhoppningen att sprint planning sker i närmare samarbete med PO's, vilket skulle innebära att prioriteringen av user stories formuleras med mer input från PO's och därmed motsvarar det verkliga behovet bättre. Viktigt att understryka är dock att utformningen av detta projekt inte involverade ett lika nära samarbete med PO's, därmed har processen anpassats efter detta.

Gap

Tidigt i projektet studera olika metoder för att ta fram korrekta user stories och ha god kommunikation med förberedda frågor kring problematik i dagsläget och önskad effekt av applikationen till PO's. Vi behöver även ta med oss lärdomarna, som att följa INVEST och skriva tydliga Definition of Done's. Förhoppningsvis ger detta effekten att samtliga teammedlemmar kan bidra med insikt i behov hos kunden och därför ta fram bra user stories på ett effektivt sätt.

För att veta hur teamet ska förhålla sig till prioritering av user stories i framtida projekt är det av vikt att bilda sig en uppfattning av hur involverade PO's kommer att vara i projektet samt hur stort beslutsfattningsansvar de önskar att ha.

The three KPIs you use for monitoring your progress

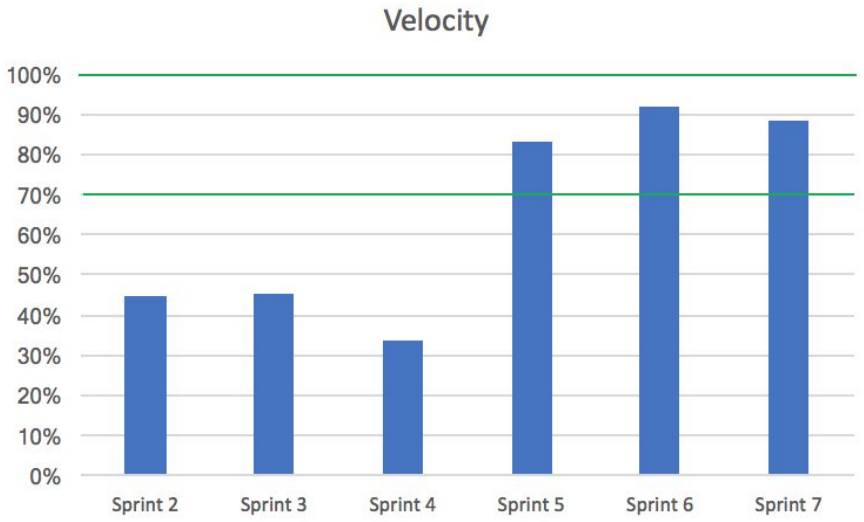
Now

Under den andra veckoreflektionen bestämdes att följande tre KPI:er skulle mätas:

- Träffsäkerhet i velocity
- Kommunikation
- Overhead

<u>Träffsäkerhet i Velocity</u>

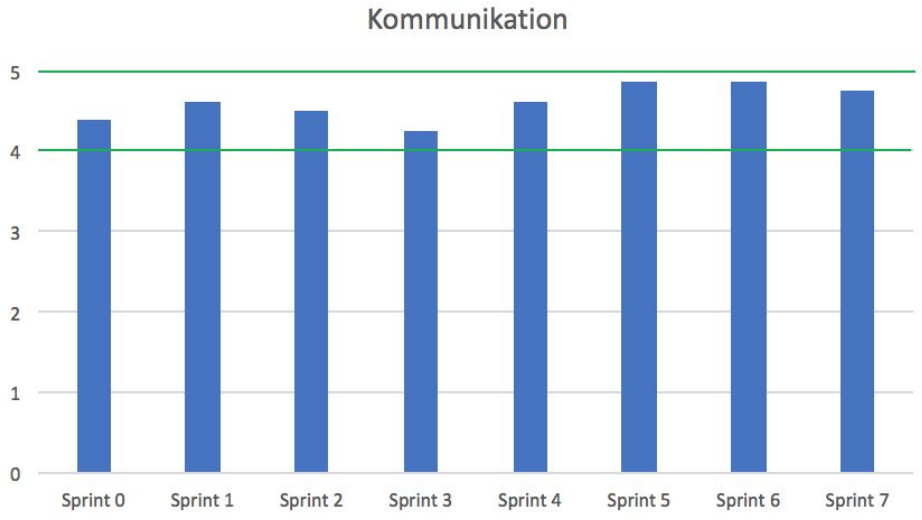
Träffsäkerhet i velocity mättes genom att vi jämförde velocity med avklarade user stories, för att därmed få ett mått på hur bra teamet kunde uppskatta user stories efter vald velocity. Velocity sattes varje vecka som antalet timmar gruppen kunde avsätta till värdeadderande aktiviteter (dvs tid för att arbeta på user stories) och blev mer eller mindre konstant. Därmed kunde vi isolera den andra variabeln - summan av estimaten för avklarade user stories. Vi kunde då mäta vår förmåga att uppskatta estimaten för user stories, samtidigt som vi mätte teamets förmåga att prestera. Den beräknades genom att dividera estimaten för avklarade user stories med velocity. Det initiala målet ansågs rimligt att placera inom ett 30% intervall från velocity, vilket ger ett intervall på 0,7-1,3. Senare insågs dock att ett utfall på större än 1 inte blev möjligt med Scrum-metodiken, eftersom att estimaten på alla user stories för en sprint summeras till velocity för sprinten. Målet reviderades då till >0,7.



Figur 9: Resultat från KPI:n Velocity. De gröna horisontella linjerna representerar godkänt resultat.

Kommunikation

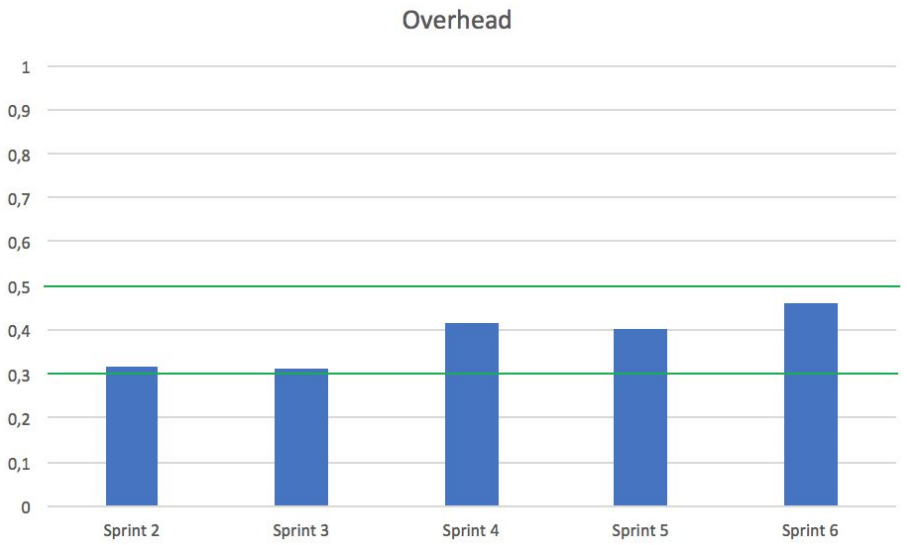
Kommunikation infördes som en mer "mjuk" och subjektiv KPI. Syftet med att mäta teamets kommunikation var att lyfta vad som fungerat bra och mindre bra, för att kunna identifiera förbättringsåtgärder för kommunikationen. Alla fick under varje sprint retrospective poängsätta sprintens kommunikation mellan 1 och 5. Sprintens kommunikationsvärde beräknades sedan som genomsnittet av dessa. Målet var att ligga på ett genomsnitt över 4.0, vilket vi lyckades uppnå varje sprint.



Figur 10: Beskrivning av varje sprints resultat i KPI:n "Kommunikation". De gröna horisontella linjerna representerar godkänt resultat.

Overhead

Overhead syftade till att mäta hur stor del av tiden som var värdeadderande för produktägaren. Varje teammedlem fick för varje sprint fylla i sin nedlagda tid uppdelad efter total tid, mötestid och tid för individuell reflektion. Den summerade mötestiden och tiden för reflektion dividerades med den summerade totaltid för att ge värdet på detta KPI. Från början var målet att hålla sig under 50%. Detta reviderades en vecka senare till intervallet 30%-50%, med insikten att en viss nivå av icke-värdeadderande tid var önskvärd för att sprint reviews, sprint retrospectives, sprint plannings och daily scrums skulle bli värdefulla.



Figur 11: KPI:n Overhead - hur gruppen fördelat sin tid under varje sprint fördelat på värdeadderande och icke-värdeadderande tid. Då denna KPI reviderades till sprint 7 är inte denna sprint med. De gröna horisontella linjerna representerar godkänt resultat.

Estimation Accuracy

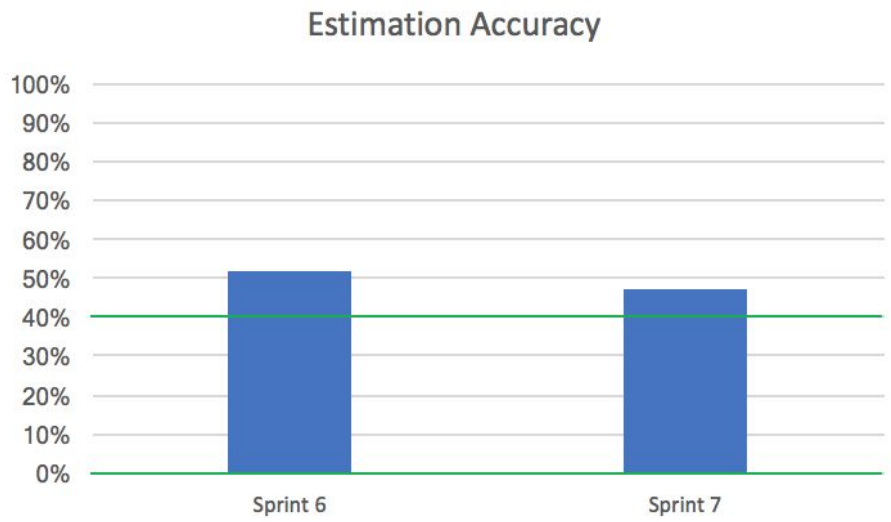
Efter vecka 5 infördes en ny KPI, som nyttjades för att fastställa hur träffsäkert vi fastställer våra estimat i våra user stories. Till skillnad från träffsäkerheten i velocity så mäter denna KPI träffsäkerheten i estimaten för varje enskild user story. Det diskuterades huruvida Velocity KPI:n fortfarande var relevant, men beslutet togs att behålla den som ett mått på teamets prestation för sprinten. Under samma vecka diskuterade vi även hur värdeadderande KPI:n overhead egentligen var eftersom våra mötestider visade sig vara mer eller mindre konstanta. I slutändan bevarade vi och anpassade KPI:n då vi ansåg att den tillfredsställde ett annat syfte, nämligen att ge motivation till att fylla i loggboken på arbetstiden. Istället för att mäta andel värdeadderande tid började vi sprint 7 att mäta antal ifyllda celler (vilket med totaltid, mötestid och tid för individuell reflektion gav totalt 24 celler). Målet var att alla skulle vara ifyllda.

För att mäta estimation accuracy beräknades först den faktiska tid det tog att genomföra en user story delat med estimatet för denna user story. Detta gjordes för varje user story som blev klar under sprinten för att sedan beräkna avvikelsen från 100% för var och en av dessa (ju lägre avvikelse desto bättre estimat). Utifrån detta fastställdes ett genomsnitt. Optimalt utfall för denna KPI var därför 0%.

Målet fastslogs till max 40%. Från tidigare erfarenhet med estimering kom gruppen genom diskussion fram till att ett fel på en arbetsdag (8 h) skulle vara rimligt. Detta motsvarar 40 % då sprinten består av 20 timmars arbete för varje lagmedlem.

Målet för estimation accuracy blev inte uppnått under någon av de två sprintar där det testades, vilket illustreras i tabellen nedan. Vi upptäckte att de estimat som ofta fick en störst procentuell avvikelse var de mindre, vilka ofta överskattades. Större user stories tenderade å andra sidan att bli underskattade. Tekniska user stories visade sig vara svårare att estimera än icke-tekniska. Troligtvis berodde detta på den bristande kunskap vi hade om Javascript och React Native, vilket ledde till en stor osäkerhet kring varje task.

Gällande sociala faktorer såsom gruppens förmåga att samarbeta har detta inte upplevts vara något problem under projektets gång. Tack vare en gemensam målbild och hög inre motivation hos samtliga gruppmedlemmar har det snarare varit aspekter så som att teknisk handledning varit svårt att få tillgång till (då det varit hög efterfrågan bland de flesta grupper) och det faktum att JavaScript varit ett nytt programmeringsspråk för en majoritet av gruppen.



Figur 12: Den sista KPI:n, Estimation Accuracy, som lades till i sprint 6. De gröna horisontella linjerna representerar godkänt resultat.

Goal

Efter att ha testat fyra KPI:er under projektets gång har vi en bättre förståelse för vad vi vill få ut vid framtida projekt. KPI:er känns som ett givande inslag i arbetsprocessen då de utgör en god utgångspunkt i diskussioner under exempelvis sprint retrospectives.

En viktig lärdom är att i ett tidigt stadie i arbetsprocessen implementera en KPI som mäter estimation av tidsåtgång för user stories. Det skulle förenkla sprint planning och ge en bättre uppfattning hur stort scope man kan ha på produkten. Vad vi förstått från litteratur kan magkänsla användas för att uppskatta tidsåtgång om det saknas historisk data (Scrum and the XP from the Trenches, s. 33). Vi har

både saknat historisk data samt haft dålig intuition för tidsåtgång, vilket är viktigt att förbättra inför framtida projekt.

I efterhand borde vi ha haft KPI för att mäta produktägarens nöjdhet med produkten och de ändringar vi gjort varje vecka. Bemötande från produktägaren har i projektet ständigt varit positivt, vilket försvårat vår uppskattning av vad de egentligen anser om produkten. Att använda sig av en sådan KPI blir därmed en målsättning för framtida mjukvaruprojekt med Scrum.

Utöver vilka KPI:er som är värdeadderande kommer vi också kunna ta med oss processen i att optimera resultaten på dessa till framtida projekt. Resultatet på kommunikation blev exempelvis gradvis bättre genom projektets gång och vi har genom detta kunnat ta med oss många lärdomar om vad som fungerat bra. Vi har till exempel lärt oss vikten av att kommunicera att en user story är redo för testning, så att den hinner testas av en annan i teamet innan sprinten är slut. Målet här blir därmed att ta med sig och implementera dessa lärdomar till kommande projekt.

Gap
Under projektets gång har vi börjat reda ut vilken typ av KPI:er vi tycker är värdeadderande. Men eftersom projektet endast varat under åtta veckor är vi ovetande hur KPI:erna kommer fungera under en längre tidsperiod. KPI:n estimation accuracy hann vi endast testa i två veckor innan projektets tog slut. Det är således svårt att uppskatta hur värdeadderande den är då det krävs fler sprintar för det. Men estimation accuracy upplevdes som ett givande KPI där det helt klart finns förbättringspotential, så det kommer att utvärderas till framtida projekt.

För att lösa gapet med produktägarens nöjdhet skulle PO kunna betygsätta ändringen från en skala på 1-10 från tidigare vecka för att peka utvecklarna (oss) i rätt riktning. En nöjdhets-KPI från PO:s sida skulle förtydliga kommunikationen, få en utvärdering på produkten svart på vitt samt se till att arbetet faktiskt är värdeadderande för uppdragsgivaren.

Gällande lärdomar för att KPI:ers prestation så har många av dessa antecknats under sprint retrospectives och gruppreflektioner. Gapet att implementera lärdomarna till nästa projekt löses därmed genom att kolla tillbaka på anteckningarna och ta med sig det som är relevant till projektet. Effekten av att förbättra KPI:erna kommer ge utslag i en effektivare arbetsprocess i framtida projekt.

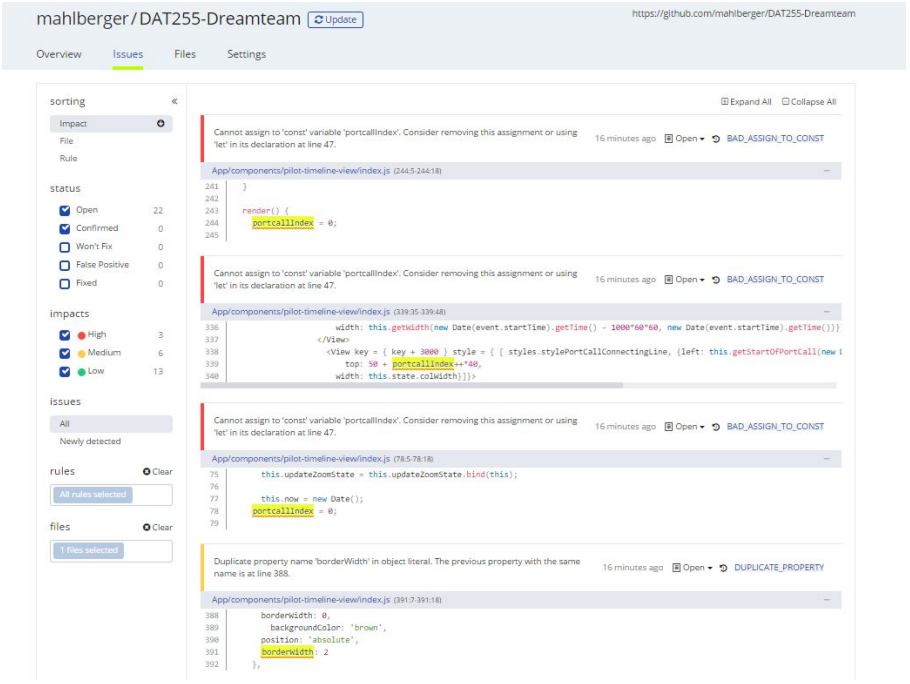
Code quality using a tool such as Findbugs (1 point if your code includes issues concerning correctness or bad style, 2 points if you have dodgy or performance issues and 3 points if the code is fine), only asses the code you have written yourself

Now
Inledningsvis tog det några sprintar innan vi kunde börja testa kod, med anledningen att vi helt enkelt inte kunde börja generera kod förrän några veckor in på projektet. Utvecklingsmiljön tog tid att få att fungera, varför kodningen inte kunde komma igång förrän senare. Gällande testningen använde vi

främst manuell testning utan vidare struktur, vilket inte varit helt optimalt men fungerande. Vi delade upp så att det på förhand var bestämt vem som skulle testa varje gruppmedlems kod. Under den näst sista sprinten bestämde vi oss av att inleda en process kring ett mer strukturerat arbetssätt för att genomföra testing. Vi tog fram en user story för att utvärdera olika testningsmetoder där vi efter feedback från andra scrumteam valde mellan JEST och DeepScan. Valet föll på DeepScan då gruppen ansåg att detta verktyg, med automatiskt statisk analys, var ett mindre tidskrävande alternativ som ändå gav värdefull input för att utvärdera kodkvalitén. Vi reflekterade över att JEST är ett kraftfullt verktyg men att den tiden som det tar att implementera detta innebar att vi genomförde ett strategiskt beslut att enbart köra DeepScan.

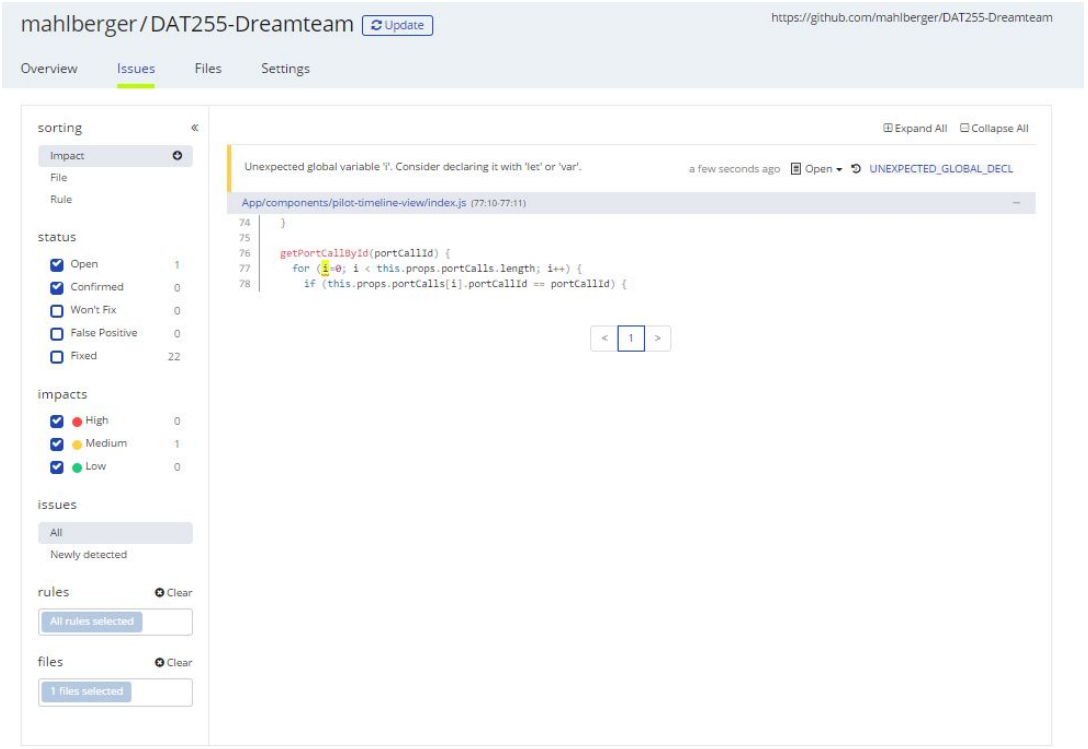
Exempel på användning av DeepScan:

Bilden nedan visar på hur den initiala testningen såg ut av en viss branch i giten. I fliken Impacts så ser vi hur många problem som hittas samt hur allvarliga respektive problem är.



Figur 13: Skärmdump från ett test i DeepScan innan problem är lösta.

Bilden nedan visar på testning efter att samtliga problem var lösta. *Note: Det går inte att få upp vyn om filen saknar fel. Därför har vi lagt in ett "dummy" fel för att kunna få fram den här bilden.*



Figur 14: Skärmdump från DeepScan efter problemen blivit lösta.

Goal

För framtida mjukvaruutveckling tror vi att ett mer organiserat användande av testing kommer vara en viktigt del för att skapa en kodbas av hög kvalitet som blir ännu viktigare när applikationens scope växer och allt mer kod återanvänds. En målsättning bör vara att kunna kombinera både statisk analys som Deep Scan tillsammans med enhetstestning i JEST eller liknande program för att se till att både logiken och koden stämmer överens med uppsatta krav. Detta är viktigt för att kunna säkerställa att koden som skapas ska kunna matcha de kraven som en framtida produktägare skulle kunna ställa.

Gap

För att kunna få ett mer organiserat användande av testning är det viktigt med etablering av en struktur/riktlinjer om hur testningen ska genomföras såväl som hur resultaten av en testsvit faktiskt ska tolkas. För att kunna nå en bra testningsprocess blir en viktig del att lära sig skriva bra enhetstester, där alla gruppmedlemmar som skriver testnings case lägger samma krav i form av undersöka extremfall för att exempelvis testa att logiken fungerar. Dessutom blir det viktigt att tolka resultaten av testsviten, exempelvis om vilka åtgärder som ska innebära ifall det upptäckts allvarliga kodbrister i jämförelse med småfel. Här behövs en kommunikation med produktägaren om hur stor del av tiden som bör läggas på kvalitet ske och därefter sättas upp threshold på testningen utifrån produktägarens krav.

The roles you have used within the team

Now

Rollerna i teamet sattes tidigt i projektet då det inte var beroende av projektet i sig utan snarare för arbetsmetodiken Scrum, något som enkelt gick att researcha via kurslitteratur, föreläsningar och sökningar på nätet. Teamet fattade demokratiska beslut gällande vilken roll som tilldelades vem. Detta gjordes genom att samtliga medlemmar fick säga vilken roll de önskade och ”motivera” varför denne önskade just den rollen, med syftet att placera rätt resurs på rätt plats och på så vis lägga grund för en effektivare process. Om någon inte hade några önskemål eller preferenser kunde övriga gruppmedlemmar nominera personen för en av de roller som fanns till förfogande. Här var det också viktigt att rollernas arbetsbörda var någorlunda jämnt fördelad för att ingen skulle arbeta mer än någon annan. Detta tillvägagångssätt fungerade väl, då vi alla önskat rak och tydlig kommunikation från start. Rollerna som sattes var följande:

- *Scrum master*
- *Person responsible of booking group rooms*
- *Secretary and meeting protocol responsible*
- *Person responsible for Trello*
- *Reminding-of-individual-reflection and uploading team reflection responsible*
- *Person responsible for contact with course examiner*
- *Git-responsible.*

Dessa roller togs fram under första veckan och baserades på de olika moment som gruppen antog skulle ingå i kursen. Under processens gång insåg gruppen att det stundtals slarvades med uppladdning av team reflections och individual reflections, varför en roll med ansvar för just detta skapades. Efter denna insats fungerade uppladdningen av dessa dokument bättre, eftersom att det kom flertalet påminnelser samt fanns en rutin för vem som gjorde vad. Rollfördelningen har under kursens gång fungerat bra och rollerna har diskuterats vid varje sprint review för att se till att vi under processens gång fortfarande har placerat rätt person på rätt position och därmed kan jobba så effektivt som möjligt.

Goal

Målet i ett framtida projekt, oavsett mjukvarubaserat eller inte, är att tidigt bestämma roller i teamet och hålla sig till dessa, förutsatt att det under projektets gång anses vara rätt fördelning av alla teammedlemmar. Detta för att det underlättar processen att ha någon som ansvarar för en specifik sak, och var och en kan då fokusera på att ens ”egen” del av projektet kontinuerligt fungerar som det ska och därmed se till att projektet som helhet är välfungerande och utvecklande.

Gap

Se till att skapa ett socialt kontrakt där varje roll specificeras tydligt. Det som även kan göras är att lägga till hur gruppen ska hantera effekter/konsekvenser om någon inte sköter sin uppgift som överenskommet. På detta sätt kan man undvika eventuella konflikter genom att hänvisa till det som alla initialt kommit överens om i kontraktet. En effekt av att specificera sina roller tydligt kan också vara att bidra till inre motivation och därmed bättre prestation hos individen, eftersom att denne känner att den har ett eget ansvar att ro projektet i hamn och inte kan förlita sig på att någon annan gör det.

The agile practices you have used for the current sprint

Now

Vid kursens start hade gruppen väldigt lite erfarenhet av att arbeta agilt med Scrum och fick därför göra mycket research kring hur man gör detta. Under projektets gång blev gruppen mer bekväm med Scrum-arbetssättet och dagliga avstämningar, reflektioner och ”agila verktyg” som hjälp blev mer som en rutin. En stor del av gruppens ”tolkning” av agila metoder och Scrum är att ha kommunikation som en grundpelare i vår process. Om någon missat ett möte har denne informerats av gruppen för att se till att alla ständigt varit uppdaterade kring vad som händer. Vi har mellan mötena kommunicerat i ett flertal gruppchattar med olika syften samt interagerat i gemensamma dokument för reflektioner. Ett betydelsefullt verktyg för vår agila process har varit Trello, vilken vi använt som en scrumboard där det tydligt visualiseras vad som behövs göras och vem som ansvarar för det. Trello har kontinuerligt använts av samtliga medlemmar och det har gjort att det alltid gått att vara ”up to date” även om det varit ett längre uppehåll mellan mötena (pga helger och högtider). Daily scrums har ägt rum samtliga dagar då teamet har träffats, vilket har skett 2-3 gånger i veckan. Det diskuterades huruvida det skulle ske även de dagar vi inte sågs, men det ansågs vara redundant då det huvudsakligen var dagarna vi träffades som vi arbetade på projektet. Våra daily scrums har fungerat som ett sätt att ge överblick kring vad som fungerat bra och vad som fungerat mindre bra, samt gett varje gruppmedlem möjlighet att be om hjälp om det har behövts.

När utvecklingsprocessen kom igång ordentligt, dvs när alla förutsättningar för att kunna börja koda fanns, beslutade gruppen att dela in oss i mindre grupper (parprogrammera) för att bli mer effektiva. Eftersom att det krävdes en del tid för att sätta sig in i en user story passade det väldigt bra med parprogrammering, då paren sinsemellan kunde ”gräva ner sig” och bli experter på just det den uppgift de ansvarade för och därefter vid behov informera eller utbilda resten av gruppen. Genom att arbeta på detta vis kunde flera områden behandlas samtidigt och utvecklingen av applikationen gick (förmodligen) snabbare än vad det hade gjort om vi arbetade som en enda stor grupp. Valet till just parprogrammering om två och två var att det var mer optimalt att programmera tillsammans jämfört med individuellt. Man får en truckfactor på två, vilket innebär att om någon skulle vara sjuk finns det fortfarande en som kan driva projektet. Fördelarna med parprogrammering märktes tydligt under det inledande strulet med utvecklingsmiljön, varpå beslutet att använda den typen av fördelning bekräftades av samtliga gruppmedlemmar. Den initiala uppdelningen har ansetts vara bra för samtliga gruppmedlemmar, varför programmeringsparen varit samma genom hela processen. En annan

anledning till att vi inte bytt par är att varje par varit väldigt insatta i vardera uppgift, därför blir det höga switching costs att byta par. Dock har det skett ett samarbete mellan grupper där det har behövts. Gruppen konstaterar att denna uppdelning varit mest optimal för projektet.

Sprint retrospective utfördes varje måndag vid slutet av genomförd sprint. Genom att ha standardiserat ramverk för mötesprotokoll med inkluderade punkter för sprint review och retrospective (se "yyyymmdd Mötesprotokoll - End Sprint X" i Google drive) säkerställdes det att dessa punkter genomfördes på ett strukturerat sätt under hela IT-projektet. Vid sprint retrospectives diskuterades KPI:erna och med utgångspunkt från dessa fördes en diskussion kring förbättringspotential för processen. Detsamma gällde sprint plannings, vilka också genomfördes på måndagar för att utgöra starten för nya sprintar. Under sprint plannings flyttades högt prioriterade user stories från product backlog till sprint backlog. Dessa behandlades sedan ytterligare, genom att dela in i tasks, estimera och definiera Definition of Done's.

Goal

För framtida projekt är ett mål att förbättra vår context switching capability - när ny information presenteras måste vi snabbt kunna pivotera och förhålla oss till nya krav och förväntningar. Syftet med målet är att kunna hantera de naturliga snabba ändringar som sker i agila projekt när produktägare/slutanvändare antingen förändrar målbilden, eller att gruppens insikt i vad den faktiska målbilden är fördjupas.

Gap

Se till att kommunikationen är tydlig och att alla medlemmar i teamet påminner varandra att reflektera över arbetssättet som används och ifall det skulle kunna optimeras på något sätt under processens gång. Resursinventering och en högre granularitet i hur arbetsfördelningen skall ske får effekten att det sociala kontraktet i högre grad representerar hur arbetet faktiskt genomförs i gruppen. Således får kontraktet också en högre legitimitet.

För att kunna förbättra vår context switching capability måste vi dels förbättra vår förmåga att snabbt processa ny information genom att direkt lyfta ny information via exempelvis daily scrums. Dessutom måste vi effektivt kunna ta beslut för att byta riktning, detta genom ett fokus på dialog med olika intressenter för att diskutera eventuella förändringar i scope

The time you have spent on the course (so keep track of your hours so you can describe the current situation)

Now

Under kursens gång har varje person antecknat sina tider i en loggbok. Där syns att teamet förhållit sig relativt väl till det mål om 20 timmar/vecka som sattes upp inledningsvis. Vissa veckor har det blivit färre timmar på grund av lediga och röda dagar, men då har gruppens timmar relativt antalet dagar

förhållit sig inom de ramar vi önskat. Totalt har vi legat strax under 20 h/vecka, vilket stundtals berott på att vi saknat tillräckligt med värdeadderande arbete för att uppnå 20 timmar. Under dessa perioder har vi anpassat oss och försökt bryta ned processen i ännu mindre delar för att få en överskådlig bild och därefter addera värdefulla saker att göra. Vi har som en av våra KPI:er mätt antalet timmar/vecka och fördelningen mellan mötestid och individuell tid, något som varit bra för processen då det bidragit till transparens i teamet samt kartlagt vad vi lagt tid på. Det har gjort att vi kunnat justera arbetssättet för att lägga tid på ”rätt saker” när det behövts, t ex effektivare möten för att kunna lägga mer tid på utveckling av appen.

Den tid vi lagt på projektet har bestått av i snitt 2 mötesdagar/vecka och individuellt (eller i par) arbete mellan mötesdagarna. De veckor innan vi kom igång med programmeringen blev antalet mötestimmar lite för många, men detta har jämnat ut sig under processens gång då kodandet flutit på. Att ha fasta tider har varit fördelaktigt för Scrum-metodiken, då vi kunnat ha en öppen kommunikation när vi suttit tillsammans och arbetat även om det skett i team. En lärdom från gästföreläsaren var att man helst ska utveckla applikationen tillsammans, vilket vi applicerat i utvecklingsprocessen.

Goal

Hålla sig till den överenskomna eller rekommenderade tiden ett projekt bör ta, eftersom att det då innebär att teamet kan göra stora framsteg på den utsatta tiden om den utnyttjas på rätt sätt. Det är också viktigt då projektets scope beror på tiden som projektmedlemmarna lägger ner. Till framtida projekt kan det även vara användbart att ha en riktlinjer kring vilken flexibilitet som är tillåten att jobba mer eller mindre än de överenskomna timmarna ifall dessa mäts på veckovis.

Gap

Fortsätta att föra loggbok för att skapa transparens kring den tid som läggs. Teamet bör också vara villiga att justera antalet måltimmar för varje vecka beroende på hur arbetsbelastningen i processen ser ut just då. Vissa steg i processen kan vara mer tidskrävande och därför kan det behövas adderas fler timmar, medan andra steg kan bli avklarade på kortare tid än den utsatta. Det är viktigt att sträva efter transparens och tydlig kommunikation eftersom att alla då kan hjälpa varandra att nå det gemensamma målet.

The sprint review (either in terms of outcome of the current week's exercise or meeting the product owner)

Now

Under den första veckan utgick vi ifrån den Lego-övning kursen inleddes med. Där identifierade vi att vi hade problem med att uppskatta rätt velocity och scope för de user stories vi tog oss an. Då bestämdes det att vi i den ”riktiga” processen skulle ”learn by doing” och införa metoder eller verktyg för att hjälpa oss på vägen, vilket vi bland annat gjorde genom våra KPI:er gällande tidsuppskattning och velocity.

I resten av kursen baserades våra sprint reviews på mötet med PO's, vilket skedde varje onsdag. Sprinten påbörjades och avslutades dock på måndagar, ett upplägg som gruppen ansåg lämpligt eftersom att det hade inneburit tidsbrist att göra en sprint review och ny sprint planning efter mötet med PO's på onsdagseftermiddagarna. Under onsdagsmötena försökte teamet att ha en demo redo att visa för PO's och på så vis få feedback på det som utvecklats under föregående sprint. Generellt var PO's nöjda med det vi åstadkommit och upplägget möjliggjorde att arbetet inte stannade upp under någon längre tid då det oftast tidigt i en sprint kunde uppstå frågor, både tekniska, och till PO, vilka vi tidigt i sprinten kunde få feedback på.

Goal

I framtida projekt skulle sprint review kunna förläggas till samma dag som mötet med PO (som då sammanfaller med start/slut av sprint). Detta i syfte att snabbt återkoppla till feedback från PO och på så sätt närmare koppla detta till senaste genomförda sprint. Se stycket under kopierat från KPI-delen:

> *För att lösa gapet med produktägarens nöjdhet skulle PO kunna betygsätta ändringen från en skala på 1-10 från tidigare vecka för att peka utvecklarna (oss) i rätt riktning. En nöjdhets-KPI från PO:s sida skulle förtydliga kommunikationen, få en utvärdering på produkten svart på vitt samt se till att arbetet faktiskt är värdeadderande för uppdragsgivaren.*

Gap

Arbetet med att flytta sprintens start/slut till en dag som sammanfaller med ett möte mot PO skulle underlättas i framtida projekt då den främsta anledningen till att det gjordes på detta sätt var att det fanns möjlighet till handledning på onsdagar i detta projekt. Effekten av en flytt av start/slut sprint till en dag som sammanfaller med möte av PO skulle vara en snabb återkoppling på genomfört arbete. Effekten av en kvantifierbar KPI är att enklare kunna se trend förändringar i PO's bedömning av produkten.

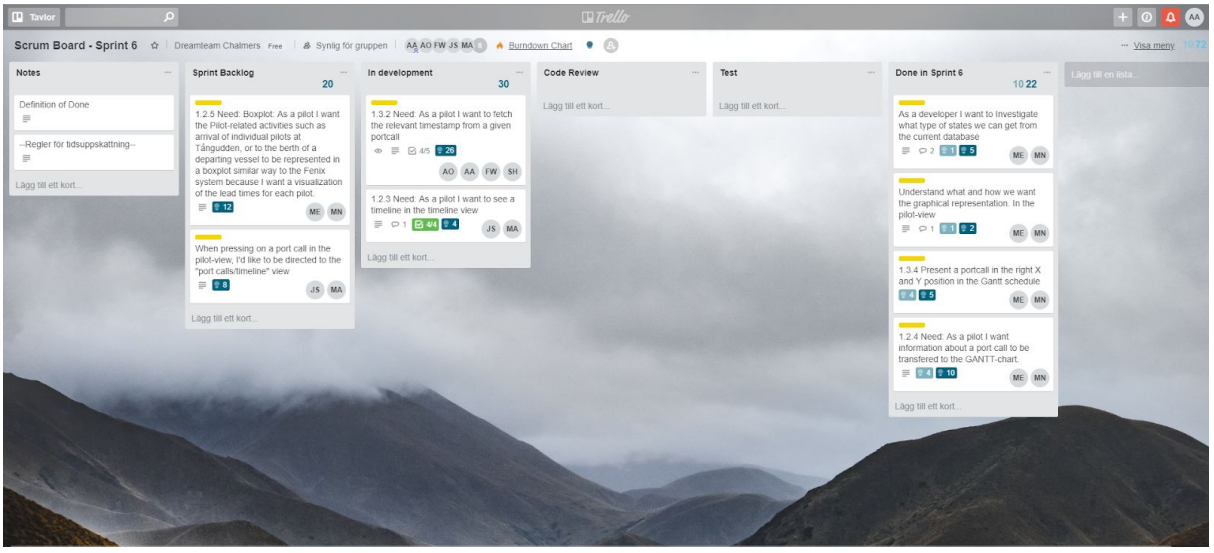
Best practices for using new tools and technologies (IDEs, version control, scrum boards etc.)

Now

Vi valde att använda Trello för att visualisera varje veckas sprint. Vi läste på om hur best-practice såg ut för Trello tillsammans med Scrum och insåg att det fanns flera olika möjligheter att strukturera upp våra boards, varvid vi testade olika typer av upplägg, utvärderade under sprint retrospective och valde det som passade gruppen bäst.

Trellostruktur:

- Det fanns en tavla som innehöll hela product-backlogen och sedan en tavla för varje sprint (Scrum Board)
- Varje kort i Scrum Board innehöll en checklista med varje tasks, en beskrivning om vad som krävs för att user storyn ska anses klar samt vilka medlemmar som ansvarar för user storyn.
- Gruppen implementerade Scrum for Trello vilket gav visuell hjälp om våra estimerade respektive faktiskt tid som spenderats på user storyn.



Figur 15: En skärmdump på vårt Trello Board för sprint 5.

För utvecklingsmiljön rörande Android Studio och React Native hade vi initialt stora problem med att få det fungerande hos alla medlemmar, speciellt för gruppmedlemmarna som använde Windows. Detta problemet hanterades genom mycket konsultation i gruppen, slack och på handledningstillfällena. Dessutom innehöll alltid varje parprogrammeringsteamen minst en Mac användare, vilket säkerställde att det alltid fanns möjlighet att kunna koda.

De flesta i teamet använde Sublime Text som editor, vilket var ett nytt verktyg för många. Det visade sig vara en smärtfri övergång från tidigare editors och IDE's som använts.

Gällande versionshantering började vi redan från första sprinten att använda GitHub, vilket initialt innebar ett del lärande då de flesta medlemmar inte använt Git/GitHub förut. Gruppen lärde sig successivt att använda GitHub, dels genom att testa sig fram såväl som genom att använda programmet GitHub Desktop vilket hade en lägre instegströskel än Git i kommandotolken. För testning använde vi verktyget DeepScan som beskrivs utförligare i "Code Quality"

Goal

I framtiden är en rimlig målsättning att kunna bemästra de olika verktygen som använts, dels att kunna känna sig ännu mer bekväm med användandet samt att utnyttja ännu fler specialdelar av verktygen för specifika ändamål. Exempelvis har både Trello och GitHub en hel del extra tillägg som kan vara värt att använda beroende på hur projektet visas, exempelvis finns det flera "Agile tools" tillgängliga som powerups i Trellos sortiment.

Gap

Ett angreppssätt som har fungerat bra under projektets gång, är att ta fram KPI:er som kräver att verktygen används för att kunna mätas (ex: Estimation Accuracy kräver lägga in tid i Trellon). Därför skulle introduktion/ändringar av KPI:er vara ett hjälpmedel för att öka användandet av verktygen. Utöver KPI:er hade en ordentlig genomgång av vilka tillägg som finns i verktygen varit användbart för att sedan kunna testa dessa under en sprint och under sprint retrospective kan gruppen avgöra om verktygen ska behållas.

Relation to literature and guest lectures (how do your reflections relate to what others have to say?)

Now

Det visar sig att vad som står i litteraturen och vad som sagts från föreläsningarna stämmer väl med verkligheten. Det vi framförallt har tagit inspiration ifrån är Michael från Trine:s upplevelser i IT-världen samt reflektionerna från "Scrum and XP from the trenches". Båda har poängterat vikten av att inte göra saker som inte är värdeadderande för processen och för produktägaren. Vi blev framförallt förvirrade efter gästföreläsningen från Michael, som verkligen poängterade vikten av att ständigt röra sig framåt och skapa värde, ofta med kort planeringshorisont. I kurser på Chalmers har det motsatta ofta lönat sig, dvs att planera och lära sig innan man implementerar. Vi brottades med detta skifte av mindset i flera veckor och det var troligtvis denna del av Scrum-metodiken som vi hade svårast att ta till oss. Mottot blev "Just do it!", vilket ofta återbesöktes när det fanns en stor tröskel att tackla problem.

Det är också många andra lärdomar som sades på föreläsningar och i litteraturen som är väl värda att plocka med sig. Ett tydligt exempel från en föreläsning, samt "breaking a feature into tasks" som är länkad på hemsidan, är att man ska ha vertikala user stories. Under föreläsningen tror man sig förstå vad som menas och hur man ska göra det. När man själv ska göra det inser man svårigheterna. Till en början när vi kodade fick många beroende user stories som inte alls var särskilt vertikala. Vi fick då gå tillbaka till litteraturen och försöka omsätta den praktiskt på våra faktiska problem. Därefter gick det betydligt mycket bättre att skapa user stories i Scrum-anda.

Det har också framgått att retrospective och review är väldigt viktigt för att få en bra process och produkt i Scrum. Det är någonting vi håller med om, då många förbättringsåtgärder har gjorts när vi har suttit och diskuterat sprint reviews.

Goal
För framtida projekt vill vi som grupp prioritera förstahandskunskaper från de som tillämpat teorier i praktiken. Det visade sig under detta projekt att dessa insikter var de som skänkte störst värde till processen. Effekten av att filtrera litteraturen genom yrkesutövares erfarenheter gör att vi snabbare kan syntetisera litteraturen på området.

En lärdom är att det är viktigt att vara kritisk till kunskapen, att den kan bero på fall till fall och vara kontextberoende. Innan man applicerar någonting som man läst/hört, är det därför viktigt att rannsaka huruvida kunskapen kan göra nytta i den givna kontexten man befinner sig i för stunden.

Gap
Det stora gapet när det gäller litteratur att är själv kunna sålla på ett stort urval av källor. Genom denna kurs har vi fått material tilldelat från extern part, och därför inte behövt orientera oss i litteratur-djungeln. För framtida arbeten kommer vi själva behöva finna relevant litteratur, någonting som bör uppmärksammas. För att lösa gapet till att få empiriska förstahandskunskaper kommer vi i framtida mjukvaruprojekt se till att få vägledning av mer seniora programmerare i användningen av Scrum (eller om det skulle vara någon annan agil arbetsmetod). Effekten av detta blir värdefull feedback och kunskap som vi kan applicera själva.

För att maximera inlärning och förståelse är det viktigt att få omsätta kunskaperna i praktiken. Det skulle vara bra att tillägna en user story med detta syfte, för att säkerställa att det faktiskt genomförs.

Gitinspector

```
Author                    Commits    Insertions      Deletions    % of changes
AndreasOpedal                   5           304            237           18.48
Martin Ahlberger                8           237             96           11.37
Mattias Eriksson               30           548            407           32.62
Max Nyman                       9           628            104           25.00
fendelin                        1            67              3            2.39
mahlberger                      6           160            101            8.91
sorsa                           4            17             19            1.23

Below are the number of rows from each author that have survived and are still intact in the current revision:

Author                    Rows      Stability          Age       % in comments
AndreasOpedal               61           20.1          0.9                0.00
Martin Ahlberger           218           92.0          0.1                0.00
Mattias Eriksson           222           40.5          0.3                6.31
Max Nyman                  471           75.0          0.1                1.27
sorsa                        3           17.6          0.7                0.00

The following responsibilities, by author, were found in the current revision of the repository (comments are excluded from the line count, if
possible):

AndreasOpedal is mostly responsible for:
    39 App/components/pilot-timeline-view/index.js
    20 App/components/side-menu-view/index.js
     2 App/navigators/appnavigator.js

Martin Ahlberger is mostly responsible for:
   218 App/components/pilot-timeline-view/index.js

Mattias Eriksson is mostly responsible for:
   208 App/components/pilot-timeline-view/index.js

Max Nyman is mostly responsible for:
   317 App/components/side-menu-view/index.js
    85 App/navigators/appnavigator.js
    63 App/components/pilot-timeline-view/index.js

sorsa is mostly responsible for:
     3 App/components/pilot-timeline-view/index.js
```

Figur 16: Visar varje individs bidrag till koden enligt Gitinspector.

Vår git var svår att automatiskt undersöka med Gitinspector. Vår uppfattning är att Gitinspector inte speglar individernas utvecklingsbidrag då vi suttit i par och utvecklat. Inte heller gruppernas bidrag verkar stämma helt överens. Svårighetsgraden på tasks har varierat till stor del och antal rader kod speglar inte helt prestationen som krävs för att skriva den. Det är möjligt att antalet insertion stämmer men då vi alla är ovana utvecklare i Javascript är det inte säkert att hög tillförsel av kod är bra.

Paren vi programmerat enligt har varit följande:

- Anton Albinsson och Filip Wendelin
- Andreas Opedal Eriksson och Staffan Hellsvik
- Jenny Sorsa och Martin Ahlberger
- Mattias Eriksson och Max Nyman

Utöver parprogrammeringen har vi arbetat i större grupper för att lösa vissa problem, varför vissa av paren enligt Gitinspector inte ser ut att ha bidragit lika mycket som andra. Som nämnt stämmer alltså inte Gitinspector överens med bidraget från de olika paren. Som exempel har utvecklingsparet Anton/Filip bidragit en lika stor del till gruppens prestation, men enligt Gitinspector har de endast ett bidrag på 2.39 %. Anledningen till detta skulle kunna vara att vi som grupp följt Scrum-riktlinjen att arbeta tätt tillsammans. Således har kod delats fritt genom att visa på sin egen skärm för en annan gruppmedlem, och det tidskrävande arbetet med problemlösning, som inte nödvändigtvis resulterar i stora avsnitt med code-commits, har också tagit mycket tid för de programmeringspar som tilldelats svåra user stories.

Då vi bara ändrat i några få filer har kommandot vi använde till Gitinspector en del som utesluter övriga filer:

```
gitinspector -x file:App/android/,App/assets/,App/cert/,App/config/,App/files/,App/ios/,App/node_modules/,App/react-native-pinch/,App/reducers/,App/util/,group\
docs/,app.json,jsconfig.json,keycloak-app-production.json,keycloak-app.json,keycloak.json,package-lock.json,package.json,README.md,andoid-icon.png,App.js,App/components/s
end-portcall-view/index.js,App/components/timeline-view/sections/operationview.js,App/components/portcall-list-view/sections/filterMenu.js,App/actions/portcallactions.js,App/com
ponents/login-view/index.js,App/components/loginkeycloak-view/index.js,App/components/vessel-lists-view/index.js,App/components/timeline-view/sections/statementview.js,App/
components/portcall-list-view/index.js,App/components/berth-timeline-view/index.js,App/actions/eventactions.js,App/components/settings-view/index.js,App/actions/berthactions.js,
App/components/about-view/index.js,App/components/timeline-view/index.js,App/components/berth-timeline-view/sections/EventDetails.js,App/components/berth-timeline-view/sec
tions/EventView.js,App/components/berth-timeline-view/sections/BerthSettings.js,App/actions/settingsactions.js,App/actions/vesselactions.js,App/components/portcall-list-view/,App
/components/berth-timeline-view/,App/components/timeline-view/,App/components/select-favorite-state-view/,App/components/top-header-view/,App/components/send-portcall-vie
w/,App/components/berth-list-view/,App/actions/,App/components/vessel-info-view/,App/components/login-view/,App/components/state-list-view/,App/components/error-view/,App
/components/mini-header-view/index.js,App/components/mini-header-view/,App/App.test.js -r
```

Slutpresentation

Projektet avslutades med en slutpresentation av det färdiga resultatet, då vi fick möta både PO’s och andra aktörer i kedjan. Syftet med slutpresentationen var att visa upp det som skapats under processens gång, samt förklara eventuella framtida implementationer för PO’s och andra intressenter. Gruppen fick positiv feedback från både PO’s, examinatorer och lots vilket antyder att vi gjort framsteg och uppnått stora delar av det önskade resultatet.

Kommentarerna vi fick från lots var:

> *"Det är en grafisk representation av schemaläggningen som liknar den programvara vi nu använder. Färgkodningen på fartygen motsvarar inte exakt de tillstånd vi har i dagsläget. Vita fartyg t ex betyder något annat i vårt system men det kanske är enkelt att ändra? Om vi får en ändring i ett bekräftat event i dagsläget byter det färg och hoppar till ett annat ställe i vyn."*

Vad skulle du vilja se för utveckling i framtiden utifrån den app vi presenterat idag?

> *"Jag skulle önska att arbetsordningen när vi lägger in eller planerar fartygen går till på samma sätt som vi arbetar idag. Idag får vi oftast en "ETA to Berth" som vi sedan arbetar oss bakåt ifrån för att planera in fartyget. Om vi på något sätt kan arbeta med det på ett mer automatiserat sätt hade det varit bra. Jag skulle också önska ett sätt att kommunicera, t ex i form av chatten som vi pratat om, med de andra aktörerna i appen så att man slipper ringa varandra när det är oklarheter så som vi gör idag."*

Summerande reflektioner kring projektet som helhet

Now
Vid projektets inledning utgjorde detta en uppgift som på flera sätt var den första av sitt slag. Att ta en kravspecifikation från PO/slutanvändare och omsätta detta till en lösning har varit en lärorik utmaning där olika intressenters behov har balanserats mot varandra. Genom löpande avstämningar mot slutanvändare och sedan produktägare har vi i gruppen tagit till oss värdet av att säkerställa att det arbete som utförs tillför värde mot produktägaren och ligger i linje med dennes önskningar samtidigt som det skall omfatta vad slutanvändaren förväntar få till sig. Att hantera dessa förväntningar, och se till att kommunicera mot dessa intressenter vad som är möjligt och inte, har varit mycket lärorikt.

En lärdom har varit att det är viktigt att vara kritisk till kunskap som presenteras i kurslitteraturen, att den kan bero på fall till fall och vara kontextberoende. Innan man applicerar någonting som man läst/hört, är det därför viktigt att rannsaka huruvida kunskapen kan göra nytta i den givna kontexten man befinner sig i för stunden.

Vi har i gruppen löpande utvecklat egna user stories för att identifiera problem som vi som utvecklare behöver överkomma för att kunna leverera värde mot produktägaren.

Genom våra tidigare kurser i Java har vi kunnat applicera framför allt en vana att gå igenom kodavsnitt och identifiera vad som är viktigt, samt olika tekniker för att inhämta information eller kunskap som vi som grupp ännu inte besitter, t ex stackoverflow eller applicerandet av programmeringstekniker.

Vi har också tagit till oss nya verktyg för att arbeta som ett utvecklingsteam, såsom Trello för arbetet med våra user stories och backlog, Github för vår versionshantering och uppdelning av arbetet med koden.

Framför allt har den största nyttan tillkommit genom att vi lärt oss tillämpa Scrumprocessen som arbetssätt vid mjukvaruutveckling. Tidigt i projektet bestod utmaningar i att få utvecklingsmiljön att fungera vilket också medförde att de tidiga sprintarna kretsade kring denna problematik. Efter de inledande sprintarna blev dock syftet med de olika delarna av Scrum tydligare.

När gruppen nådde den fas där samtliga medlemmar, arbetandes i par, var engagerade i olika user stories blev det tydligt hur Scrum på ett agilt sätt främjar informationsdelning, att gruppen strävar mot samma mål, och att arbetet prioriteras på det sätt som PO önskar. På grund av de olika gruppernas sammankopplade roller i hamnen fick även Scrum of Scrums ett tydligare syfte för gruppen då det ibland uppstod behov att ha informationsdelning även över grupperna, t ex gällande olika states, eller eventuella behov från andra grupper på den egna gruppen.

Goal
Att i ett framtida IT-projekt, nu med större mognad och erfarenhet inför utmaningarna i att utveckla agilt, få arbeta med Scrum för att få mer erfarenhet av de olika delar av processen som denna kurs varit en inledande orientering till. Syftet är att känna sig bekväm i, och på ett professionellt sätt kunna utföra, de olika roller som vi under kursen kommit i kontakt med, såsom Scrum master, produktägare och utvecklare.

I avseendet Scrum of Scrums upplevs detta arbetssätt som mycket överförbart på andra typer av projekt än mjukvaruutvecklings-mässiga sådana och det hade varit intressant att använda sig av detta arbetssätt i en annan kontext. Främst är det den effektiva informationsdelnings-aspekten som upplevs värdeskapande.

Gap
Att få mer erfarenhet av att: På olika sätt testa den kod som presterats. Att leda en grupp i rollen som Scrum master. Att agera produktägare i ett utvecklingsprojekt. Att få ytterligare erfarenhet av att arbeta som utvecklare i ett IT-projekt. Den sammanlagda effekten av att tillgodogöra sig dessa erfarenheter skulle uppfylla målet ovan, i att vara en väl bevandrad “Scrummare” och utvecklare.