DAT255: Software Engineering Project

Slutreflektion

En utvärdering av gruppens prestation i ett Software Engineering-projekt

LEVIN AHLSELL
RIKARD ERIKSSON
CLARA HOLLÄNDER
STINA STRÖBY
ALICE ÖSTBERG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Software Engineering Project
Unicorn
Chalmers University of Technology
Gothenburg, Sweden 2018

Contents

1
Introduktion

I detta kapitel ges en övergripande introduktion till vad gruppen haft för ambitioner med projektet, vad dessa ambitioner faktiskt resulterade i, samt vilka lärdomar vi som grupp kan ta med oss i framtida projekt.

1.1 Scope

Efter att uppgiften blev tillkännagiven var det tydligt att värde skulle skapas primärt för de som arbetade med förtöjning. För att förtydliga deras behov tog vi tidigt kontakt med AB Klippans Båtmansstation som arbetar med en generisk version av Portable CDM-appen och är ansvariga för förtöjningen i Göteborgs hamn.

De som arbetade med förtöjning och appen hos AB Klippans Båtmansstation hade många förbättringsförslag. Bland annat synliggjordes det att många funktioner saknades och för att komplettera appen behövde flertalet andra externa tjänster nyttjas, bland annat en gratis webb-tjänst med gps-funktion samt tillgång till lotsarnas system. Ultimat ansåg de att Portable CDM hade använts för all dessa funktioner men att det fanns för många brister i applikationen.

Viktiga funktioner som identifierades saknas (utan inbördes prioritering):

1. En GPS-funktion där alla fartyg runt Göteborgs hamn kunde överblickas.
2. Fullständig information om alla fartyg. För varje relevant rad som inte fyllts i under ”Vessel Info” i appen behöver förtöjarna hitta informationen på annat vis (via webb-sidor, register etc.)
3. Det fann ingen funktion för att kunna göra anteckningar i appen.
4. En push-notis som förvarnade om ett specifikt fartyg närmade sig hamnen.

Alla dessa funktioner ansåg vi efter första tillfället skulle vara möjliga att genomföra samt att det skulle skapa stort värde för förtöjarna. Därför motiverades det att det valda scopet inte skulle frångå eller utökas utöver dessa fyra viktiga funktioner.

User stories skapades efter dessa funktioner och inbördes prioritering valdes utifrån estimerad tidsåtgång och därmed svårighetsgrad i kombination med värdeskapande för Klippans Båtmansstaton. Att lägga till en rad om tonnage under Vessel Info i appen ansågs vara den enklaste user storyn och prioriterades därför först som en “mjukstart”. Ambitionen var även att kunna koppla denna rad till databaser och au-

tomatiskt kunna hämta ner denna typ av information om den skulle saknas. Dock visade det sig att det inte fanns tillgång till databasen och vi nöjde oss med att manuellt lägga in värde för tonnage hos 10 stycken fartyg. Dock förbereddes appen för att dels kunna lägga in information om tonnage vid registrering av nya fartyg samt att kunna kopplas samman till en databas i framtiden.

Den andra user storyn blev således att utveckla ett kommentarsfält för varje fartyg. Detta visade sig dock mer avancerat än först beräknat och scopet fick avgränsas från det att skapa ett kommentarsfält med global lagring till att innefatta ett med lokal lagring (alltså på respektive enhet och inte enheter sinsemellan). Vi hittade dock ändå ett sätt att skapa värde för samtliga parter. Genom ett kommentarsfält under vesselinfo skapar vi värde på följande sätt; Klippans båtmän får en plats i appen där dom kan skriva ner vad dom behöver veta till nästa gång båten anländer på ett intuitivt sätt. Information som önskas är av skiftande karaktär, men b.la. nämndes typ av tossar, tonnage och telefonnummer till kapten.

En del av den här informationen går att hitta online som privatperson, men företag är tvungna att köpa datan. Det visade sig att köp av data från sådana databaser var mycket dyrt och ett inköp av den storleken skulle behöva övervägas noga. Genom att skapa ett kommentarsfält som på sikt kommer kunna lagra och spara kommentarer tvärs enheter får även produktägaren se vilken information som är viktigt för förtöjarna, eller andra grupper som använder kommentarsfunktionen då de kan se vilken typ av data som ofta antecknas. Detta ger produktägaren två val; antingen överlåts informationssamlingen till intresserade slutanvändare som bygger en databas med hjälp av kommentarsfältet och på så sätt skapar en värdefull datasamling genom co-creation. Eller så visar det sig att informationen är så pass efterfrågad att det kan vara väl investerade pengar i att köpa in informationen från en externt part.

För att vidareutveckla funktionen till en global lagring behöver en databas skapas vilket inte bör vara alltför komplicerat för en kunnig i SQL. Oavsett resoneras denna funktion kunna generera god överblick över användarnas önskemål för information om respektive fartyg. Märks det till exempel att merparten kommenterar och lägger till information om tonnage kan det ge en indikation om att värde finns att prenumerera på en databas med givet tonnage för respektive fartyg.

Även user storyn om push-notiser påbörjades men slopades på grund av tidsbrist. Att skapa en notis sågs inte som något tekniskt avancerat, utan snarare låg svårigheten i att få notisen att lämpligt interagera med appen. Det skulle troligen inte heller ge samma värde i förhållande till nedlagd tid som ett kommentarsfält skulle göra för förtöjarna.

Sist prioriterades GPS-funktionen. Tekniskt sett planerades detta att det skulle genomföras genom ett API till en hemsida - marinetraffic.com. Men då detta ansågs för avancerat och tidskrävande gavs denna funktion lägst prioritet. Det resonerades även att förtöjarna redan nyttjade andra GPS-tjänster i sitt dagliga arbete och att de var vana vid att nyttja andra program för GPS-funktioner. Vid diskussion med

AB Klippans Båtmansstation var denna funktion ultimat en del av appen men att den inte var av största prioritet.

Till framtiden
Vad som har varit värdefullt i detta, och vad som kan tas med till framtida projekt, är integrationen av slutanvändaren i ett så pass tidigt stadie. Speciellt när man använder en metodik likt Scrum där stor vikt läggs just vid att prioritera uppgifter. Att inte involvera slutanvändaren riskerar att projektets riktning gravt felprioriteras och att onödiga resurser tas i anspråk, främst i form av slösad tid. Den insikten det har gett att faktiskt ha träffat AB Klippans Båtmansstation har varit ovärderlig för projektets utformning och avgränsningar, likväl som prioriteringar.

En annan mycket positiv aspekt av att träffa slutanvändaren har varit att det har bidragit med ökad motivation då det åsynliggjort vem värdet faktiskt skapats för. Deras motivation för och inblick i ämnet har smittat av sig och genomsyrat projektet, något som säkerligen är aktuellt för merparten av projekt med en fristående slutanvändare.

En mer konkret aspekt som kommer tas med i framtida projekt är värdet av att faktiskt undersöka komplexiteten av en uppgift innan ett tidsestimat sätts. Vi har haft en tendens att underestimera tidsåtgång och svårighetsgrad på merparten av genomförda user stories. Detta har gjort att vi inte avgränsat vårt scope till den grad som den slutgiltiga produkten faktiskt innefattar och gett en intern känsla av att vi arbetat långsamt samt att produkten har färre funktioner än vad som presenterats för P.O. Dock är vi fortfarande ödmjuka inför faktumet att det är oerhört svårt att förutse hur komplext ett problem är eller hur lång tid det faktiskt kommer ta. Poängen är snarare att vi borde efterforskat mer om innebörden av respektive uppgift innan tidsestimat satts för att undvika fundamentala felbedömningar.

Dessutom tror vi att en lärdom vi kommer ta med oss är värdet av att diskutera och bolla idéer med P.O. som med sin tekniska bakgrund kan komma med värdefull insikt i vad som är möjligt och vart tekniska avgränsningar bör göras.

Även utifrån Scrums metodik med att dela in och prioritera user stories i ett tidigt skede kan projektets omfång hyfsat snabbt fastställas och snabbt återkopplas till slutanvändaren för att snabbt förtydliga om projektet rör sig i rätt riktning. Dessutom kan det skapade värdet dessutom snabbt fångas upp av den som värdet är tänkt att tillföras till.

2
Metod

I följande kapitel beskrivs den metod som utnyttjats i projektet samt vilka lärdomar som kan tas med i framtida projekt. Först beskrivs det social contract som togs fram i projektets början, därefter beskrivs de verktyg vi utnyttjat samt vilka roller som tilldelats. Vi förklarar även hur vi har validerat appens beteende under utvecklingen.

2.1 Social Contract

Under den första sprinten skrevs ett social contract, alltså ett ”socialt kontrakt”, i vilket det sattes upp riktlinjer för hur vi skulle arbeta agilt under de olika sprinterna för att kunna nå vårt gemensamma mål. Syftet med projektet klargjordes även så att hela teamet var medvetna och kunde jobba efter det.

Under den första sprinten var vi inte helt insatta i vad projektet gick ut på, så vissa delar av det sociala kontraktet kom att utvecklas och förändras under senare sprintar. Under den sjätte sprinten genomförde vi därmed en förändring gällande ett av de olika KPI:erna, ”Quantity of Code”, till ”Perceived Success” och att alltså mäta den upplevda individuella prestationen istället för att mäta mängden producerad kod. Förändringen gjordes på grund av att kursens fokus till största del inte låg på kodningen, utan på det agila arbetet kring kodandet – inkluderande allt mellan att sköta kommunikation med Product Owner till att skapa user stories och lära sig ny programvara. För vidare information, se avsnittet KPI:er.

Under den andra sprinten tilldelades roller till gruppens medlemmar för att kunna strukturera arbetet bättre och på så sätt arbeta mer effektivt som grupp vilket beskrivs utförligare under kapitlet Roller.

Sprinterna definierades även till att påbörjas på torsdag morgon och till att avslutas under onsdagskvällen under respektive vecka.

Det sociala kontraktet byggdes upp utifrån följande punkter:

- Desired outcome
- Success criterias
- Three KPI:s for monitoring our progress
- Roles
- Empowerment and Self organisation

- Platform - How do we communicate with each other
- Meetings

Hela gruppen var med vid upprättande av kontraktet och var därmed införstådd i det. Under veckornas gång förhöll vi oss i största möjliga mån till det sociala kontraktet, men på grund av koordingeringsproblem, som en följd av att tid behövde läggas på andra projektkurser eller exempelvis jobb, fanns under vissa sprintar svårigheter att samla hela gruppen för möten och samarbete.

Bristen på möten i helgrupp ställde därmed högre krav på kommunikation och uppdatering i back-logen för det, vid individuellt arbete, inte skulle finnas några oklarheter om vad som skulle göras. Till en början var kommunikationen bristfällig, vilket resulterade dubbelarbete och tveksamheter till vad som var påbörjat och inte och det tog därmed längre tid att slutföra vissa user stories eller tasks än vad som från början hade varit nödvändigt. Allteftersom gruppen blev mer van och bekväm med att arbeta med de olika kommunikationsverktygen som fanns att tillgå blev också processen mer tydligt upplagd – tasksen gjordes mindre, delegerades och tidsåtgång estimerades på respektive task. På så vis kunde ett mer önskvärt arbetssätt, i enlighet med vad som beskrivits i vårt social contract, upprätthållas.

Genom teletype, ett verktyg i Atom som möjliggör par/gruppkodning, gick det att kringgå det faktum att vi ofta befann oss på olika platser. Genom att undvika ställtider kunde fler effektiva timmar med samarbete erhållas.

Till framtiden
Utifrån vårt eget social contract har vi lärt oss att tydlighet och struktur är A och O. Även att tidigt i processen bryta ner user stories i små tasks och att estimera tidsåtgången över dessa för att senare kunna delegera ut uppgifterna till gruppmedlemmar underlättar processen. Det är också viktigt att uppdatera backlogen kontinuerligt för att gruppen ska kunna följa arbetet och hålla sig uppdaterade. Inplanering av stående veckomöten som alla är medvetna om och prioriterar och planerar sin tid utefter är också något som till framtida projekt bör tydliggöras.

Vi kommer även ta med oss att det är mycket värdefullt att skriva ett social contract tillsammans innan man går in i större projekt då detta förtydligar alla medlemmars ståndpunkt samt vad som förväntas. Det ger även grund till vad som bör göras om medlemmar missköter sig, vad ens acceptanskriterier är, gruppens roller etc.

2.2 Utnyttjade verktyg

Nedan beskrivs de verktyg som har använts under projektet samt hur vi har valt att använda dem. De flesta av verktygen var helt nya för oss och vi har lärt oss en hel del på att nyttja dessa nya teknologier. Det förs även resonemang om hur våra lärdomar kan användas i framtida projekt.

Git/GitHub/GitHub Desktop
För versionshantering har vi använt oss av Git, där vi laddat upp och sammanställt vårt projekt i GitHub. Via GitHub har merparten av all filhantering skett, oavsett om det handlat om kod eller om andra typer av filer. Respektive användare har dessutom använt sig av GitHub Desktop för att enklare kunna jobba med branches och commits från respektive dator. Detta har även möjliggjort att vi som användare kan committa filer tillsammans om de har parprogrammerats vilket ger ett bättre återspeglat repo.

Vår versionshantering har dock inte varit helt oproblematisk. Det tog tämligen lång tid att förstå hur vi som grupp skulle jobba med just Git, särskilt med tanke på att det tog lång tid att få igång den givna appen. Dessutom började vi alla med att individuellt forka repot, vilket vi snabbt insåg skulle bli problematiskt då vi behöver jobba i samma repo för att kunna bearbeta olika branches och få alla aktuella uppdateringar.

En felaktig .gitignore-fil skrevs dessutom som gjordes att över 3000 filer mergades vid varje commit. Detta gjorde det i princip omöjligt att följa vad som faktiskt skedde vid varje commit och försvårade all eventuell backtracking.

Vi som grupp har dock lärt oss mycket om att arbeta med Git och versionshantering, främst med tanke på icke-existerande förkunskaper samt att det känns som att vi gjort alla fel man kan tänkas göra. Oavsett så tycker vi att Git är en mycket bra metod som verkligen tydliggör tillsammans med GitHub om vem som gjort vad, speciellt om man arbetar i ett större team.

Atom/Teletype
För att skriva kod har vi använt oss av Atom som är en gratis programvara för textredigering och som utvecklats av samma skapare som GitHub, därför med bra interoperabilitet mellan Atom och Github. Ytterligare argument att vi valde Atom framför till exempel Sublime Text är snabbare sammanslagning med GitHub samt tillägget Teletype. Just Teletype används för att flera användare av ett visst repo kan i realtid jobba i samma branch utan att koden faktiskt har commitats upp till repot. I realtid kan man alltså se ändringar i koden som den andra användaren av Atom och Teletype utför.

Atom och Teletype har fungerat mycket bra vid kodning och Teletype har varit ett bra verktyg för att effektivisera parkodningen. Hela gruppen är därmed mycket nöjda med att använda Atom för att skriva JavaScript och skulle kunna tänka sig att använda programmet i framtiden Vi har även varit mycket nöjda med versionshantering via GitHub och kan tänka oss att använda detta i framtida projekt, oavsett val av programmeringsspråk.

Node JS/Expo
För att kunna köra appen så som den varit skapad har vi behövt använda oss av Node JS och Expo. De flesta i gruppen har haft problem med att få igång samt att

köra Expo, dock har inte det inte funnits något alternativ då appen skapats för att använda Expo. Dock har vi inte varit ensamma med att ha problem med Expo utan vi har förstått att det har varit så för merparten av de andra grupperna.

Android Studio/Emulator
För att köra appen använde vi oss av Android Studio och en tillhörande Emulator. Då ingen i gruppen ägde en Android-telefon användes emulatorn för att testa appens funktion. Det har dock fungerat mycket bra, när den till slut fungerade, då alla användare på egen hand kan testa sin nuvarande kod utan tillgång till någon annans hårdvara.

Att få igång Android Studio och Emulatorn var dock inte helt enkelt. Flertalet sprinter spenderades just med detta samt att undersöka fel i den givna appens koden. Flertalet emulatorer fick även skapas innan de faktiskt fungerade samt att appen kunde köras på dem.

I framtiden ser vi definitivt poängen med använda en emulator framför en faktisk hårdvara då detta gör det betydligt smidigare att testa funktionen. Just Android Studios har vi inte utnyttjat speciellt mycket mer än att förmedla koden till emulatorn och kan inte uttala oss så mycket om. Men när allting väl funkar kan vi ändå uppskatta programmets funktion.

Slack/Trello
För att kommunicera i projektet har vi dels använt oss av en slack-kanal samt av Trello. Via Slack har all kontinuerlig info gått samt att kursinfo även förmedlats här liksom möjligheten att fråga resterande kursmedlemmar om olika problem. Via Trello har vi haft vår Scrum-board, se avsnittet Scrum-board under kapitlet om Scrum för mer detaljer om hur vi utnyttjat denna. Generellt sett har vår Scrum-board samlat alla user stories samt vår sprint och product backlog.

Överlag har kommunikationen fungerat bra i gruppen. Dock var vår Scrum-board tämligen ostrukturerad i början då den var full med uppgifter utan intern struktur. Efter några sprinter fick vi dock ordning på den genom att ta fram relevanta user stories efter förtöjarnas önskemål och vi delade upp dessa i tasks efter färgkodning. Vi gjorde även relevanta tidsestimat till varje task och user story som inför varje sprint flyttades till listan ”In Progress” om den skulle behandlas.

Slack var inget nytt verktyg för merparten av oss men alla är överens om att Slack är mycket smidigt i projekt med många medlemmar då olika trådar kan anpassas så att de enbart visas för berörda. Dessutom blir det enklare att strukturera upp informationsflöden om de sker i olika kanaler. Trello var dock nytt för merparten av gruppens medlemmar och vi såg alla mervärdet i att utnyttja verktyget för att skapa en strukturerad Scrum-board. För alla projekt som är lite mer avancerade än att de kan struktureras upp med en simpel “Att göra”-lista kan Trello vara ett användbart verktyg.

Lynda.com/YouTube/Annat
I början av projektet krävdes mycket ny kunskap. Då alla i gruppen i princip började från noll när det gällde git, apputveckling, Javascript, React Native etc. Bland annat användes Lynda.com och YouTube för att gå igenom flertalet video-tutorials om olika ämnen. Dessutom lästes olika artiklar om olika tekniker och funktioner.

Vad vi kommer ta med oss i framtiden är att det finns hur mycket information som helst att tillgå på Internet, framför allt när det handlar om tekniska kunskaper så som allt relaterat till apputveckling. Det gör att man inte nödvändigtvis behöver betala eller läsa några akademiska kurser för att få relevant och aktuell kunskap inom IT och andra agila verktyg.

Terminalen
För att styra alla program och funktioner har vi använt oss av terminalen. Dock har vi inte varit helt konsekventa då vi valt att använda ett GUI för Git i form av GitHub Desktop. Överlag har terminalen gett oss djupare förståelse hur programmen samspelar samt att vi får direkt status av olika aktiva program.

2.3 Roller

Inom projektet har vi använt Scrums fördefinierade roller: Scrum-master, Product Owner och Development team. Rollerna tilldelades första veckan och har bestått under hela projektets gång. Med tanke på att projektet drevs av fem oerfarna studenter inom mjukvaruutveckling har alla medlemmar hjälpts åt under projektets gång med diverse uppgifter.

En representant från gruppen tog ansvaret att vara Scrum-master. Scrum-master har haft kontakten med de övriga grupperna samt representerat gruppen vid Scrum of Scrums. En representant från gruppen tog ansvaret att vara Product Owner, som framöver kommer förkortas som PO. PO ansvarade för att förstå och informera om det arbete förtöjarna samt PortCdm frågat efter. PO har medfört en tydlig kommunikation mot extern, ansvarat för informationsflödet och försett resterande gruppmedlemmar med information. I detta projekt ingick alla medlemmar i development team eftersom att alla arbetade med utvecklingen av produkten.

Rollfördelningen har fungerat till viss del men det finns en hel del lärdomar att ta med sig till framtida projekt.

”Inte tydlig uppdelning i vem som gör vad eller förväntad deadline. Alla ’tasks’ finns i trello men distributionen har varit näst intill osynlig. Detta måste förbättras till nästa sprint.”

– Team reflection 7

Det är viktigt att definiera vad rollerna innebär i ett tidigt skede. Vår okunskap när det kommer till att jobba enligt Scrum samt erfarenheter av IT-management fanns inte vid projektets start vilket har påverkat utfallet. För framtiden och mer

omfattande projekt under längre tidsspann har vi en del lärdomar att beakta när det kommer till roller.

Till framtiden
Bra vore att särskilja tydligt på ansvarsområden när det kommer till Scrum-master och PO så att det blir en god balans sinsemellan. PO kan ta mer utrymme och ansvara ytterligare för den riktning projektgruppen tar. PO får gärna tydliggöra målet och milstolparna på resan och sedan ligga på projektgruppen med tydliga deadlines så att utrymme för feedback från intressenterna möjliggörs. PO hade önskvärt även fått jobba ytterligare med att se till att user stories går i linje med MVP och är utformade med rätt detaljnivå så att när en user story har högsta prioritet är den redo för projektgruppen att påbörja och bygga.

Scrum-master kan ha ytterligare fokus på processen och projektgruppen. En stor del av Scrum-master-rollen är att försäkra sig om att ha en bra process genom att motverka distraktioner, exempelvis att personer jobbar osynkat, och hålla laget borta från att ta på sig för mycket, exempelvis ha för stort scope på en user story.

Vid två motsatta styrkor som drar i motsatta riktningar kan projektgruppen hitta en balans. Vi vill tänka på Scrum-master och PO som de två motsatta krafter som håller projektgruppen i balans. När det kommer till development team tar alla med sig lärdomen att jobba med en sprint backlog och att det är viktigt med kommunikation och eget ansvar. Att man redovisar vad man har gjort, vad man ska göra och hur man tar sig dit, samt delar med sig av lärdomar. Med det sagt, träning ger färdighet.

2.4 Validering av appens beteende

Under de tidiga sprintarna låg fokus på att få igång programvara, förstå denna samt på att förstå hur den skulle appliceras. Detta främst då då gruppen inte hade någon erfarenhet av apputveckling eller de program som skulle användas. Det var först under sprint fem som alla i gruppen hade fått igång appen och det första som då gjordes var att kartlägga den nuvarande appen, se Appendix 1 samt att göra en skärminspelning på appens ursprungliga funktiner.

När appen väl var igång var nästa steg att börja utveckla och implementera funktioner. I och med att gruppen tidigare aldrig hade gjort något liknande tillämpades Trial-and-Error-metodik för detta. En kodsekvens lades alltså till en början in där den troddes passa in. Därefter utfördes testning av koden med hjälp av emulatorn och om eventuella brister åskådliggjordes utfördes förändringar i kodsekvensen, som därefter fick genomgå samma process igen tills önksad effekt var tillhandahållen. Denna iterativa process besrkivs i figur 2.1.

Testningsprocessen inkluderade även ett moment bestående av att stämma av utvecklad funktion med PO och förtöjarna med hjälp av en mockup-film, vilken beskrev

funktionen i utvecklingsfas. Detta medförde att de olika intressentera fick en chans att ge feedback och även säga till om de önskade något som inte låg i linje med den tänkta funktionen. Den feedback som mottogs för kommentarsfälts-mockupen fick dock endast positiv feedback och därmed sågs det som ett tecken på att vi skapade värde för förtöjarna och produktägarna.

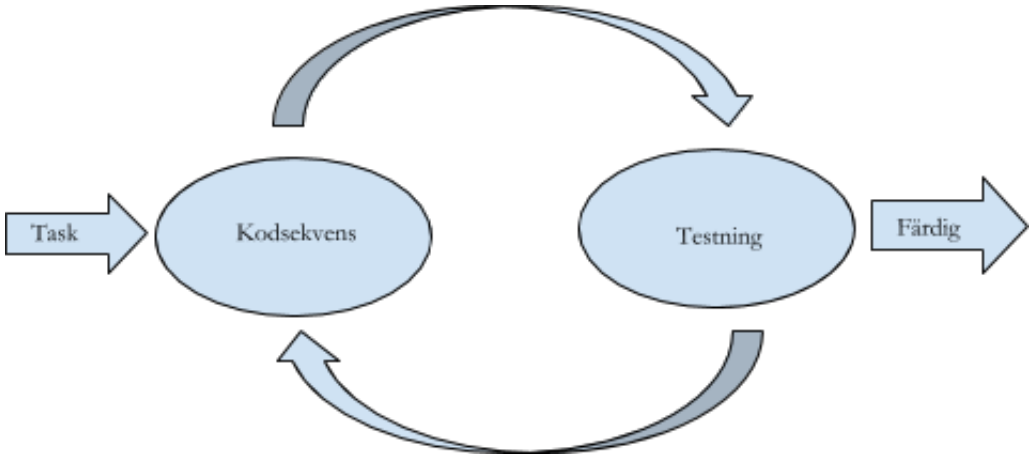

Figure 2.1: Arbetsprocess med trial-and-error-metodik.

Koden testades även i Deep Scan, vilket kan läsas ytterligare om i avsnittet Kodkvalité.

Till framtiden

Att ha en kontinuerlig kommunikation med intressenterna för appen, alltså de man skapar värde för, är något som kommer att läggas stor vikt vid i framtida projekt. Att utveckla funktioner med Trial-and-Error-metodik öppnar upp för stor inlärningsmöjlighet som en följd av att man “tvingas” söka efter information då ett fel, ett ”error”, uppstår. Därmed fås en konstant växande inlärning, eftersom man alltid kommer att stöta på saker som man inte kan lösa direkt och därmed behöver hitta ytterligare information om.

3
Agilitet och Scrum

Nedan följer en beskrivning av de agila verktyg som har använts. Först presenteras hur verktyget har använts under projektets gång och därefter presenteras hur vi tar med oss verktyget i framtida projekt. Då projektets struktur baserats på Scrum kommer merparten av de agila verktygen vara relaterade till just Scrum.

3.1 User stories

Utifrån projektets utformning skapades user stories för att fånga in önskade funktioner. De formulerades utifrån användarnas, alltså förtöjarnas, perspektiv och innefattar främst en värdeskapande funktion (alltså ingen specifik teknisk lösning). Utifrån dessa user stories skapades underliggande tasks, alltså delades problemet upp till mindre delar som sedan kunde delas upp sinsemellan i gruppen. För varje task gjordes ett tidsestimat, detta ledde även till att varje user story fick ett, utifrån tasksen, totalt tidsestimat.

User stories skapades först efter att vi träffat förtöjarna hos AB Klippans Båtmansstation när deras önskade värde kunde formuleras och utifrån dem har det tekniska arbetet fortskridit dock med en del modifieringar gällande tasks, prioriteringar och tidsestimat.

I framtiden kommer vi att ta med oss hur pass lättöverskådlig en user story faktiskt är och hur man på ett väldigt enkelt sätt kan fokusera enbart på det som skapar värde för slutanvändare. Dessutom att fokuset på en user story blir en funktion istället för enbart skapa någonting tekniskt och hur mycket mer värde det faktiskt genererar för slutanvändaren (vertikal istället för horisontell tårtbit). Så även om man jobbar i ett projekt som inte tillämpar Scrum kan vi alla se att det finnas värde i att skissa på user stories för att tidigt fånga upp önskat resultat från slutanvändaren.

3.2 Scrum-board

I projektet har vi använt oss av en Scrum-board via programmet Trello. Scrumboardens utformning visas i figur 3.1. På vår Scrum-board har samtliga user stories och uppgifter samlats där deras status snabbt kan överblickas genom att flytta vardera objekt till rätt status. Följande indelning av status är gjord efter föreläsarnas rekommendation utav Stories, To Do, In progress, Testing och Done. Dessutom har respektive task möjligheten att tilldelas en person samt att kunna kommenteras

och modifieras av alla användare.

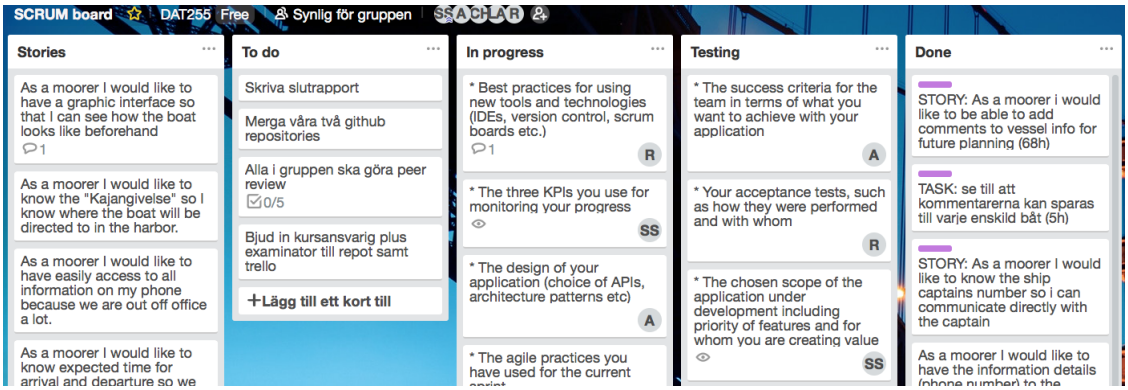

Figure 3.1: Scrum-board som visar gruppens arbetsprocess.

I ovan skärmbild är vår Trello i slutskeded och just därför finns en hel del objekt som inte är indelad i STORY eller TASK samt att de saknar tidsestimat. Observeras dock objektet som befinner sig i Done kan tydligare överblick över projektets struktur ges. Även färgkodning har nyttjats för att förtydliga i vilken user story respektive task tillhör.

Scrum-boarden har gett en mycket bra överblick över projektet och visat vad som bör göras samt utav vem. Samtliga i gruppen känner att de kan ta med den strukturen Scrum-boarden givit oss och uppskattar även möjligheten att i realtid kunna uppdatera projektets status som når ut till alla medlemmar. Oavsett om man genomför IT-projekt eller inte finns det definitivt en poäng i att dela upp projektet i mindre tasks och strukturera upp uppgifterna till olika deltagare.

3.3 Sprinter

Ett utav de viktigaste agila verktygen vi har utnyttjat är att dela upp projektet i sprinter. Totalt sett har 8 sprinter genomförts samt en sista "inofficiell sprint" där projektet presenterats samt att slutgiltiga reflektioner sammanställts. Sprinterna har löpt på under hela projektets gång och strukturerades tidigt upp (kring vecka två i projektet) från torsdag till onsdag.

Sprinterna har inletts med ett startande möte på torsdag och avslutas under onsdagen med att alla gruppmedlemmar har sammanställt en group reflection samt varsin individual reflection som laddats upp på det gemensamma GitHub-repo.

Att arbeta i sprint har definitivt varit en utmaning för vår grupp. Att vi dessutom valde att lägga upp våra sprinter från mitten av veckan till nästa mitt av veckan har gjort det bitvis "hoppigt" och svårstrukturerat, speciellt med tanke på flertalet röda dagar och långhelger. För framtiden kommer vi nog ta med att det skulle varit lättare att planera sprinter om det följde en vanlig kalendervecka utan en helg

i mitten samt att planera bättre kring lediga dagar så att samma mängd arbete fortfarande kan produceras i sprinten. Det finns även visst resonemang om att det bör vara en mindre viloperiod mellan sprinterna, något som naturligt hade kunnats ge om sprinterna hade planerats att följa kalenderveckor och vila ges under helgen.

Inför framtida projekt ser gruppen att det finns vissa fördelar med att jobba i sprinter. Finns det ett uttalat antal timmar som skall produceras i varje sprint gör det förhoppningsvis att ett projekt kommer igång redan från start och att den nedlagda tiden blir hyfsat jämt fördelat. Dock finns det en del administrativa aspekter som tillkommer om man arbetar efter en sprint-modell (bl.a. retrospectives och stand-up meetings). Dessutom är det en utmaning att anpassa varje sprint efter tid att tillgå under olika delar av projektet samt att det krävs att alla parter skall kunna närvara vid kontinuerliga stand-up meetings.

3.4 Stand-up meetings

Varje dag på en sprint bör innehålla ett snabbt stand-up meeting, där definitionen innefattar att det ska vara så snabbt att alla medlemmar inte ens behöver sitta ner. På grund av andra åtaganden och geografisk distans ansågs detta helt omöjligt för vårt projekt. Istället lades krut på att genomföra ett längre stand-up meeting (alltså snarare ett sit-down meeting) och resterande information uppdaterades via vår Scrum-board under varje sprint. Alltså blev våra stand-up meeting istället ett sprint planning-möte.

Gruppen såg dock funktionen med ett stand-up meeting och resonerade att vi troligtvis hade arbetat mer effektivt om vi hade fått till fler möten. Detta då vi hade kunnat bolla idéer och hjälpt varandra på ett simplare vis.

För framtiden hade stand-up meetings säkert varit ett bra verktyg. Dock kräver det en del, som sagt att medlemmarna inte har några andra åtaganden som krockar samt att alla befinner sig på samma geografiska plats. Att lägga ner den tiden som krävs för att genomföra ett telefonmöte (eller annan kommunikation) kan inte anses rimligt i jämförelse med tidsåtgången för ett stand-up meeting. Det krävs också att medlemmarna har den tiden undanlagt för projektet.

3.5 Retrospectives

Varje sprint har utvärderats genom att genomföra en retrospective i slutet, i form av en team reflection samt en individual reflection. Strukturen av dessa följde vissa angivna punkter. Alla i gruppen har genomfört alla sina individuella reflektioner och har medverkat på gruppreflektionerna, dock har inte alla gruppmedlemmar medverkat på alla gruppreflektioner då andra åtaganden har kommit emellan. Viktigt att poängtera dock är att minst två medlemmar har skrivit varje reflektion och att alla medlemmar har godkänt den, med möjlighet att modifiera den, innan den har

publicerat.

Just att alla medlemmar inte har medverkat på varenda gruppreflektion har gjort att den har besvarats på lite olika vis från sprint till sprint. Dock visar detta dynamiken i uppgiften samt hur svårtolkade flertalet av de förutbestämda frågorna faktiskt var. Dessutom har inte alla frågor alltid varit relevanta under projektets gång varav att de vid vissa sprinter blev helt utelämnade, till exempel var det svårt att besvara frågor om kodkvalité i början av projektet när ingen kod ännu blivit producerad.

Överlag har retrospectives fungerat bra för gruppen och alla håller med om att det finns viss relevans i att kontinuerligt utvärdera projektet. Dock har antalet frågor som vi har behövt reflektera om varit många, och enligt vissa för många. Vi ser alla relevansen om att reflektera om dessa frågor i ett slutgiltigt skede men faktumet att vi inte producerat någonting relevant till flertalet frågor vissa sprinter inverkade snarare stress och hopplöshet i gruppen.

Oavsett tror vi det har varit bra att hela gruppen fått sammanstråla för att utvärdera sprinten. Dock tror vi att det hade gett mer om vi hade fått lägga upp den utvärderingen mer självständigt samt att vi borde haft en tidsgräns, såsom max 1h, så att dessa utvärderingar inte har tagit alltför mycket av värdeskapande tid.

4
Resultat

I detta kapitel beskrivs vad projektet resulterade i med avstamp i avklarade user stories, applikationens design struktur, kodkvalité, KPI:er samt acceptanstest.

4.1 User Stories

Vid initieringen av projektet var konceptet user stories nytt för oss. Under Lego-Scrum-övningen erhölls en klarare bild av hur de bör vara uppbyggda och vad deras syfte är. Under en föreläsning presenterades skillnaden mellan “Epics”, “User Stories” och “Tasks” och hur de korrelerar med varandra. En “Epic” består av flera “User stories”, som i sin tur kan bestå av flera “Tasks”. Vikten av att fokus ska ligga på att skapa värde belystes och att man genom “Slice-the-Elephant”-metodik kan skapa små delmål som ändå skapar värde för kunden.

För att få en inblick i vad kunden värderar bokades ett möte in hos förtöjarna för att se hur de arbetar, vilka utmaningar de ställs inför dagligen samt vilken information de behöver ha för att kunna utföra sitt arbete. Utifrån den kunskapen kunde User Stories skapas som sedan bröts ner i tasks. Prioritering gjordes i samråd med PO och förtöjarna. Samtidigt behövdes det has i åtanke att våra programmeringskunskaper var bristfälliga och att vi därmed behövde vara noga med att skapa user stories som faktiskt var möjliga att utföra.

Att genomförbarheten inte existerade upptäcktes på vissa User Stories efter att mycket arbete redan hade lagts ner, vilket i viss mån innebar en känsla av att vi slösat dyrbar tid. Ett exempel på detta är en user story som handlade om att lägga till “Tonnage” för respektive båt. Efter att mycket tid lagts ner för att få detta att fungera, och för att implementera funktionen i appen, visade det sig att PortCDMs databas som tillhandahöll information om båtarna inte innehöll den efterfrågade informationen om tonnaget, utan den informationen behövde i så fall köpas in till ett dyrt pris. Därmed fick den user storyn förenklas. Trots att vi inte kunde slutföra vissa ursprungliga user stories erhölls dock en hel del bra kunskap att ta med sig till andra user stories. Att jobba i motvind kan ofta innebära en bättre inlärning.

Till en början estimerade vi inte tidsåtgången för att slutföra en specifik user story eller task, vilket innebär att de ibland inte kunde slutföras under loppet av sprinten. Under senare sprintar insåg vi dock vikten av estimering och började därmed bryta ner user stories i mindre tasks som, enligt tidsestimeringen, skulle hinnas med under

sprinten. Estimeringen var dock svår att göra på de ingående delarna eftersom vissa moment kunde ta två minuter en dag och 30 timmar en annan sprint.

Som tidigare nämnt gjordes prioritering av user stories i samråd med både förtöjarna och PO, samtidigt som förkunskaper och tidsestimat spelade in. En user story som kunde skapa högt värde på ett förhållandevis enkelt sätt var att skapa ett kommentarsfält, i vilket önskvärd information kunde läggas till. Exempel på önskvärd information kan exempelvis vara tonnaget på båten, som diskuterades tidigare, eller ifall båten har stålvajer i förtöjningslinorna eller inte (eftersom stålvajer innebär att det behövs dubbel mankraft för att utföra förtöjningsjobbet). Den efterfrågade datan om tonnaget finns dock att tillgå på en annan databas – som inte är kopplade till appen – och med hjälp av kommentarsfältet kan förtöjarna, eller andra aktörer, lägga till information efterhand och således skapa en egen databas för framtida bruk.

Första gången båten kommer in i hamnen behöver alltså kommentarer om tonnage, stålvajer, telefonnummer till båten etc. läggas in på den aktuella båten, men nästa gång båten anländer till en hamn finns den informationen redan inlagd. PortCDM kan då även få en indikation på vilken information som används frekvent och som saknas och därmed överväga om datan, trots att den är dyr, bör köpas in och läggas till i databasen. På ett relativt enkelt vis kunde därmed stort värde att skapas för både förtöjarna och produktägarna. Kontentan är att stor vikt bör läggas vid applicering av konceptet “Slicing-the-Elephant” och skapa vertikala user stories och tasks, och inte horisontella, för att genom varje slutförd task och user story kunna leverera något som faktistkt skapar värde.

Till framtiden
Till framtida projekt kommer vikten av vertikaltitet vid upprättande av user stories och tasks att belysas för att varje slutfört moment, hur litet det än må vara, ska leverera värde till produktägare och slutanvändaret. User stories fungerar därmed som ett smidigt sätt att möta efterfråga utan att behöva skapa formella, tidskrävande, kravspecifikationer, som dessutom kräver mycket adminstrativt arbete vid uppdatering av krav. Användning av user stories istället för traditionella kravspecifikationer ligger i linje med ett snabbrörligt och snabbföränderligt - dvs agilt - arbetssätt och om dessa upprättas vertikalt och priorieteras utefter upplevt värde i förhållande till nedlagd tid är user stories mycket bra.

4.2 Applikationens design

Strax efter att vi fått igång en fungerande arbetsmiljö i gruppen, kartlades appen för att förstå applikationens utformning och initiala design (se Appendix 1). Därefter låg prioriterat fokus på att skapa värde för PO och förtöjarna genom att utveckla appen till det bättre. Prioriterad user story var att jobba med ett kommentarsfält och utveckla sidan ”vessel info” till det bättre. Under arbetsgången skickades en mockup till förtöjarna för att få feedback på design och utformning. Mockupen skickades som en film där hela funktionen med kommentarsfältet illustrerades, i figur 4.1 presenteras en bild som är ett urklipp från filmen. Den feedbacken vi fick var

främst att det såg bra ut och att vi var inne på rätt spår. Vi utvecklade produkten så att den liknade vår mockup.

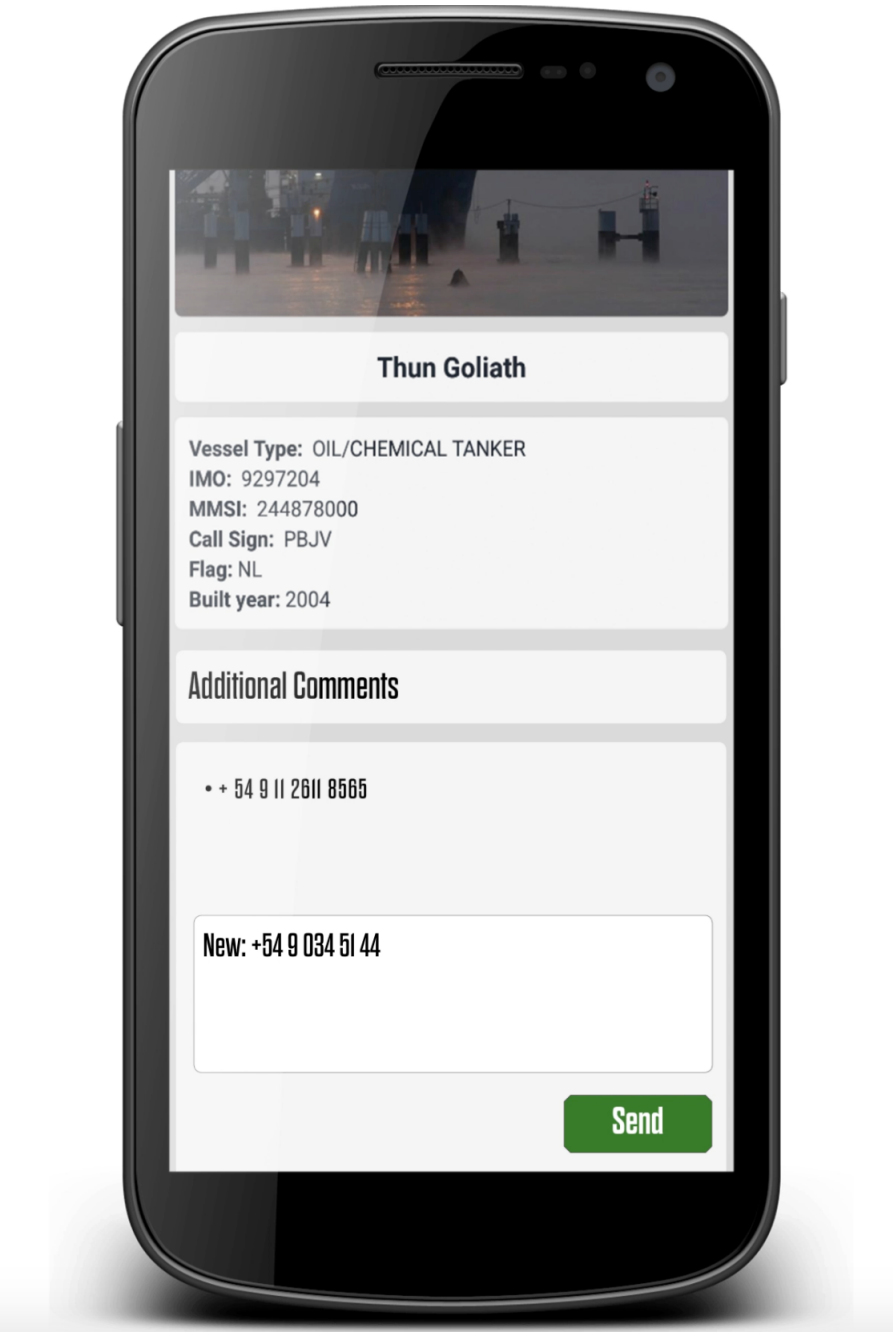

Figure 4.1: En del av den mockup som skickades till PO och förtöjarna för feedback

Med vår begränsade tid och kunskap när det kommer till kodning utvecklade vi inte appen enligt något speciellt designmönster när det kommer till kod. Detta kan ha påverkat kodkvalitén och försvåra om någon extern part skulle fortsätta arbeta med koden.

Vi valde att använda samma css-style som fanns sedan tidigare för att hålla appens design konsekvent. Vi experimenterade alltså inte med något nytt gränssnitt, vilket hade kunnat vara aktuellt om fokus hade legat på att jobba mer mot användarvänlighet. Vi hade absolut kunnat jobba mer med att få feedback från slutanvändarna och testa förändringarna på olika användare för att uppnå bättre användarvänlighet. Vilket är en lärdom vi tar med oss till framtida projekt. Nu vet vi hur vi gör tester och att de är värdefulla för att få feedback och driva projekt i rätt riktning.

Till framtiden
Om vi hade vidareutvecklat appen hade det definitivt funnits en poäng i att vidare efterforska om vilka designmönster som hade kunnat vara relevanta samt att utforma funktionerna efter det. Även, som nämnt, att vidare testa funktionerna och dess utformning hos slutanvändarna. Då hade det kanske även varit relevant att se över hur appen faktiskt är designad och vidare utforma hur den kan förbättras med avseende på hur förtöjarna använder den. Bland annat diskuterades det att skala ner appen och ta bort onödig funktionalitet. Men även att förstora det som slutanvändaren nyttjar i appen.

4.3 Applikationens struktur

Den strukturella överblicken för appen förtydligades med en kartläggningen av appen, se Appendix 1. Det gjordes i ett tidigt skede så att projektgruppen lättare kunde överblicka upplägget på den givna appen. Den initiala tanken var att även kunna presentera den slutgiltiga appen med en uppdaterad version för att lättare vissa PO eller andra intressenter om vad som har skapats. Förändringarna i applikationen var inte tillräckligt omfattande för att motivera en ny kartläggning.

Vad som hade varit bra initialt, hade varit att djupdyka i appens struktur när det kommer till kod. För att lättare förstå uppbyggnaden av appen och vilka klasser som integrerar med varanda. Ett mer specificerat UML diagram hade därför varit användbart och säkerligen, i högre tackt, hjälpt gruppen framåt i utvecklingen.

Vi som grupp har alltså inte gjort någon omfattande strukturell överblick, genom UML-diagram, utan istället nyttjat den givna informationen om appen. Detta har skett bland annat via devtools.portablecdm.eu där officiell information från appens skapare har publicerats. Dessutom har vi som sagt gjort en inofficiell kartläggning av appens funktioner, se Appendix 1.

För att få grepp på klasstrukturen och hur metoder relaterade till varandra i appen togs även hjälp av handledare som visade på strukturen i Redux som bygger på actions och reducers. Detta gjorde att vi kunde använda existerande funktioner som skal för att kunna bygga vår kommentarsfunktion.

Till framtiden
Vi har insett att vi definitivt bör göra fler UML-diagram vid liknande projekt i

framtiden, framförallt om man jobbar med en ny programvara som man inte själv har varit med att skapa. Oavsett om man faktiskt gör ett riktigt UML-diagram kan det alltid vara bra att göra en kartläggning av strukturen inom det man faktiskt arbetar inom. Detta förtydligar även om olika gruppmedlemmar har olika bild av hur appen faktiskt fungerar och är uppbyggd.

4.4 Kodkvalité

Kodens kvalitet fastslogs i projektets början som ett framgångskriterie, se Framgångskriterier.

Kodens faktiska kvalitet testades i slutet av projektets gång med Deepscan. Detta beror på att de ändringarna som gjordes till en början inte ansågs vara tillräckligt omfattande för att behöva testas samt att kod först började produceras efter sprint 5.

Sökningen resulterade i 214 errors och ”poor code quality”. Det framkom dock också att nästan alla av dessa fel ligger i originalkoden och inte i det som vi själva utvecklat. Endast ett av de påträffade felen låg i den nyutvecklade koden och handlade endast om en variabel. Vilket gör att om endast den nyligen tillagda koden utvärderas så anses den ha god kodkvalitet.

En trolig anledning till detta är att koden som vi skrivit är förhållandevis simpel och därav inte riskerar att frammana substantiella problem. Hade scopet för utveckling varit större hade iterativ granskning av kodens kvalitet varit mer aktuell. Exempelvis genom att arbeta med test driven development och där först kod som fungerar skrivs och sedan revideras till att ha god kvalitet med hjälp.

FindBugs som planerades användas för att utvärdera kodkvalité blev istället ett verktyg i själva kodandet då detta program var god hjälp vid identifiering av fel och problem i koden. För det ursprungliga ändamålet, alltså att utvärdera kodvalité, blev alltså DeepScan ett bättre verktyg.

Till framtiden
För framtiden ser vi att utvärdera kodkvalité kan vara relevant om stora mängder kod produceras. Vi ser även att det skulle fungera ännu bättre om all kod har producerats av samma team då detta kan ge en bättre bild av teamets faktiska output och att bedömning inte delvis sker på kod som någon annan producerat.

FindBugs har fungerat mycket bra i syftet att just hitta fel löpande vid kodskriving och är ett bra verktyg som kan komma att användas i framtiden vid större projekt.

4.5 KPI:er

I början av projektet valdes följande tre KPI:er:

- *Quantity of Code*: Hur många rader kod som produceras. Detta för att kunna jämföra mellan olika veckor hur stort värde i form av kod har tillförts, detta ansågs även ge en överblick över arbetsbördan under projektet.
- *Individual Contribution*: Hur mycket tid vardera person har lagt ner. Detta dels för att ge en indikation att vi alla ligger runt 20 h/vecka samt att det kan ge en indikation om hur mycket tid vardera person behöver lägga ner inför varje vecka. Se mer under avsnittet för kodkvalité för att se vilka verktyg som utnyttjats.
- *Quality of Code*: Kvalitén av koden producerad. För att på något sätt även kunna mäta kvalitén av koden i relation till kvantiteten.

Dessa KPI:er valdes främst med avseende på att ett IT-projekt kommer genomföras med en hel del produktion av kod. Det blev dock ganska snabbt tydligt för oss att vi inte skulle producera så mycket kod de första veckorna samt att det primära fokuset faktiskt inte borde ligga på tillförsel av rader kod. Dock insåg vi detta först i den sjätte sprinten då vi beslutade att ändra vår första KPI - Quantity of Code - till att istället fokusera på upplevd framgång - Perceived Succes.

- Perceived Success: För att få en överblick över hur varje individ i gruppen känner hur de har lyckats i sprinten införs denna KPI där ett värde mellan 1-5 skall väljas av varje gruppmedlem. Ett medeltal av detta ger även en indikation av hur hela gruppen har presterat.

I början av projektet kunde varken andelen kod och kvaliteten av kod mätas då ingen faktiskt kod producerades. Istället lades fokus på hur mycket tid som spenderats på kursen vilket varierade från sprint till sprint. Överlag har gruppen lagt mellan 15 - 20 h per person och sprint under hela projektets gång. Det finns inte heller någon indikation om att tiden spenderat har varierat nämnvärt under projektets gång eller mellan individer. För vidare informarion, se Spenderad tid.

I den femte sprinten började kod att produceras och de andra två KPI:erna började först då bli relevanta. Kodkvalitet mättes dock först i vecka 7 och 8 via FindBugs som ansågs som ett bra verktyg i perspektivet att tidigt hitta problem och fel i koden snarare än att mäta kodens kvalité. Med detta i åtanke har alltså denna KPI inte varit mest relevant för projektet och skulle helst ha bytts ut mot något mer relevant. Detta är dock enkelt att säga nu när projektet är över och då vi faktiskt var övertygade om att vi skulle producera mer kod i projektes början än vad som slutligen gjordes.

Perceived Success blev en mycket bra KPI för gruppen och gav en bra överblick över varje sprints resultat. Dock har vi ju som sagt enbart beaktat detta KPI under de tre senaste sprinterna vilket inte ger en fullständig överblick över projektet.

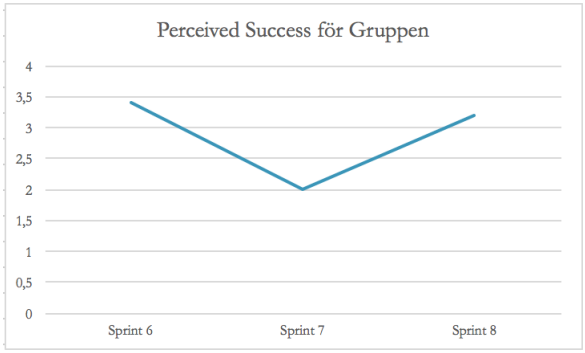

Figure 4.2: Presenterar resultat från KPI:et ”Percieved Success” över sprint 6, 7 och 8.

Till framtiden
Till framtiden ser vi definitivt fördelen med att nyttja vissa KPI:er då detta kan ge bra överblick över ett projekt samt bra underlag för att kunna utvärdera ett projekt i efterhand. Dock har vi definitivt insett vikten av och svårigheten i att välja rätta och relevanta KPI:er. Oftast behövs det en viss insikt och erfarenhet av ett projekt innan man kan inse vad som faktiskt är relevant och mätbart. Som rekommendation inför framtiden skulle vi då ta med oss att det kan vara bra att konstant utvärdera KPI:ernas relevans för projektet, något vi hade behövt göra mycket tidigare.

Dock är vi mycket nöjda med KPI:en Perceived Success då vi känner att den är så pass generell så att den kan utnyttjas i de flesta projekt samt att den ger en bra överblick för gruppens mående samt upplevda produktivitet.

4.6 Acceptanstest

Acceptanstester är något som under projektets gång varit svårt att definiera. Både på grund av en saknad av förkunskaper inom ämnet, vilket gjorde att vi inte var helt säkra på hur ett acceptanstest skulle skapas samt vad Definition of Done (D.o.D.) för de olika delar faktiskt skulle vara.

Testerna är något som har utvecklats under tiden med olika syften men fokuserar framförallt på två områden, dels om vi valt rätt koncept som uppfyller behovet hos slutanvändaren och tillför värde för PO. Dels om den producerade koden faktiskt uppfyller de funktioner som de är tänkta att göra.

För att kontrollera så att våra koncept och idéer fyller ett behov hos användaren använde vi oss av mockups som vi skickade till AB Klippans Båtmansstation innan utveckling. Vår kontaktperson där kunde då komma med feedback på vad vi ville utveckla och vi kunde på så sätt konstatera att idéerna vi hade var rimliga och utvecklingen kunde påbörjas.

För att sedan sätta detta i arbete och arbeta agilt med utvecklingen presenterades vår software till PO. Där kunde vi få feedback på att det vi utvecklar hela tiden

tillför värde och ha möjlighet att revidera vårt upplägg om detta inte var fallet eller vi var på väg i fel riktning. Under vår tid var detta inte ett problem och utvecklingen fortsatte hela tiden åt samma håll.

Svårigheterna med acceptanstester har för oss istället kommit till själva koden. Brist på kod-kvantitet har gjort det svårt att utforma och använda tester för koden. Dvs, när en stor del av arbetet gått åt till att försöka sätta sig in i existerande kod och försöka förstå hur vi ska bygga ut applikationen, och det är svårt att testa kod när det inte finns någon.

När vi faktiskt har kodat något har bristande förkunskaper gjort det svårt att definiera acceptanskriterier för kodkvalitet. Istället har acceptanskriteriet för koden varit att den uppfyller funktionen den är ämnad att göra. Effektiva datastrukturer och algoritmer har inte tagits hänsyn till.

Under sista sprinten när vi arbetade med kommentarsfältet började vi arbeta med test driven development. Målet var att skapa en lista som användaren kunde lägga till element i och att listan sedan skulle skrivas ut rad för rad i en container. Genom att skapa testfunktioner som utnyttjade terminalen kunde vi först arbeta mot målet att element lades till i listan korrekt. Efter detta kunde vi förflytta fokus till att kommentarerna skulle skrivas ut ordentligt. Alltså, genom att skriva testfunktioner som representerade delar av det slutliga målet kunde mindre tasks skapas som gjorde det lättare att ta sig till det slutgiltiga målet.

Till framtiden
Inför framtida projekt ser gruppen relevansen av Acceptance tests och har i viss mening börjat förstå vad de innebär. Omfattningen av gruppens kodande har gjort att acceptance tests i förhållande till kodkvalitet inte kunnat göras men detta är något som kommer med mer erfarenhet och mer kod som faktiskt kan testas. Gruppen är dock överens om att en gemensam överenskommelse av kodstandard behöver upprättas för att kunna kunna arbeta gemensamt. I vårt fall blev denna kodstandard endast “fungerande”.

Fortsättningsvis kommer även TDD användas för att möjliggöra nedbrytning av uppgifter ytterligare. detta kombinerat med tydliga riktlinjer för vad som anses vara godtagbar kvalitet på kod gör arbetet med D.o.D. smidigare och möjliggör ett gott samarbete inom arbetslaget då det går att bygga vidare på varandras kod.

5
Diskussion

I följande kapitel förs en diskussion kring våra veckovisa utvärderingsmöten, framgångskriterier, spenderad tid samt projektets förhållande till litteratur.

5.1 Sprint Review

Våra sprint reviews har sett lite olika ut genom veckorna. Vi har tolkat sprint review som ett tillfälle för oss att diskutera allmänt kring hur nöjda vi är/var med sprinten. Inledningsvis var det kanske lite trögt. Men efter Lego-workshopen och efter att gruppen satt sig in i uppgiften blev det allt bättre även om det inledningsvis rådde en viss frustration kring de tekniska aspekterna.

Eftersom att det tekniska löstes individuellt haltade också vårt agila arbetssätt något och vi kände oss inte helt nöjda. När allting börjat fungera blev också vårt arbetssätt effektivare. Vi använde oss effektivt av Scrum metodik och började beta av user stories. Något vi aldrig riktigt lyckades med är tidsestimering, där våra estimat var helt fel i de flesta fall. Till slut lyckades vi ändå med det vi hade i vårt scope, trots en hel del motgångar. Men som vi skrev i reflektion 8 ”Det kan tyckas att detta innebär att vi misslyckats, men vi anser att vi på vägen lärt hos en hel del”.

Till framtiden
Om vi nu i efterhand ska reflektera kring vad vi kunde gjort annorlunda när vi gjorde våra retrospectives så tror vi att insikterna från vecka 8 redan borde finnas med från start. Vi satt många timmar och försökte felsöka appen för att få allting att starta och när man misslyckas med något så känns det som att det är bortslösad tid. Det tog dock 8 veckor för oss att förstå att det är en del av inlärningsprocessen - och en nödvändig sådan. Vi tror att genom att tidigt skapa en förståelse kring att “misslyckas också är att lyckas” så fås ett mer effektivt arbete och framförallt minskas frustrationen i gruppen.

Hur detta ska åstadkommas är lite svårare att svara på. Det är snarare en fråga om erfarenhet som vi förhoppningsvis tar med oss till nästa projekt. Vi tror också att erfarenheten från ett processorienterat projekt också kommer leda till minskad frustration vid framtida projekt. Ett hjälpmedel för att kunna lösa detta ändå kan vara t.ex. genom effektiv användning av en Scrum-board via Trello.

Trello är mer eller mindre uppbyggt på att slutföra olika uppgifter och kan således ses

som ett produktorienterat hjälpmedel. Detta förstärks ytterligare om uppgifterna får rubriker såsom: “Se till så att kommentarer sparas lokalt”. Här tror vi istället att vi med fördel kan använda oss av en processorienterad approach när vi “slice the elephant”. Uppgifter såsom “Skapa dig en förståelse kring olika sätt att lagra information vid app-kodning” hade kunnat vara en mer processorienterad uppgift. På så sätt hoppas vi kunna minska frustrationen inom gruppen vid varje sprint review, vilket förhoppningsvis leder till en bättre process och slutprodukt.

5.2 Framgångskriterier

När projektet initierades så enades projektgruppen att anpassa målen med de erfarenheter som fanns sedan tidigare. Framgångsfaktorer för oss var därför att utveckla fungerade attribut till appen och hålla god kodkvalité genom en strukturerad process och upprätthålla bra gruppdynamik.

Att utveckla *fungerade attribut*, kan vara allt ifrån att ändra design, lägga till meningar i appen till att bygga funktioner. Något som kan anses vara en vag definierad eller relativ enkel framgångsfaktor men för oss som grupp, där alla var nybörjare med javascript som språk, ansågs som en viktig framgångsfaktor. Det är viktigt att se att även det lilla räknas och att motivera alla gruppmedlemmar att försöka och sätta mål som matchar erfarenhet. Att komma igång med programmen och bli bekväm med arbetsmiljön, inkluderat att samarbeta på ett repo och att få upp appen i emulatorn var framgång i sig. Under projektets gång klarades svårare och svårare moment av och vi landade i ett resultat som alla är väldigt stolta över.

För framtida projekt, nu när vi är mer bekväma i arbetsmiljön, är vi redo att definiera specifika typer av attribut som skulle anses vara framgångsfaktorer och ha delmål på vägen. Fortfarande är de små stegen viktiga men man skulle kunna definiera tydligare att ”det här är slutmålet” och ha milstolpar på vägen. Detta skulle kunna visualiseras i en tidslinje och i slutet kan man mäta hur framgångsrikt projektet blev i form av hur nära slutmålet man kom. Det i sin tur skulle ge en överskådlig bild över ”var är vi på väg?” och ”hur tar vi oss dit?”.

God kodkvalité initierades för att utveckla hållbart och se till att man kan bygga vidare på det vi skapar. Vi hade goda intentioner men ställer oss rätt kritiska till denna framgångsfaktor då vi i projektets början var nybörjare och inte riktigt visste vad god kodkvalité innebar. Vår avsaknad av expertis tillsammans med mängden producerad kod gör det svårt att mäta kvalitén på vår kod. Dessutom har vi undersökt kodkvalitén som fanns i appen vid projektets början med hjälp av ”Deepscan” verktyg och fick insikten att den i dagsläget är låg och att det är grunden som vi bygger på snarare än att förändra redan existerande kod. Om vi skulle fortsatt med att eftersträva god kodkavlité skulle fokus behöva skiftas till att läggas på att revidera koden som finns i dagsläget snarare än att utveckla nya attribut. Enligt den felsökningen vi har gjort finns 214 problem som inte lösts med koden. Här skulle man kunna jobba igenom koden och eftersträva att lösa dessa problem och på så sätt uppnå god kodkvalité.

Att ha en *strukturerad process* valdes som framgångsfaktor naturligt då vi under projektets gång skulle arbeta agilt. Struktur är fundamentalt för att lyckas jobba likt Scrums ramverk och med kortare sprintar. Vi har skapat en Scrum board i Trello för att strukterera vår backlog och inför varje sprint uppdaterades denna. Detta arbetssätt var nytt för samtliga och vi lyckades hålla struktur i form av att veckovis köra sprintar, uppdatera vår Scrum board och bryta ner våra user stories i mindre uppgifter och därefter reflektera över processen. Något som utmanade vår struktur var vår förmåga att estimera tidsåtgång för olika uppgifter samt att bryta ner uppgifterna i mindre delmoment. Orsaken till detta var bristande kunskap och en lärdom som vi tar med oss är att det är bättre att ha väldigt små uppgifter som kan distribueras ut för att kunna jobba mer parallellt och räkna med att hela gruppen rör sig i rätt riktning. I framtiden kommer mer tid läggas på detta för det kommer i sin tur främja en ännu mer strukturerad process.

Att ha *god gruppdynamik* ansåg vi gemensamt vara viktigt för att arbeta effektivt tillsammans. God gruppdynamik främjar en transparens i hur det går och hur alla mår. Vi har generellt haft bra gruppdynamik. Mot slutet av projektet gick det lite upp och ner, frustrationen över motgångar kodmässigt lyste igenom. Samtidigt som det var ett stresspåslag när alla inte kunde lägga 20 timmar per vecka på kursen vilket påverkade gruppdynamiken. Märkbart är hur gruppdynamiken går hand i hand med att ha en strukturerad process. Lärdomen vi tar med oss är att jobba mer med strukturen för att kunna sätta upp tydliga förväntningar på oss själva när det kommer till kodningen och på varandra när det kommer till tidsplanering.

Till framtiden
Lärdomar som vi tar med oss från projektet efter att utvärderat framgångskriterierna är att anpassa dem ännu mer efter gruppmedlemmarnas erfarenheter. För att göra det krävs en bättre genomgång och diskussion kring vilka resurser vi har att jobba med och vad vi vill lägga fokus på vid projektets start.

Sammanfattningsvis tar vi med oss till framtida projekt att bryta ner uppgifter i flera delar och se värdet i att ta små steg framåt. Det kommer både bidra till effektivare utveckling samt förenklad tidsestimering vilket i sin tur leder till bättre förutsättningar för en mer strukturerad process. Om projektgruppen upprätthåller en bra strukturerad process kommer även gruppdynamiken påverkas positivt.

5.3 Spenderad tid

Tiden som lagts ner på kursen har genom veckorna varierat något. De första fem veckorna dokumenterade vi att alla lagt ner runt 20 timmar i veckan. Därefter gick timmarna ner något till runt 15 timmar under vecka 6 och 7. Anledningen till att timmarna sjönk något där berodde dels på deadline för kandidatarbetet och dels på röda dagar. Detta gjorde att vi kanske inte hann genomföra allt vi ville genomföra, även om vi gjorde lite av ett ryck sista veckan och ökade antalet timmar igen.

Till framtiden
Till framtida projekt tror vi att det är väldigt viktigt att antalet timmar varje vecka är samma. Om det är 15, 20 eller 40 timmar påverkar självklart, men för att kunna planera sitt scope och åtagna uppgifter måste nedlagda timmar vara konstant. Således anser vi att vi kanske borde börjat lite lugnare de första veckorna och hållit oss till maximalt 20 timmar och på så sätt frigjort tid till andra kurser. Då hade vi kunnat fortsätta vidare i samma takt även under vecka 6 och 7 som vi gjorde under resterande veckor.

För att lyckas med detta bör arbetsbördan ses ur ett mer helhetligt perspektiv, där även andra kurser eller arbete räknas med. På så sätt kan vi utforma arbetet så att den krävda nedlagda tiden dels är rimlig i förhållande till kursens omfattning, men även att tiden som läggs ner är konstant så att planering och tidsestimering kan fungera bättre.

5.4 Litteraturförankring

Vår relation till litteratur och andra informationskällor har skiftat med tiden. Inledningsvis låg ett stort fokus på att sätta sig in i agil utveckling och då framförallt metodiken Scrum. Relationen med vad vi gjorde dessa sprinter och vad vi läste var väldigt stark. Alla började med att läsa hela, eller delar av Kniberg, H. (2015) Scrum and XP from the Trenches - 2nd Edition. Vid sprintens slut gick vi igenom litteraturen för att alla skulle vara på samma kula på hur vi skulle komma att använda Scrum. Detta har haft stor betydelse för oss som grupp, då vi som studenter från Industriell Ekonomi, med begränsade kunskaper inom programmering snarare skulle komma att ha ett processorienterat fokus än ett resultatorienterat.

De fyra första veckorna drevs egentligen två linjer. Den ena var processorienterad där vi läste, gick på föreläsningar och gjorde bla. lego-Scrum-övningen. Dessutom lades stort fokus på att få förståelse för hur versionshantering genom Git funkar, hur man utvecklar en app samt hur vi ska förhålla oss till alla nya program som introducerades via olika typer av tutorials via video eller artiklar. Den andra linjen handlade om att försöka få alla program att fungera så att vi skulle kunna starta appen. För att kunna göra det söktes framförallt information via internet på sidor såsom stackowerflow, men även genom andra grupper och över kursens gemensamma slackgrupp.

Därefter inleddes en period där vi satte in oss i slutanvändarens problem och kontext. Genom två studiebesök hos slutanvändarna på AB Klippan Båtmansstation och en guidad tur genom hamnen lärde vi oss väldigt mycket om det dagliga arbetet, och vad som hade skapat värde för förtöjarna. Därefter hade alltså samtliga gruppmedlemmar en god förståelse av Scrum metodik och agil utveckling, alla hade fått appen och verktygen att fungera och alla förstod våra slutanvändare och deras bransch. Då gick vi vidare till att försöka realisera de user stories som sattes upp under tidigare veckor i samråd med förtöjarna för att skapa värde.

Till framtiden
I slutändan ledde denna taktik med en uppdelning av fyra olika litteraturområden till ett bra resultat. Vad som hade varit intressant att ändra till framtida projekt är att vi istället utför litteratursprintar, där varje person läser något inför veckomötet och sedan återberättar detta för resten av medlemmarna. På så sätt skulle vi kunna bygga en bredare bas av information och litteratur. Denna uppdelning hade kunnats implementeras på det mesta. Exempelvis om en person satte sig med att få allting att fungera tekniskt, så kan de andra bara klona repot när det fungerar, medan andra läser på hur mooringverksamheten fungerar.

Vi tror att nyckeln till att lyckas med detta ligger i att se till så att gruppmedlemmarna vid projektets start ser över hela projektperioden och förutom att dela in projektet i veckovisa sprintar också i olika större sprintar. För detta projekt var det då Scrum, teknik, slutanvändare och implementation. Om vi hade inlett med att reflektera kring hur dessa fyra områden bör behandlas vid inledningsfasen hade vi kanske kommit igång tidigare och kunnat arbeta mer effektivt.

6
Slutsats

Inledningsvis hade gruppen en del motgångar. Vi hade stora problem med de tekniska aspekterna av appen och att lösa dessa problem skulle ta upp en stor del av kursens tid. Efter att dessa problem var lösta insåg gruppen att en del av våra uppsatta user stories inte skulle gå att genomföra. Men efter att alla gruppmedlemmar satt sig in i Scrum metodologi vände vindarna. Trots vissa tekniska svårigheter hittade vi ett sätt att på ett effektivt sätt skapa värde för samtliga inblandande parter genom ett kommentarsfält och tillhandahållande av ytterligare information.

Insikterna vi tar med oss från detta projekt är många, och av skiftande karaktär. Framförallt så har vi lärt oss mycket om hur ett IT-projekt faktiskt kan se ut, med problematiska starter och många misslyckanden. Vi har även lärt oss parktiskt om agilt utvecklande och Scrum, något man hör mycket om idag och som känns väldigt aktuellt.

Trots att projektets inledning skulle komma att kantas av misslyckanden så tar vi med oss något viktigt från det. Nämligen insikten att vi faktiskt lär oss något när vi misslyckas, och att ett misslyckande paradoxalt nog kan ses som ett steg framåt.

Vidare har vi också fått en förståelse för hur värde skapas för inblandande parter. Genom våra studiebesök på AB Klippan Båtmansstation och Port of Gothenburg har vi fått insikter i hur deras verksamhet och utmaningar ser ut, men också på hur vi kan skapa värde för dem. Att studiebesök ger större insikter i en given verksamhet kanske är självklart, men vad vi lärde oss var att en liten insats från vår del kan skapa stort värde för någon annan.

Avslutningsvis anser vi i gruppen att kursen har varit utmanande och tuff, men att vi i slutändan ändå går härifrån med känsla av glädje och stolthet över vad vi åstadkommit.

7

Appendix 1

Nedan presenteras den kartläggning av appen som genomförts. Appens funktionalitet gås igenom samt innehållet i varje vy. Under respektive bild förklaras vad som gås igenom.

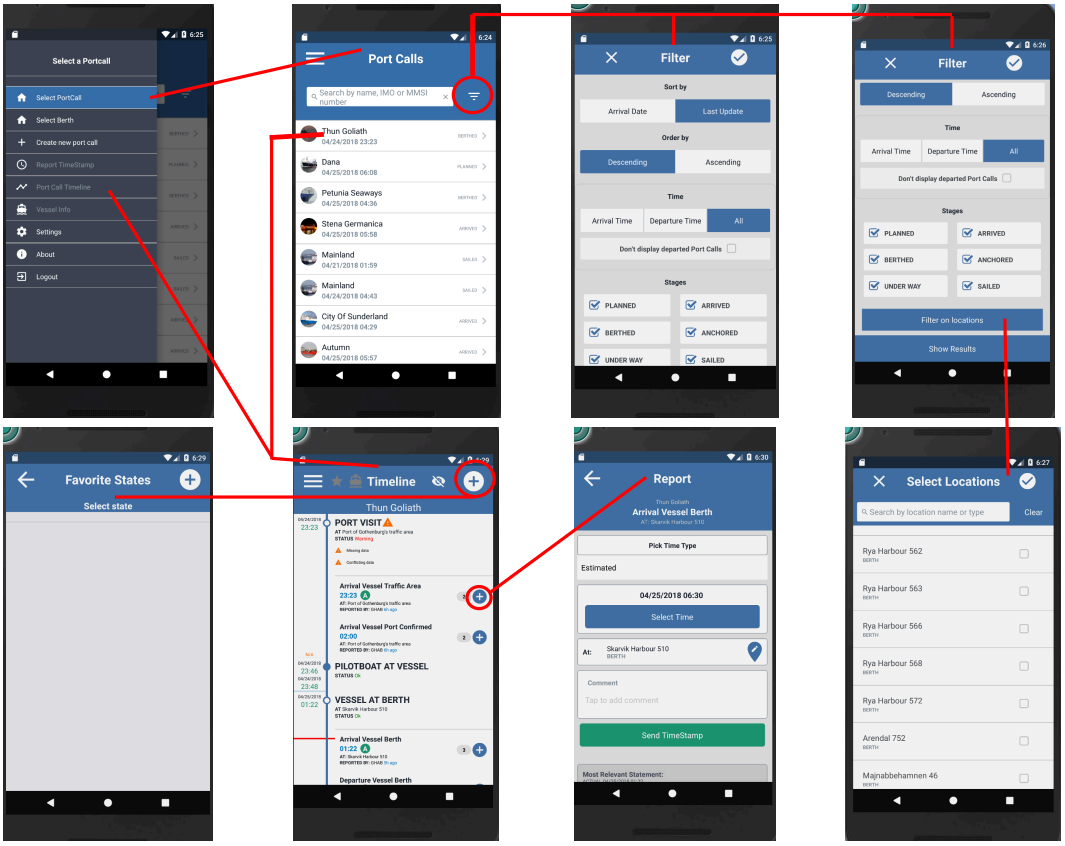

Figure 7.1: Illustrerar vyn för select port call och att man kan filtrera ordningen för båtarna. Samt hur man kan filtrera, genom att bland annat välja 'select location'. Figuren visar även vad som händer när man väljer ett port call och hur vyn för 'timeline' ser ut. Även hur man lägger till information på timeline genom sidan 'report'.

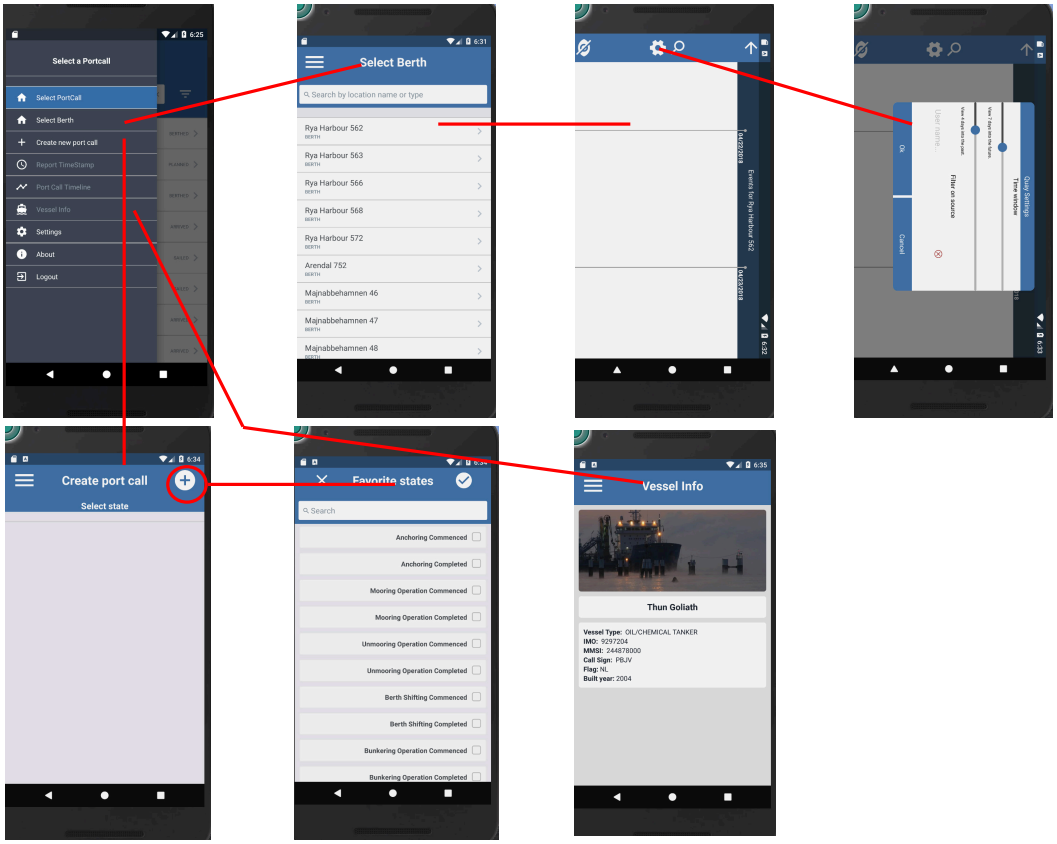

Figure 7.2: Illustrerar menyalternativen, 'Select Berth', 'Create port call' samt 'Vessel Info'. Vid Create Port Call väljs (genom att klicka på +) vilken aktör du väljer att anropa ifrån och på så sätt kan du rapportera timestamps som loggas på tidslinjen från respektive aktör.

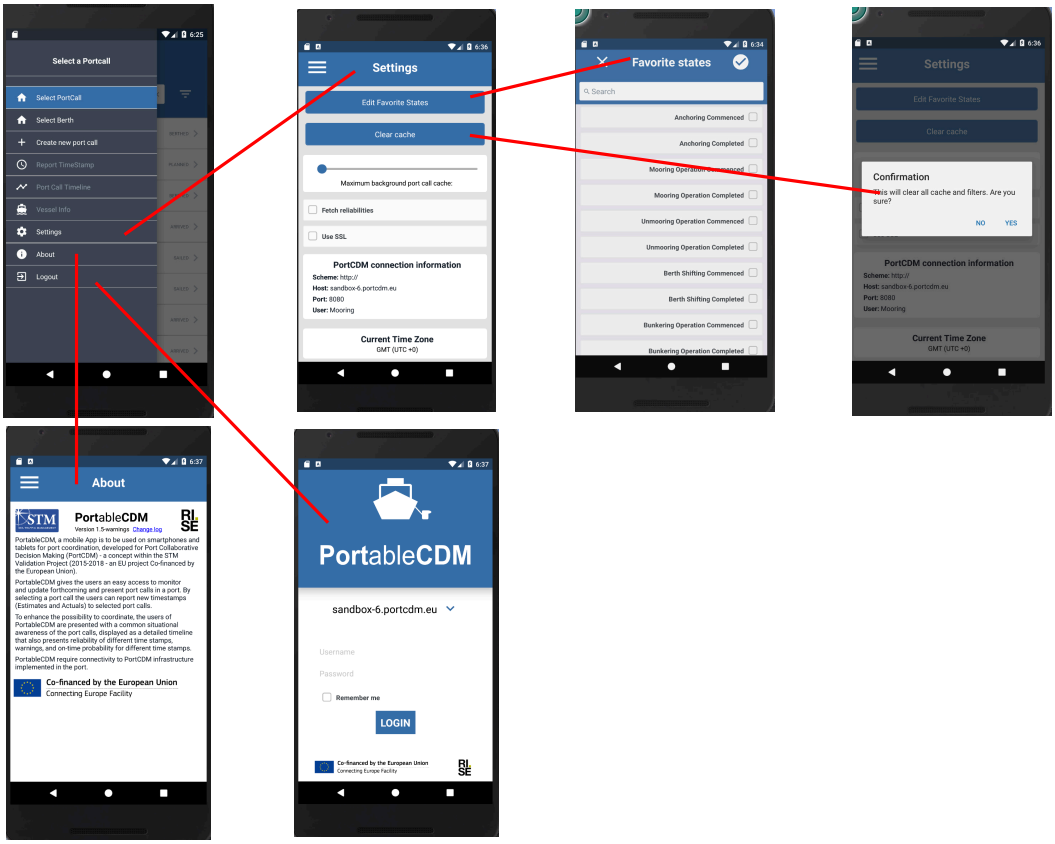

Figure 7.3: Illustrerar menyalternativen 'Settings', 'About' och 'Log Out'. Visar även hur man i inställningar kan ställa in 'favorite state' samt hur man tar bort filter som satts på port call.